Documentación del sistema de código abierto *Opentrad Apertium* de traducción automática de transferencia sintáctica superficial

AUTORES:
Mikel L. Forcada
Boyan Ivanov Bonev
Sergio Ortiz Rojas
Juan Antonio Pérez Ortiz
Gema Ramírez Sánchez
Felipe Sánchez Martínez

COORDINADORA:
Mireia Ginestí Rosell

Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant

28 de marzo de 2006

Índice general

Introducción

La presente documentación describe el sistema *Opentrad Apertium*[1], uno de los sistemas de traducción automática de código abierto creados en el marco del proyecto "Traducción automática de código abierto para las lenguas del estado español". Se trata de un sistema de traducción automática de transferencia sintáctica superficial y está pensado inicialmente para la traducción entre pares de lenguas relacionadas, aunque algunos de sus componentes se han utilizado también en la arquitectura de transferencia sintáctica profunda (*Opentrad Matxin*) que se ha desarrollado en el mismo proyecto para el par español–euskera. *Apertium* traduce actualmente entre los pares español–gallego y español–catalán[2], y puede utilizarse para la construcción de traductores entre otros pares de lenguas relacionadas como danés–sueco, checo–eslovaco, etc.

Los demás sistemas de traducción automática disponibles hasta el momento para los pares `es-ca` y `es-gl` son mayoritariamente comerciales o utilizan tecnología de propiedad, con lo que es muy difícil adaptarlos a nuevos usos; además, utilizan diferentes tecnologías para los distintos pares de lenguas, lo que hace muy difícil integrarlos en un único sistema de gestión de contenidos multilingües.

Una de las principales novedades de la arquitectura que presentamos es que se ha liberado bajo una licencia de código abierto (licencia GNU GPL para los programas y Creative Commons para los datos) y es de distribución gratuita. Esto significa que cualquier persona que tenga los conocimientos lingüísticos e informáticos necesarios podrá adaptar o ampliar el producto para crear un nuevo sistema de traducción automática, incluso para nuevos pares de lenguas relacionadas. Esperamos, por ello, que la introducción de esta arquitectura de traducción automática de código abierto solucione algunos de los problemas mencionados (existencia de diferentes tecnologías para diferentes pares, dificultad de adaptación de las

[1] En adelante, nos referiremos a él simplemente como *Apertium*

[2] Con la denominación *catalán* nos referimos también a la variante dialectal valenciana de este idioma.

arquitecturas de código cerrado a nuevos usos, etc.) y favorezca el intercambio de datos lingüísticos ya existentes utilizando los formatos basados en XML que se definen en esta documentación. Al mismo tiempo, creemos que ayudará a cambiar el modelo de negocio actual en este campo, basado en licencias, por un modelo basado en la prestación de servicios.

Cabe mencionar que es la primera vez que el Gobierno español ha subvencionado un proyecto de código abierto de esta envergadura, aunque la adopción de software de código abierto por las administraciones españolas no es algo nuevo.

En los capítulos de la presente documentación se explican detalladamente las características del sistema Apertium, con la información organizada de la siguiente manera:

- Capítulo 1: **descripción general del sistema** de traducción automática de transferencia superficial y de los módulos de los que se compone.
- Capítulo 2: descripción del **formato del flujo de los datos** que circulan de un módulo a otro.
- Capítulo 3: **especificación de los módulos** del sistema. Para cada módulo se describe: el *funcionamiento del módulo*, el *formato de los datos* sobre los que trabaja el módulo, y los *compiladores* correspondientes. Este capítulo se divide en las siguientes secciones:
  - sección 3.1: *Módulos de procesamiento léxico*, en la que se describe el funcionamiento del analizador morfológico, del módulo de transferencia léxica, del generador morfológico y del postgenerador (apartado 3.1.1), el formato de los diccionarios con los que trabajan estos módulos (apartado 3.1.2) y los compiladores de los mismos (apartado 3.1.3)
  - sección 3.2: *Desambiguador léxico categorial*, en la que se describe el funcionamiento del desambiguador (apartado 3.2.1), el formato de los datos lingüísticos con los que trabaja (apartado 3.2.2.
  - sección 3.3: *Módulo de preproceso del transfer*, en la que se describe el módulo que se ejecuta antes del módulo de transferecia estructural para realizar ciertas operaciones en las unidades multipalabra
  - sección 3.4: *Módulo de transferencia estructural*, en la que se describe el funcionamiento del módulo (apartado 3.4.1), el formato de las reglas de transferencia estructural (apartado 3.4.2).

  - sección 3.5: *Desformateador y reformateador*, en la que se describe el funcionamiento de estos módulos (apartado 3.5.1), las reglas de especificación de formato (apartado 3.5.2) y el proceso de generación de los módulos (apartado 3.5.3)

- Capítulo 4: describe cómo **instalar el sistema** y cuáles son los requisitos para ello, así como la manera de **utilizar el traductor**.

- Capítulo 5: describe cómo se pueden **modificar los datos lingüísticos** del traductor, es decir, los diccionarios, los datos de desambiguación léxica categorial y las reglas de transferencia estructural que se han creado para el español, el catalán, el gallego. Además, aquí se describen brevemente las características de los datos existentes para los tres pares de lenguas del proyecto.

Los ficheros a los que hace referencia este informe se pueden consultar y descargar desde la página web del proyecto en Sourceforge: `http://apertium.sourceforge.net/`. En esta página se pueden descargar los paquetes para la instalación, así como consultar los archivos individualmente, en el repositorio de CVS (`http://cvs.sourceforge.net/viewcvs.py/apertium/`). Los sistemas de traducción automática también se pueden probar a través de internet en `http://xixona.dlsi.ua.es/prototype/`.

Agradecimientos:

El presente trabajo se ha beneficiado de las aportaciones de muchas personas e instituciones:

- El Ministerio de Industria, Comercio y Turismo ha financiado este trabajo a través del proyecto "Traducción automática de código abierto para las lenguas del Estado español" del programa PROFIT, claves FIT-340101-2004-3 y su ampliación FIT-340001-2005-2.

- Trabajadores y becarios de otros proyectos de traducción automática de la Universidad de Alicante: Míriam Antunes Scalco, Carme Armentano i Oller, Raül Canals i Marote, Alicia Garrido Alenda, Patrícia Gilabert i Zarco, Maribel Guardiola i Savall, Javier Herrero Vicente, Amaia Iturraspe Bellver, Sandra Montserrat i Buendia, Hermínia Pastor Pina, Antonio Pertusa Ibáñez, Francisco Javier Ramos Salas, Marcial Samper Asensio y Miguel Sánchez Molina.

- Las empresas e instituciones que han financiado estos otros proyectos de traducción automática: el Ministerio de Ciencia y Tecnología, la Caja de Ahorros del Mediterráneo, la Universitat d'Alacant y Portal Universia, S.A.

- Iñaki Alegria, del grup Ixa de la Euskal Herriko Unibertsitatea, por su lectura atenta de versiones anteriores de este documento.

Capítulo 1

El ingenio de traducción de transferencia sintáctica superficial

Este capítulo describe brevemente la estructura del ingenio de traducción automática de transferencia superficial, basado en el de los sistemas español–catalán interNOSTRUM [2, 5, 4] y español–portugués Traductor Universia [7, 17], ambos desarrollados por el grupo Transducens de la Universitat d'Alacant. Se trata de un sistema clásico de traducción automática indirecta que utiliza una estrategia de transferencia sintáctica parcial similar a la de algunos sistemas comerciales de TA para ordenadores personales.

El diseño del sistema permite obtener sistemas de TA *rápidos* (que traducen decenas de miles de palabras por segundo en ordenadores de sobremesa comunes) y cuyos resultados son, a pesar de los errores, razonablemente inteligibles y fáciles de corregir. En el caso de lenguas emparentadas como las que nos ocupan (español, gallego, catalán), una traducción mecánica palabra por palabra (con un equivalente fijo) presentaría errores que, en su mayoría, se pueden resolver con un análisis bastante somero (un análisis morfológico seguido de un análisis sintáctico superficial, local y parcial) y con un tratamiento adecuado de las ambigüedades léxicas (principalmente de la homografía). El diseño adoptado sigue estas directrices con resultados muy interesantes. La arquitectura de Apertium utiliza transductores de estados finitos para el procesamiento léxico, modelos ocultos de Markov para la desambiguación léxica y procesamiento de patrones basado en estados finitos para la transferencia estructural.

El ingenio de traducción consiste en una *cadena de montaje* de ocho módulos, que puede verse representada en la figura 1.1. Para facilitar el diagnóstico y la comprobación independiente, los módulos se comunican entre sí en formato de texto. De esta manera, siempre que se quiera se

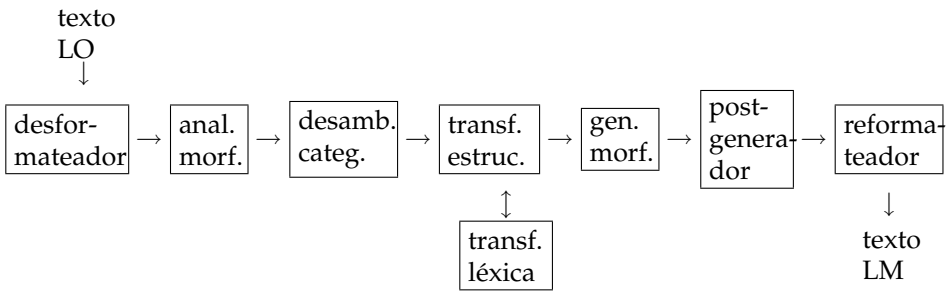

Figura 1.1: Los ocho módulos que forman la cadena de montaje del sistema de traducción automática por transferencia sintáctica superficial.

puede comprobar cuál es la entrada y la salida de los mismos y, cuando se produce algún error en la traducción, examinar por separado la salida de cada uno de ellos para detectar dónde se localiza el error. Asimismo, la comunicación vía texto permite que los módulos se usen de manera aislada, independientemente del resto del sistema de traducción automática, para otras tareas de procesamiento del lenguaje natural, y posibilita la construcción de prototipos con módulos modificados o añadidos.

Se ha elegido que el formato de los ficheros de datos lingüísticos sea XML[1] por su interoperabilidad, por su independencia del juego de caracteres y por la existencia de numerosos útiles y librerías que facilitan el análisis de los datos codificados en este formato. Como bien se afirma en [8], XML es el estándar emergente para la representación y el intercambio de datos en Internet. Las tecnologías en torno a XML incluyen mecanismos muy potentes para el acceso y la manipulación de documentos XML que probablemente tendrán un impacto significativo en el desarrollo de herramientas para el procesamiento del lenguaje de natural y de corpus anotados.

Los módulos de los que se compone Apertium son los siguientes:

- El *desformateador*, que separa el texto a traducir de la información de formato (RTF, HTML, etc.); su especificación se describe en el apartado 3.5.1. La información de formato es encapsulada de manera que los módulos restantes la tratan como blancos entre palabras. Por ejemplo, ante el texto HTML en español:

```
es <em>una señal</em>
```

[1] `http://www.w3.org/XML/`

el desformatador encapsula las etiquetas HTML entre corchetes y proporciona como salida:

```
es [<em>]una señal[</em>]
```

Las secuencias de caracteres entre corchetes son tratadas por el resto de módulos como simples espacios en blanco entre palabras.

- El *analizador morfológico*, que segmenta el texto en *formas superficiales* (FS) (las unidades léxicas tal como se presentan en los textos) y entrega para cada FS una o más *formas léxicas* (FL) consistentes en un *lema* (la forma base comúnmente usada para las entradas de los diccionarios clásicos), la *categoría léxica* (nombre, verbo, preposición, etc.) y la información de flexión morfológica (número, género, persona, tiempo, etc.). La división de un texto en FS presenta aspectos complejos debido a la existencia, por un lado, de contracciones (*del*, *teniéndolo*, *vámonos*) y, por otro, de unidades léxicas de más de una palabra (*a pesar de*, *echó de menos*). El analizador morfológico permite analizar estas FS complejas y tratarlas adecuadamente para que sean procesadas por los módulos posteriores. En el caso de las contracciones, el sistema lee una única forma superficial y da como salida una secuencia de dos o más formas léxicas (por ejemplo, la contracción *del* sería analizada como dos formas léxicas, una para la preposición *de* y otra para el artículo *el*). Las unidades léxicas de más de una palabra (multipalabras) son tratadas como formas léxicas individuales y, según su naturaleza, reciben un tratamiento específico.[2]

  Al recibir como entrada el texto de ejemplo proveniente del módulo anterior, el analizador morfológico proporcionaría como salida:

```
^es/ser<vbser><pri><p3><sg>$[ <em>]
^una/un<det><ind><f><sg>/unir<vblex><prs><1><sg>/unir
<vblex><prs><3><sg>$
^señal/señal<n><f><sg>$[</em>]
```

  donde cada forma superficial es analizada como una o más formas léxicas. Así, *es* es analizada como una FS con lema *ser*, mientras que *una* recibe tres análisis: lema *un*, determinante indefinido femenino singular; lema *unir*, verbo en presente de subjuntivo, 1ª persona del

[2] Para más detalles sobre la el tratamiento de las unidades léxicas de más de una palabra, véase la página 42.

singular, y lema *unir*, verbo en presente de subjuntivo, 3ª persona del singular.

Este módulo se genera a partir de un diccionario morfológico para la lengua origen (LO), cuyo formato se especifica en el apartado 3.1.2.

- El *desambiguador léxico categorial*, elige, usando un modelo estadístico (modelo oculto de Markov), uno de los análisis de una palabra ambigua de acuerdo con su contexto; en el ejemplo utilizado, la palabra ambigua sería la forma superficial *una*, que puede analizarse de tres maneras diferentes. Las formas superficiales ambiguas por tener más de un lema, más de una categoría o representar más de una flexión son muy comunes (en las lenguas románicas, en torno a una de cada tres palabras) y son una fuente muy importante de errores de traducción en caso de elegir el equivalente incorrecto. El modelo estadístico se entrena sobre corpus suficientemente representativos de textos en LO.

  El resultado de procesar mediante el desambiguador el texto de ejemplo proporcionado por el analizador morfológico sería:

  ```
  ^ser<vbser><pri><p3><sg>$[ <em>]^un<det><ind><f><sg>$
  ^señal<n><f><sg>$[</em>]
  ```

  en que se ha elegido la forma léxica correcta (el determinante) para la palabra *una*.

  La especificación del desambiguador léxico categorial se describe en la sección 3.2.

- El *módulo de transferencia léxica*, que gestiona un diccionario bilingüe y es invocado por el módulo de transferencia estructural, lee cada FL en LO y entrega la FL correspondiente en lengua meta (LM). El diccionario contiene un único equivalente para cada forma léxica de la LO; esto significa que no se realiza ningún tipo de tratamiento de la polisemia. Las multipalabras son traducidas como una unidad. En el ejemplo utilizado, cada una de las formas léxicas se traduciría al catalán de la siguiente manera:

  ```
  ser<vbser> ⟶ ser<vbser>
  un<det> ⟶ un<det>
  señal<n><f> ⟶ senyal<n><m>
  ```

Este módulo se genera a partir de un diccionario bilingüe, descrito en el apartado 3.1.2).

- El *módulo de transferencia estructural*, que detecta y trata patrones de palabras (sintagmas) que exigen un tratamiento especial por causa de las divergencias gramaticales entre las lenguas (cambios de género y número, reordenamientos, cambios preposicionales, etc.). Este módulo se genera a partir de un archivo de reglas que describen la acción que debe realizarse para cada patrón. Ante la frase de ejemplo, el patrón formado por `^un<det><ind><f><sg>$ ^señal<n><f><sg>$` sería detectado por una regla determinante–nombre, que en este caso modificaría el género del determinante para que concuerde con el nombre; el resultado sería:

  ```
  ^ser<vbser><pri><p3><sg>$[ <em>]^un<det><ind><m><sg>$
  ^senyal<n><m><sg>$[</em>]
  ```

  La especificación del formato del archivo de reglas de transferencia estructura, que se inspira en la descrita en [5], se encuentra en la sección 3.4.

- El *generador morfológico*, que genera a partir de la forma léxica en lengua meta una forma superficial flexionada adecuadamente. El resultado para la frase de ejemplo sería:

  ```
  és[ <em>]un senyal[</em>]
  ```

  Este módulo se genera a partir de un diccionario morfológico, cuyas características están descritas en el apartado 3.1.2.

- El *postgenerador*, que realiza algunas operaciones ortográficas en LM tales como contracciones y apostrofaciones, y que es generado a partir de un archivo de reglas de transformación con un formato similar al de los diccionarios anteriores. Su formato se especifica en la sección 3.1.2. En la frase de ejemplo utilizada no hay que realizar ninguna contracción ni apostrofación.

- El *reformateador*, que reintegra la información de formato original al texto traducido; el resultado para la frase de ejemplo sería la correcta conversión del texto a formato HTML:

  ```
  és <em>un senyal</em>
  ```

La especificación del reformateador se describe en la sección 3.5.1.

Los cuatro módulos de procesamiento léxico (el analizador morfológico, el módulo de transferencia léxica, el generador morfológico y el postgenerador) utilizan un único compilador, basado en un tipo de *transductores de estados finitos* [4], concretamente en transductores de letras [14, 11] cuyas características se explican en 3.1.3.

Capítulo 2

Especificación del formato del flujo de datos entre módulos

2.1. Introducción

Para enteder y hacer más efectivo el procesamiento de documentos es necesario especificar el formato de la información que circula como texto entre los módulos del traductor. El diseño propuesto (véase sección 1) impone la necesidad de utilizar tres tipos de flujo de datos distintos como se expresa en la figura 2.1.

El formato de flujo está basado en texto para facilitar, entre otras cosas, la tarea de diagnóstico sobre posibles errores del sistema ya que resulta sencillo manipular el flujo para reproducir aquellos fenómenos que se desee probar y modificarlos para comprobar el resultado. Otra ventaja destacable es la posibilidad de comprobar de manera independiente la salida de cada módulo del traductor y, junto a esto, la rapidez en la construcción de prototipos para comprobar su funcionamiento global, la validez de los datos lingüísticos, etc.

Estos flujos de datos son:

- *Flujo de datos con formato:* Es el texto en su formato original, sin ningún otro tipo de marca: XML, texto ANSI, RTF, HTML, etc. Por ser el formato original de los documentos, de este flujo de datos no hay nada que especificar salvo el tipo de formato de que se trata.

- *Flujo de datos sin formato:* Consiste en el texto con *superblancos*, marcas que encapsulan el formato (véase el apartado 3.5.1) y que son tratadas como si de blancos se tratase (con algunas salvedades) por los módulos lingüísticos del sistema. Este formato es el que genera

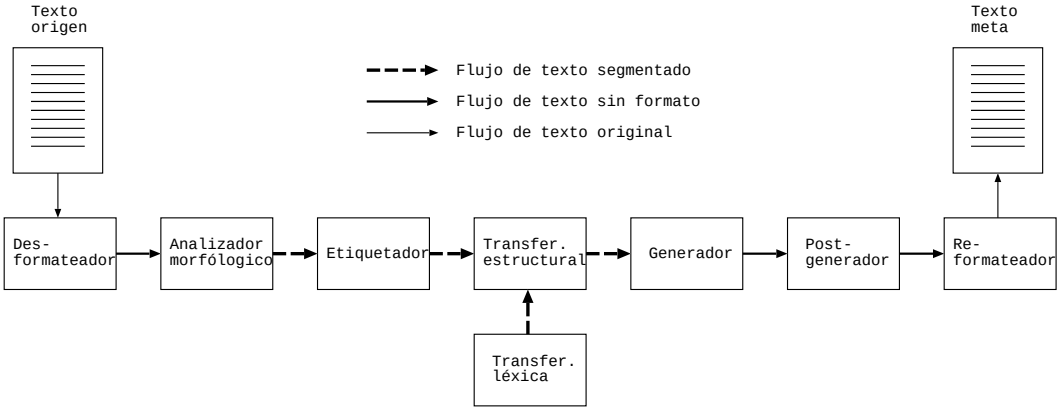

Figura 2.1: Los diversos tipos de flujo de datos en el sistema de traducción automática. Véase el texto para la definición de los diferentes tipos de flujo.

el desformateador, y el que utiliza el reformateador para generar el documento traducido final.

- *Flujo de datos segmentado:* Además de superblancos, este formato tiene delimitadas mediante marcas las unidades léxicas que se debe traducir. Estas marcas son establecidas por el analizador morfológico, y son eliminadas por el generador, que entrega las formas superficiales finales.

A continuación describiremos las caracaterísticas del flujo de datos empleado entre módulos del traductor. A grandes rasgos, se trata de un formato en texto llano con marcas de carácter que poseen un significado especial. Dicho formato está orientado al procesamiento en servidores que traduzcan grandes volúmenes de texto.

Algunos de los formatos que puede tratar el traductor pueden contener bloques extensos de información en formato binario —como por ejemplo RTF, que puede incluir imágenes de mapa de bits—. Para facilitar un procesamiento eficiente de este tipo de documentos se especifica en el apartado 3.5.1 una forma de extraer esta información para restituirla tras la traducción.

2.2. Flujo de datos sin formato

El flujo de datos sin formato es producido como salida por el desformateador y por el generador, y es usado como entrada por el analizador

morfológico, por el postgenerador y por el reformateador.

En el subapartado de esta sección se especifica la forma de delimitar *superblancos* y *superblancos extensos*. Para los ejemplos, tomaremos como referencia el documento HTML de la figura 2.2.

```
<html>
  <head>
    <title>Título</title>
  </head>
  <body>
    <p>Frase
       dividida</p>
  </body>
</html>
```

Figura 2.2: Ejemplo de documento HTML

Los elementos estructurales que debe incluir este flujo de datos son los siguientes:

- *Superblancos*. Se trata de bloques que incluyen segmentos de información de formato incluida en los documentos, cuando estos son cortos.

- *Superblancos extensos*. Son marcas que se usan para especificar aquellos documentos externos que incluyan segmentos de información de formato del documento que se está procesando, cuando estos son largos.

- *Texto*. El texto de los documentos susceptible de ser traducido.

- *Finales de frase artificiales*. Cuando el formato de los documento induzca una separación de frases que no esté marcada por ningún signo de puntuación (por ejemplo, los títulos que no acaban en punto o el contenido de las celdillas de una tabla), el formato debe instrumentar un mecanismo (invisible para el usuario) que permita marcar estos finales de frase.

- *Protección de caracteres especiales (para el caso del flujo no XML)*. Los caracteres que se deben proteger para que no entren en conflicto con los usados en el propio formato del flujo de datos.

2.2.1. Formato de flujo

Este formato se basa en el formato utilizado en los sistemas de traducción interNOSTRUM [2, 5, 4] y Traductor Universia [7, 17].

En este tipo de flujo se usan los carácteres `[` y `]` para marcar los *superblancos* tal y como se muestra a continuación:

```
[contenido del superblanco]
```

Si se trata de *superblancos extensos*, el nombre del fichero se especifica utilizando el carácter de arroba `@`:

```
[@nombre de fichero]
```

El *texto* se encuentra fuera de las marcas de superblanco.

Los *finales de frase artificiales*se expresan mediante un punto y un superblanco vacío inmediatamente a continuación.

```
.[]
```

Los **caracteres protegidos** se muestran en la siguiente tabla:

Nombre	Carácter	Forma proteg.	Significado
Arroba	`@`	`\@`	Superblanco externo
Barra	`/`	`\/`	Divisor de acepciones
Circunflejo	`^`	`\^`	Inicio de FL
Corchete de apertura	`[`	`\[`	Inicio de blanco
Corchete de cierre	`]`	`\]`	Fin de blanco
Dólar	`$`	`\$`	Fin de FL
Mayor que	`>`	`\>`	Inicio de símbolo gram.
Menor que	`<`	`\<`	Fin de símbolo gram.

En la figura 2.3 muestra cómo quedaría encapsulado el documento de la figura 2.2.

2.3. Flujo de datos segmentado

El flujo de datos segmentado es el que circula entre los módulos que manejan información lingüística dentro del traductor. En este flujo las palabras se encuentran delimitadas y etiquetadas. Hay dos tipos de flujo segmentado:

```
[<html>
  <head>
    <title>]Título.[][</title>
  </head>
  <body>
    <p>]Frase[
       ]dividida.[][</p>
  </body>
<html>]
```

Figura 2.3: El documento de la figura 2.2 con el formato encapsulado usando corchetes

- *Flujo segmentado ambiguo*. Su característica principal es que las palabras tienen forma superficial y potencialmente varias formas léxicas (multiformas léxicas). Este flujo es el formato en el que el analizador morfológico proporciona los datos de entrada al desambiguador léxico categorial (consulte el esquema 3.2 de la página 51 para obtener una descripción detallada del flujo segmentado ambiguo).
- *Flujo segmentado no ambiguo*. Sólo tiene una forma léxica para cada palabra y no incluye la forma superficial. En este formato transitan los datos desde el desambiguador al módulo de transferencia y desde éste al generador (consulte el esquema 3.3 de la página 54 para ver el formato del flujo segmentado no ambiguo).

Por otro lado, además de la información que ya marca el flujo de datos sin formato, el nuevo flujo deberá permitir marcar la siguiente información:

- *Unidades léxicas*. Una unidad léxica consta de la forma superficial (para el caso del flujo segmentado ambiguo) más una o más formas léxicas (que corresponden a los posibles análisis de la FS) con sus símbolos gramaticales.
- *Formas superficiales (flujo segmentado ambiguo)*. Se trata de la palabra tal y como se encuentra en el texto original.
- *Formas léxicas*. El lema de la palabra con los símbolos gramaticales correspondientes.
- *Símbolos gramaticales*. Marcan los atributos gramaticales de la forma superficial correspondiente.

```
[<html>
  <head>
    <title>]^Título/Título<n><m><sg>$^./.<sent>$[][</title>
  </head>
  <body>
    <p>]^Frase/Frase<n><f><sg>$[
       ]^dividida/dividir<vblex><pp><f><sg>$^./.<sent>$[][</p>
  </body>
<html>]
```

Figura 2.4: Ejemplo de flujo segmentado con el formato encapsulado en formato no XML, correspondiente al documento HTML de la figura 2.2.

Para delimitar *palabras* se usan los caracteres de comienzo de palabra '^' y de fin de palabra '$' como se muestra a continuación:

```
^palabra$
```

Para delimitar la *forma superficial* y las sucesivas *formas léxicas* se usa el carácter de separación /. Este separador sólo tiene sentido en el flujo segmentado ambiguo ya que el no ambiguo sólo tiene la forma léxica. Su forma es la siguiente:

```
^forma superficial/forma léxica 1/...$
```

Las formas léxicas pueden incluir símbolos (generalmente situados al final), tal y como se muestra en el ejemplo de la figura 2.4.

Capítulo 3

Especificación de los módulos

3.1. Módulos de procesamiento léxico

3.1.1. Descripción del funcionamiento

Una de las aproximaciones más eficientes al procesamiento léxico usa transductores de estados finitos (TEF) [10, 15]. Los TEF son una clase de autómatas de estados finitos que se describen en el apartado siguiente. Los TEF pueden ser usados como analizadores morfológicos de una sola pasada y se pueden implementar de manera muy eficiente. En este proyecto se usa una clase de TEF denominados transductores de letras [15, 6, 4]; de hecho, cualquier transductor de estados finitos se puede convertir siempre en un transductor de letras. Garrido y colaboradores [4, 6] dan una una definición formal de los transductores de letras usados en este trabajo; informalmente, un transductor de letras es una máquina idealizada que se compone de:

1. Un conjunto (finito) de estados, es decir, de situaciones en las que se puede encontrar el transductor según va leyendo, de izquierda a derecha, las letras o símbolos de la entrada. Entre los estados del conjunto se distinguen:

   a) Un único estado inicial: el estado en el que se encuentra el transductor antes de procesar la primera letra o el primer símbolo de la entrada.

   b) Uno o más estados de aceptación, a los que sólo se llega después de haber leído completamente una entrada válida y, por tanto, sirven para detectar las palabras válidas.

2. Un conjunto (también finito) de transiciones de estado, cada una de las cuales se compone de:

   a) el estado de partida

   b) el estado de llegada

   c) la letra o el símbolo de entrada

   d) la letra o el símbolo de salida

   Para permitir que la salida y la entrada tengan longitud diferente en cualquier momento, se permite que el símbolo de entrada esté ausente, que el símbolo de salida esté ausente o que ambos estén ausentes. Esta circunstancia se indica normalmente usando un símbolo especial (el símbolo vacío).

El transductor construye, cada vez que lee un símbolo de la entrada, una lista de estados *vivos* o *activos*, cada uno de los cuales tiene una salida asociada (una secuencia de símbolos). El funcionamiento del transductor de letras es distinto para cada tipo de operación de procesamiento léxico. Por ejemplo, en el análisis morfológico se intenta leer la entrada más larga que reconoce el diccionario (modo "de izquierda a derecha, segmento concordante más largo").

1. Iniciación: se hace que el conjunto de estados vivos contenga un único estado vivo: el estado inicial, con la palabra vacía () como salida asociada al estado.

2. Si desde alguno de los estados del conjunto de estados vivos actual se llega a otros estados mediante transiciones que no tienen símbolo de entrada, se añaden estos estados al conjunto de estados vivos, y se les asocia la salida obtenida alargando las salidas asociadas con el símbolo de salida que se haya encontrado en las correspondientes transiciones. Esta operación de ampliación del conjunto de estados vivos se efectúa hasta que no sea posible añadir más.

3. Se lee un símbolo de la palabra de entrada.

4. Se crea un nuevo conjunto de estados vivos con aquellos a los que se llega mediante transiciones que tienen ese símbolo como entrada, y se asocia a estos estados las salidas alargadas con los correspondientes símbolos de salida encontrados en las transiciones.

5. Si el conjunto actual tiene algún estado vivo se va al paso 2.

6. Se recorren hacia atrás los conjuntos de estados vivos hasta que encontramos algún conjunto en el que aparecen estados de aceptación. Los análisis morfológicos serán las salidas asociadas a dichos estados, y la posición de lectura se fija a la posición inmediatamente posterior a la de ese conjunto (para que pueda volver a ser procesada por el transductor en la siguiente pasada).

No todos los estados de aceptación son de la misma clase, y esto añade algunas condiciones más al proceso de aceptación, para permitir el tratamiento de palabras desconocidas o de palabras que van unidas a otras, como se explicará más adelante.

Como puede verse, el transductor lee la palabra de entrada sólo una vez como caso promedio, de izquierda a derecha y símbolo a símbolo, y mantiene una lista tentativa de posibles salidas parciales que va actualizando y podando según lee la entrada. Cuando los transductores de letras se usan como analizadores morfológicos o como lematizadores, leen una forma superficial y escriben la(s) forma(s) léxica(s) correspondiente(s). En este caso, los símbolos de entrada son las letras de la forma superficial y los de salida las letras necesarias para formar los lemas y los símbolos especiales que representan el análisis morfológico, como, por ejemplo, `<n>`, `<f>`, `<2p>`, etc.

El funcionamiento de los transductores en otras tareas de procesamiento léxico es similar.

Tratamiento de mayúsculas y minúsculas en los diccionarios

Una misma palabra a la entrada de un módulo de procesamiento léxico puede presentar aspectos diferentes respecto al formato de mayúsculas y minúsculas. Los casos más frecuentes son los siguientes:

- Toda la palabra está en minúsculas.
- Toda la palabra está en mayúsculas.
- La primera letra de la palabra está en mayúsculas y el resto en minúsculas (caso más típico de los nombres propios).

Las transducciones en el diccionario también se pueden encontrar en los tres estados. La forma en la que un término se encuentre en el diccionario se usa para tratar de descartar acepciones de la palabra de la siguiente manera:

- Si la letra de la entrada es mayúscula y en el estado actual de análisis tenemos transiciones concordantes en minúsculas, estas transducciones se realizan.

- Si la letra de la entrada es minúscula y en el estado actual no tenemos transiciones concordantes en minúsuculas, no se realizan.

Esta política nos facilita que no figuren acepciones de nombres propios para una forma superficial que no empieza por mayúscula.

La condición de mayúscula o minúscula de una letra en una palabra será respetada a la salida del traductor a menos que se decida cambiar dicha condición. Esta opción está disponible en el módulo de transferencia estructural para casos como el de traducción de una palabra en origen por dos palabras en la lengua meta: `Canté` –¿`Vaig cantar`.

3.1.2. Formato de los datos: los diccionarios

Criterios generales para el diseño de los diccionarios

La experiencia del grupo Transducens de la Universitat d'Alacant en la construcción de sistemas de traducción automática ya operativos y públicamente accesibles entre lenguas románicas (`es`, `ca` y `pt`) ha inspirado los aspectos fundamentales del diseño completo del sistema de traducción automática de transferencia sintáctica superficial que se describe en este documento y de su aplicación a las lenguas románicas de España (`es`, `ca` y `gl`). En cierto sentido, se podría decir que en el presente proyecto sólo se ha tenido que adaptar (reescribir en un formato estandarizado e interoperable) las especificaciones y los programas ya usados en proyectos ya operativos.

En particular, el diseño de los diccionarios se ha seguido realizando a partir de una arquitectura que pretende independizar al máximo la lengua origen de la lengua meta, incluso sabiendo que se trata de diccionarios orientados a la traducción y que no conviene elaborarlos completamente por separado. El formato utilizado sirve para especificar tanto los diccionarios morfológicos (monolingües) como los diccionarios bilingües.

El formato de los diccionarios, al igual que el de los demás datos lingüísticos (fichero de definición del desambiguador y reglas de transferencia estructural) es XML[1], un estándar internacional utilizado en numerosos proyectos de tratamiento del lenguaje natural que, gracias a la existencia de numerosos útiles y librerías, se está convirtiendo en una herramienta

[1] `http://www.w3.org/XML/`

muy potente para la representación y el intercambio de datos lingüísticos (véase el artículo [8]).

Los diccionarios están diseñados de manera que puedan compilarse para elaborar *transductores de letras* con ellos, por cuestiones de eficiencia. Para más información sobre transductores de letras como caso particular de transductores de estados finitos véase el artículo [6].

Los transductores de letras que se elaboran a partir de los diccionarios (morfológicos, bilingües y de postgeneración) procesan cadenas de caracteres de entrada para ofrecer cadenas de salida. De este modo, los diccionarios están formados por entradas consistentes en parejas de cadenas que constituyen las entradas y las salidas de los transductores.

La herramienta más potente de estos diccionarios es la definición y el uso de *paradigmas*. Como en las lenguas romances muchos lemas comparten una misma flexión (existen regularidades en la flexión), resulta útil y directo agrupar estas regularidades en paradigmas de flexión para evitar tener que escribir todas las formas de cada palabra. Los paradigmas permiten representar las entradas del diccionario de manera compacta y optimizar la velocidad de construcción del diccionario. Una vez establecidos los paradigmas más frecuentes en un diccionario, un lingüista no tiene que preocuparse en la mayoría de ocasiones por toda la flexión de un nuevo término, puesto que introducir una palabra que se flexiona se reduce generalmente a indicar el lema e identificar la flexión entre los paradigmas previamente definidos. Además, el uso de paradigmas reduce los requisitos de memoria, favorece la construcción de transductores de letras eficientes y aumenta la velocidad de compilación [11]. En los diccionarios bilingües no se han utilizado paradigmas (aunque se podrían utilizar), puesto que la mayor parte de la información de flexión se procesa de manera implícita, como se explica en la página 37.

Tipos de diccionarios

En el sistema tenemos tres tipos de diccionarios: diccionarios morfológicos (monolingües) de cada uno de los idiomas implicados (español, catalán y gallego); diccionarios bilingües para los distintos pares de traducción (español–catalán y español–gallego), y diccionarios de postgeneración para cada uno de los idiomas (que más que un diccionario con lemas e información morfológica son un pequeño diccionario de las transformaciones ortográficas que pueden sufrir las palabras al entrar en contacto entre sí). La estructura de los tres tipos de diccionarios está especificada mediante la misma DTD (*definición de tipo de documento*), que puede consultarse en el apéndice A.1.

Los **diccionarios morfológicos** se usan tanto para construir analizadores morfológicos —el módulo del sistema de traducción que se usa para obtener todas las formas léxicas posibles para una forma superficial dada en la lengua origen— como generadores morfológicos —el módulo que se ocupa de generar la forma superficial en la lengua meta a partir de la forma léxica de cada palabra—. Estos dos módulos se obtienen a partir de un mismo diccionario morfológico, según el sentido en que el sistema lo lea: leído de izquierda a derecha se obtiene el analizador, y de derecha a izquierda, el generador.

La estructura de bloques típica de estos diccionarios consta de:

- *Una definición del alfabeto*. Esta definición se usa exclusivamente para la construcción del analizador morfológico; en concreto, para que éste pueda recortar las palabras desconocidas y las de las secciones condicionales (ver **??** de forma adecuada; el generador morfológico no necesita esta definición.

- *Una definición de los símbolos*. Consiste en una declaración de los símbolos gramaticales que se usarán en las entradas del diccionario (véase en el Apéndice B una lista con los símbolos gramaticales utilizados en este proyecto).

- *Una definición de paradigmas*. Para poder referirse a ellos en las secciones del diccionario o en otros paradigmas.

- *Una o más secciones de diccionario de recorte condicional*, de tipo `standard`. Para incluir la mayoría de las palabras del diccionario.

- *Una o más secciones de diccionario de recorte incondicional*. Para incluir ciertas palabras que siguen algún patrón regular o se recortan independientemente del texto que les sigue (ver discusión en el apartado **??**). En los diccionarios morfológicos de catalán, las palabras que requieren un recorte incondicional se reparten en dos secciones: una para las formas que necesitan que se coloque un blanco a continuación (por razones de procesamiento de las formas léxicas), como *l'* o *d'*, y otra para signos de puntuación, números y otros signos.

Los **diccionarios bilingües** son los que representan en el sistema el proceso de la transferencia léxica, esto es, la asignación de la forma léxica de la LM que corresponde a cada forma léxica de la LO. De cada diccionario bilingüe se obtienen dos *productos* según el sentido en el que el sistema los lea: leídos de izquierda a derecha se obtiene el módulo de transferencia

léxica en uno de los sentidos de traducción, y leídos de derecha a izquierda, en el otro sentido. Para los diccionarios bilingües de este proyecto se ha convenido que el español se situará siempre en la parte izquierda de las entradas, y las demás lenguas (catalán y gallego), en la parte derecha. Así, por ejemplo, el diccionario bilingüe español–gallego se leerá de izquierda a derecha para la traducción `es-gl` y de derecha a izquierda para la traducción `gl-es`. En aplicaciones como las de este proyecto, estos diccionarios no tienen paradigmas: se construyen mediante entradas genéricas en las que casi nunca se especifica más que su lema y categoría gramatical y no hay información de flexión.

El esquema de bloques usado en este proyecto para los diccionarios bilingües es el siguiente:

- *Una definición de los símbolos*. Consiste en una declaración de los símbolos gramaticales que se usarán en las entradas del diccionario.

- *Una sección única de diccionario*. Para incluir las correspondencias bilingües del diccionario.

Los **diccionarios de postgeneración** se usan para llevar a cabo las operaciones de transformación ortográfica, contracción, apostrofación, etc. que sean necesarias una vez generadas las formas superficiales de la lengua meta debido al contacto entre palabras. Como este tipo de operaciones se pueden expresar como traducciones de cadenas de caracteres se ha optado por usar el mismo tipo de diccionarios. Implícitamente, se entiende que las partes del texto cuyo procesamiento no se especifica se copian tal como llegan. En estos diccionarios es útil la definición de paradigmas para expresar cambios sistemáticos en los fenómenos de contacto entre palabras. A diferencia de los demás tipos de diccionarios, estos no tienen símbolos gramaticales, ya que procesan formas superficiales.

El esquema de bloques de los diccionarios de postgeneración es el siguiente:

- *Una sección de definición de paradigmas*. Para componer las entradas.

- *Una sección de diccionario*. Para incluir los patrones para realizar la postgeneración.

En la siguiente tabla se presentan de forma resumida los sentidos de lectura de los diccionarios y su utilidad para las lenguas románicas del presente proyecto:

```
<?xml version="1.0" encoding="iso-8859-15"?>
<dictionary>
  <alphabet>abcdefghijk ... ABCDEFGH ... çñáéíóú</alphabet>
  <sdefs>
    <!-- ... -->
  </sdefs>
  <pardefs>
    <!-- ... -->
  </pardefs>
  <section ...>
    <!-- ... -->
  </section>
  <!-- ... -->
</dictionary>
```

Figura 3.1: Uso de los elementos <**dictionary**> y <**alphabet**>

Diccionario	Sentido de lectura	Función
Morfológico	izquierda–derecha	análisis de `es`, `ca` y `gl`
	derecha–izquierda	generación de `es`, `ca` y `gl`
Bilingüe	izquierda–derecha	traducción `es-ca` y `es-gl`
	derecha–izquierda	traducción `ca-es` y `gl-es`
Postgeneración	izquierda–derecha	postgeneración para `ca`, `es` y `gl`

Descripción del formato de los diccionarios

En esta sección se presentan los principales elementos del formato en el que se construyen los diccionarios. La definición formal (una DTD) se da en el apéndice A.1. En las páginas 37, 37 y 40 se describen las características particulares de las entradas de los tres tipos de diccionarios (morfológicos, bilingües y de postgeneración respectivamente).

Elemento diccionario <dictionary>

Es el elemento raíz, que incluye todo el diccionario. Incluye una definición de caracteres alfabéticos, una definición de símbolos (que son las etiquetas morfológicas de las palabras), una definición de paradigmas de flexión y una o varias secciones de diccionario, en las que se encuentran las entradas correspondientes a las formas léxicas (formadas por pares de forma superficial–forma léxica). En la figura 3.1 se muestra la estructura básica de bloques de un diccionario genérico.

```
<sdefs>
  <sdef n="n"/>
  <sdef n="det"/>
  <sdef n="sg"/>
  <sdef n="pl"/>
  <!-- ... -->
</sdefs>
```

Figura 3.2: Uso del elemento <**sdefs**>

Elemento alfabeto <alphabet>

Permite especificar una definición de caracteres alfabéticos. El objeto de esta especificación es que los módulos que procesan la entrada mediante transductores de letras puedan recortarla en palabras individuales.

En la actual especificación, sólo tiene sentido la definición de este elemento en los diccionarios morfológicos, por ser necesario para el análisis. La figura 3.1 muestra un ejemplo de uso de este elemento.

Elemento sección de definición de símbolos <sdefs>

Agrupa todas las definiciones de símbolos de un diccionario (<**sdef**>). Un ejemplo se muestra en la figura 3.2.

Elemento definición de símbolos <sdef>

Se trata de un elemento vacío (no delimita ningún contenido): sirve para especificar los nombres de los símbolos gramaticales que se usan en este diccionario mediante los valores del atributo *n*. Estos símbolos son los que se utilizarán para etiquetar morfológicamente las formas léxicas. En la figura 3.2 se ilustra un ejemplo del ámbito en el que se usa este elemento. En el Apéndice B puede verse una lista de todos los simbolos gramaticales utilizados en los diccionarios de este proyecto.

Elemento de sección del diccionario <section>

Define en su interior las palabras que serán reconocidas por el diccionario. La razón de dividir el diccionario en secciones es porque ciertos grupos de formas —como por ejemplo, las que proceden de la identificación de ciertos patrones regulares, o las que pudieran depender de un dialecto específico— necesitan en ocasiones un procesamiento diferente.

```
<section id="principal" type="standard">
<!-- ... -->
</section>
<section id="patterns" type="inconditional">
<!-- ... -->
</section>
```

Figura 3.3: Uso del elemento <**section**>

Uno de los procesamientos específicos que se resuelven mediante la definición de secciones en el diccionario es el problema del procedimiento de *recorte* (división de palabras) en el análisis morfológico. La mayor parte de formas se recortan mediante un criterio condicional, atendiendo a si viene un carácter no alfabético —es decir, no definido en <**alphabet**>— después del carácter que está siendo procesado. Pero hay otras formas, como las catalanas *l'* o *d'*, que requieren un modo de recorte incondicional, sin esperar a leer lo que viene después —ya que si viene un carácter alfabético después, forma parte de la *siguiente* palabra—. Estas formas que requieren de un recorte incondicional se incluyen en una sección específica del diccionario. Otros tipos de procesamiento también pueden resolverse mediante estas divisiones.

Se usa el valor del atributo `type` para expresar el tipo de reconocimiento de las cadenas que se usa en cada sección del diccionario: los valores del atributo son `standard` para casi todos los propósitos del diccionario (condicional), `postblank` para las formas que requieren un recorte incondicional y la colocación de un blanco, e `inconditional` para las demás formas de recorte incondicional.

El atributo `id` se usa para otorgar un identificador (un nombre) a las secciones del diccionario.

Elementos de las entradas <e>

La entrada es la unidad básica del diccionario o de la definición de paradigma. Las entradas consisten en una concatenación en cualquier orden de parejas de cadenas <**p**>, transducciones identidad <**i**>, referencias a paradigmas <**par**> o expresiones regulares <**re**>. La estructura y el significado de estos elementos se explica más adelante (en las páginas 28, 29, 32 y 33 respectivamente).

Se usan dos atributos opcionales con esta entrada. Uno es `r` (de *restriction*) que define si esa entrada se tiene que tener en cuenta durante la

lectura del diccionario sólo de izquierda a derecha (`LR`) o sólo de derecha a izquierda (`RL`). Si no se especifica nada se supone que la entrada se debe tener en cuenta en ambas direcciones.

En los diccionarios morfológicos, la restricción de dirección `LR` se utiliza para que una FL sea analizada pero no generada (por ejemplo, por tratarse de una FL en una variante dialectal que se desea reconocer pero no generar) y la restricción `RL` para que una palabra sólo se genere pero no se analice (necesario, por ejemplo, para las formas con marca de activación del postgenerador, véase la página 35).

En los diccionarios bilingües, las restricciones `LR` y `RL` sirven para que la traducción se haga sólo en un sentido: por ejemplo, en un diccionario bilingüe `es-ca`, `LR` indicaría que la FL sólo se traduce de español a catalán, y `RL` sólo de catalán a español. Lo ilustraremos con un ejemplo: los adverbios españoles *aún* y *todavía* los traducimos al catalán por una misma palabra, *encara*. El adverbio catalán *encara* sólo podemos traducirlo al español por uno de las dos, y decidimos traducirlo por *todavía*. En este caso, debemos escribir dos entradas en el diccionario bilingüe: la que empareja *aún* con *encara* debe tener la restricción `LR` (traducción sólo de `es` a `ca`) y la que empareja *todavía* con *encara* no debe tener ninguna restricción (traducción en ambos sentidos).

Las restricciones de dirección en diccionarios bilingües son necesarias también en el caso de palabras con género por determinar ("GD") o número por determinar ("ND") (véase la página 37).

El otro atributo opcional de las entradas es el nombre del lema *lm*. Debido a la utilización de paradigmas para representar las regularidades en la flexión de las unidades léxicas, las entradas de los diccionarios morfológicos contienen la parte del lema que es común a todas las formas flexionadas, es decir, contienen el lema cortado en el punto en que empieza la regularidad del paradigma (por ejemplo, los adjetivos españoles *distinto*, *absoluto* y *marino* aparecen en las entradas como *distint*, *absolut* y *marin*, siendo el resto de sus formas flexionadas común a todos ellos y descrito en un paradigma). Este hecho puede provocar dificultades en la comprensión del diccionario. Por ello, en las entradas se ha añadido este atributo, que contiene el lema completo de la forma léxica, de modo que el diccionario gane en claridad y los lingüistas puedan solucionar problemas rápidamente. En los diccionarios bilingües, que no suelen tener referencias a paradigmas,[2] no suele ser necesario usarlo.

[2]Podrían tener referencias a paradigmas, pero no lo hemos considerado necesario para las lenguas involucradas.

```
<p>
  <l><!-- ... --></l>
  <r><!-- ... --></r>
</p>
```

Figura 3.4: Uso del elemento <**p**>

Elemento pareja de cadenas <p>

Este elemento básico de los diccionarios se usa en cualquier tipo de entrada para indicar la correspondencia entre dos cadenas de símbolos, correspondencia que especifica una transformación léxica que será realizada por un camino de estados en el transductor de estados finitos [4] resultante.

Están definidas por un par de elementos internos. Uno es el izquierdo (elemento <**l**>, *left*) y el otro el derecho (elemento <**r**>, *right*). La estructura de esta etiqueta presenta el aspecto que se muestra en la figura 3.4.

Una pareja <**p**> debe incluir estas dos partes aunque una de ellas puede estar vacía, lo que se corresponde con borrar (o insertar) una cadena. Los elementos <**l**> y <**r**> tienen la misma estructura interna y los mismos requisitos. Dentro de cada pareja puede haber texto y referencias a símbolos gramaticales (que en las lenguas del presente proyecto, con flexión sufijal, se suelen poner al final en cualquier cantidad). En el exterior de las etiquetas <**l**> y <**r**> de las parejas de cadenas no hay nada.

Elemento referencia a símbolo (o etiqueta) <s>

Las referencias a símbolos sirven para especificar la información morfológica de una FL y se usan en cualquier punto dentro de una pareja de cadenas, es decir, dentro de <**l**>, <**r**> o de ambos, como si se tratasen de caracteres individuales, aunque en las lenguas del presente proyecto se suelen poner al final de las parejas y siempre en el mismo orden para el mismo tipo de palabra. Este orden lo decide el lingüista según como quiera caracterizar morfológicamente las FL de los diccionarios, y debe ser el mismo en todos los diccionarios del sistema para que las operaciones de transferencia léxica y estructural funcionen correctamente. Así, por ejemplo, en los diccionarios de este proyecto, un nombre común lleva primero el símbolo de categoría (*n*, nombre), luego el de género (*m*, masculino, *f*, femenino, *mf*, masculino–femenino), y finalmente el de número (*sg*, singular, *pl*, plural, *sp*, singular–plural). En el Apéndice B se ofrece una lista con los símbolos gramaticales utilizados en los diccionarios de este proyecto y

```
[1]

<e lm="perro">
  <p>
    <l>perr</l><r>perr</r>
  </p>
  <par n="abuel_o__n"/>
</e>

[2]

<e lm="perro">
  <i>perr</i>
  <par n="abuel_o__n"/>
</e>
```

Figura 3.5: Uso del elemento <**i**>: las entradas [1] y [2] son equivalentes

el orden que se ha establecido para definir los distintos tipos de palabra.

En los diccionarios morfológicos las referencias a símbolos se usan en los paradigmas de flexión y en las entradas que no incluyen ninguna referencia a paradigma. En los diccionarios bilingües suelen indicarse únicamente los primeros símbolos gramaticales (normalmente sólo el primero) de cada FL, ya que el resto se copia automáticamente de la FL en lengua origen a la FL en lengua meta (si coincide en ambas lenguas).

Para especificar el símbolo al que nos referimos se usa el atributo (obligatorio) *n*. Se requiere que el símbolo esté definido en la sección de definición de símbolos (<**sdefs**>).

Elemento transducción identidad <i>

Debe ser visto como una forma de escribir una pareja de cadenas en la que la parte izquierda y la parte derecha son idénticas. Por ejemplo, a todos los efectos las dos entradas que se muestran en la figuras 3.5 son equivalentes. La ventaja de escribir las entradas en esta notación es que resultan más compactas y también más legibles.

Elemento sección de definición de paradigmas <pardefs>

Este elemento agrupa todas las definiciones de paradigmas del diccionario, contenidas cada una en un elemento <**pardef**>, como se muestra

```
<pardefs>
  <pardef n="abuel_o__n">
    <!-- ... -->
  </pardef>
  <!-- ... -->
</pardefs>
```

Figura 3.6: Uso del elemento <**pardefs**>

en la figura 3.6.

Elemento de definición de paradigmas <pardef>

Define un paradigma de flexión del diccionario. Un paradigma se puede entender como un pequeño diccionario de transformaciones alternativas que se pueden concatenar a partes de las palabras (o a entradas de otro paradigma) para especificar regularidades en el procesamiento léxico de las entradas del diccionario, como, por ejemplo, regularidades en la flexión. Para especificar estas regularidades, cada paradigma es una lista de entradas <**e**> como las del diccionario, es decir, tiene la misma estructura que una sección del diccionario <**section**>; por tanto, las entradas consisten en una pareja (<**p**>) con parte izquierda (<**l**>) y parte derecha (<**r**>). Dentro de estos elementos se puede incluir texto o símbolos morfológicos <**s**>.

Como en el caso de las definiciones de símbolos, las definiciones de paradigmas tienen un atributo *n* que especifica el nombre del paradigma para poder utilizarlo posteriormente dentro de las entradas del diccionario. Así, en cada entrada del diccionario bastará con indicar el nombre del paradigma que le corresponde para que queden especificadas todas sus posibles formas.

El ejemplo de definición de paradigma que se apunta en la figura 3.6 se desarrolla en la figura 3.7. Para ilustrar qué información se expresa mediante el paradigma se presenta la siguiente tabla:

Raíz (FS y FL)	Desinencia (FS)	Análisis (FL)
abuel	o	o<n><m><sg>
abuel	a	o<n><f><sg>
abuel	os	o<n><m><pl>
abuel	as	o<n><f><pl>

```
<pardef n="abuel_o__n">
  <e>
    <p>
      <l>o</l>
      <r>o<s n="n"/><s n="m"/><s n="sg"/></r>
    </p>
  </e>
  <e>
    <p>
      <l>a</l>
      <r>o<s n="n"/><s n="f"/><s n="sg"/></r>
    </p>
  </e>
  <e>
    <p>
      <l>os</l>
      <r>o<s n="n"/><s n="m"/><s n="pl"/></r>
    </p>
  </e>
  <e>
    <p>
      <l>as</l>
      <r>o<s n="n"/><s n="f"/><s n="pl"/></r>
    </p>
  </e>
</pardef>
```

Figura 3.7: Uso del elemento <**pardef**> para definir la morfología flexiva de substantivos de cuatro terminaciones como *abuelo, -a, -os, -as*.

```
<pardef n="ex">
  <e r="LR"><p><l>ex<b/></l><r>ex</r></p></e>
  <e><i>ex</i></e>
  <e r="LR"><p><l>ex-</l><r>ex</r></p></e>
  <e><i/></e>
</pardef>
```

Figura 3.8: Uso del elemento <**pardef**> en el paradigma usado para el prefijo *ex*.

Este paradigma se asigna a todos los sustantivos (n) que se flexionan igual que *abuelo*, como *alumno*, *amigo* o *gato*, y está pensado para ser usado como *sufijo* en las entradas del diccionario. En general se pueden usar paradigmas en cualquier punto de las entradas del diccionario (siempre que tenga sentido, claro está). Podemos pensar en los paradigmas como transductores que se insertan en el punto en el que se haga referencia a ellos. En la figura 3.8 se muestra un ejemplo de un paradigma definido para ser usado como prefijo. Es el caso del paradigma que se usa para analizar y generar las palabras que empiezan por *ex*, *ex-*, etc., como *ex-presidente*, *exministro*, *ex director*, etc., con sus variaciones ortográficas (*ex* con guión, sin guión unido y sin guión separado por un blanco <**b**/>, véase la pàgina 3.1.2); el lema entregado simplemente añade *ex* sin guión ni blanco al lema del resto. Las restricciones de dirección ("LR") que aparecen en el ejemplo se usan para decidir qué forma es la que va a generar el traductor. La transducción identidad vacía (<**i**/>) es necesaria en este caso para analizar y generar la palabra sin el prefijo *ex*.

Las entradas de un paradigma pueden incluir referencias a otros paradigmas con la condición de que hayan sido definidos previamente en el fichero. Por otro lado, por el momento, una definición de paradigma no puede incluirse a sí mismo ni directa ni indirectamente.

Los paradigmas se usan en los diccionarios morfológicos para el análisis y la generación de formas léxicas. Para los pares de lenguas de este proyecto, no es necesaria la definición de paradigmas en los diccionarios bilingües (página 37).

Elemento referencia a paradigma de flexión <par>

Se utiliza dentro de las entradas para indicar qué paradigma, de los que se han definido en <**pardefs**>, sigue la entrada en cuestión. Gracias a las referencias a paradigmas no es necesario escribir en cada entrada de un diccionario morfológico todas las formas flexionadas del lema. El atributo

```
<e lm="perro">
  <i>perr</i>
  <par n="abuel_o__n"/>
</e>
```

Figura 3.9: Uso del elemento <**par**>

n sirve para especificar a qué paradigma se hace referencia.

El resultado de insertar una referencia a paradigma en una entrada es la creación de tantas parejas de cadenas como casos especifique el paradigma. Por ejemplo, la entrada de la figura 3.9, con la referencia al paradigma "abuel_o__n"(que está definido en la figura 3.7), equivale a realizar una entrada con cada pareja de cadenas del paradigma concatenada al lema (es decir, con cada forma flexionada del lema), como se representa en la figura 3.10. En ella puede verse como el paradigma ofrece siempre como cadena derecha (<**r**>) el lema (*perro*) junto con los símbolos gramaticales correspondientes a la forma superficial, ya que es a partir del lema que se realizan las operaciones de transferencia.

El uso adecuado de paradigmas, además de permitir la creación de diccionarios compactos, favorece la velocidad de compilación y rebaja la necesidad de memoria durante la misma, ya que durante el proceso de compilación es posible crear una única estructura de datos para cada uno de la mayor parte de paradigmas [11].

Elemento expresión regular <re>

En los lenguajes naturales también hay patrones que se pueden reconocer como expresiones regulares: por ejemplo, los signos de puntuación, los números (latinos o romanos), las direcciones de correo electrónico o de páginas de internet o cualquier tipo de código identificable mediante estos mecanismos.

Para estos usos se utiliza la cadena que contiene la etiqueta <**re**>. El compilador lee la definición de la expresión regular y la transforma en un transductor que se integra en el resto del diccionario y que traduce todas las cadenas que concuerdan con la expresión por cadenas idénticas.

La sintaxis de la implementación actual de estas expresiones regulares procesa un subconjunto de las expresiones regulares de Unix, que incluye los operadores `*`, `?`, `|` y `+`, además de agrupaciones mediante paréntesis y rangos de caracteres opcionales como por ejemplo `[a-zA-zñú]` o sus versiones negadas, por ejemplo `[^a-z]`.

```
<e>
  <p>
    <l>perro</l>
    <r>perro<s n="n"/><s n="m"/><s n="sg"/></r>
  </p>
</e>
<e>
  <p>
    <l>perra</l>
    <r>perro<s n="n"/><s n="f"/><s n="sg"/></r>
  </p>
</e>
<e>
  <p>
    <l>perros</l>
    <r>perro<s n="n"/><s n="m"/><s n="pl"/></r>
  </p>
</e>
<e>
  <p>
    <l>perras</l>
    <r>perro<s n="n"/><s n="f"/><s n="pl"/></r>
  </p>
</e>
```

Figura 3.10: Entrada equivalente a la de la figura 3.9 que muestra el resultado de insertar la referencia a paradigma <**par**> con el paradigma definido en la figura 3.7.

```
<e>
  <re>[0-9]+([.,][0-9]+)?(%)?</re>
  <p><l/><r><s n="num"/></r></p>
</e>
```

Figura 3.11: Uso del elemento <**re**> en una entrada que sirve para la identificación de números arábigos.

```
<e lm="hoy en día">
  <p>
    <l>hoy<b/>en<b/>día</l>
    <r>hoy<b/>en<b/>día<s n="adv"/></r>
  </p>
</e>
```

Figura 3.12: Uso del elemento <**b**>

Por analogía, se pueden ver como si se tratase de elementos <**i**>, pero con la característica de que pueden identificar cadenas que en algún caso pueden ser infinitas (como los números).

En la figura 3.11 se muestra cómo etiquetar cantidades expresadas como números arábigos en el diccionario.

Elemento de bloque de blancos <b>

Se usa para expresar la presencia de blancos entre palabras de una unidad multipalabra (vea la explicación de las multipalabras en la página 42). Se puede usar dentro de los elementos <**i**>, <**l**> y <**r**>. En la figura 3.12 se puede ver la entrada correspondiente a la expresión *hoy en día*: los espacios existentes entre las palabras aparecen como elementos <**b**/> dentro de las cadenas izquierda y derecha.

Los blancos pueden ser caracteres de espaciado normales o bloques de información sobre el formato del documento encapsulados por el desformatador (*superblancos*, ver sección 3.5.1).

Elemento de activación del postgenerador <a>

El elemento de activación del postgenerador <**a**> se usa para indicar que algunas palabras en la lengua meta son susceptibles de sufrir transformaciones ortográficas por contacto con otras palabras; por ejemplo, ser apostrofadas, contraídas, escritas sin espacios intermedios, etc. Estas

```
<e r="RL" lm="de">
   <p>
      <l><a/>de</l>
      <r>de<s n="pr"/></r>
   </p>
</e>
```

Figura 3.13: Uso del elemento <**a**> en un diccionario morfológico

transformaciones deben realizarse tras la generación de las formas superficiales en la lengua meta, ya que antes, con las palabras aisladas, no es posible saber qué palabras entrarán en contacto. Así, es el módulo siguiente al generador (llamado postgenerador) el que se encarga de tales operaciones. Para indicar qué palabras debe procesar el postgenerador, se usa este elemento en la parte de la forma superficial de las entradas correspondientes de los diccionarios morfológicos.

El ejemplo de la figura 3.13 muestra su uso en un diccionario morfológico de catalán para la preposición *de*, la cual, si aparece delante del artículo definido masculino singular o plural (*el, els*) forma una contracción (*del, dels*). La etiqueta <**a**/> incluye información que provoca la activación del postgenerador, el cual comprueba si la preposición viene seguida de las palabras que provocan su contracción y, de ser así, la realiza (véase la página 40 para más detalles). La restricción `RL` indica que esta entrada es sólo de generación, ya que no tiene ningún sentido para el análisis.

Elemento de marcado de grupos <g>

Este elemento se usa para definir, dentro de los elementos <**l**> y <**r**>, grupos que requieren un tratamiento especial más allá del tratamiento palabra por palabra. Se utiliza en las multipalabras con flexión para señalar el principio y el final de la forma o formas léxicas que van unidas a la palabra flexionada y que, junto a ella, forman una unidad inseparable. En la página 42 se explican con más detalle los distintos tipos de multipalabra, y en la figura 3.22 del mismo puede verse un ejemplo de utilización de este elemento.

Elemento de unión de formas léxicas <j>

Este elemento se utiliza sólo en la parte izquierda de las entradas (<**r**>) para indicar que las palabras que forman parte de una multipalabra son

tratadas como formas léxicas individuales y, por lo tanto, tienen cada una un símbolo morfológico. Estas multipalabras serán tratadas como una unidad por el analizador y por el desambiguador hasta llegar al módulo auxiliar `pretransfer` (véase el apartado 3.3), que se encarga de separar las formas léxicas que las componen para que lleguen al módulo de transferencia como formas independientes. Si se desea que estas formas lleguen al generador como formas unidas, formando una multipalabra otra vez, debe haber una regla de transferencia estructural que realice la acción de agruparlos (véase el apartado 3.4.2). Si, por el contrario, estas formas deben ser sólo de análisis, la entrada debe tener la restricción `LR`.

En la página 42 se ilustra más detalladamente el uso de este elemento. Puede verse un ejemplo de uso en la figura 3.20 de la mencionada sección.

Particularidades de los distintos tipos de diccionarios

Las entradas de los diccionarios tienen algunas características diferenciadas según el diccionario de que se trate. Aunque en la especificación precedente han quedado apuntadas algunas de estas particularidades, a continuación las presentamos de manera más exhaustiva.

Diccionarios morfológicos

En estos diccionarios, que se usan para construir los analizadores y los generadores morfológicos, se deben marcar mediante <**a**/> las formas superficiales que una vez generadas pueden necesitar ciertas transformaciones ortográficas por contacto con otras palabras, operaciones de la que se encarga el postgenerador. Como se trata de marcas que sólo se generan, las entradas que contengan estas marcas deben ser sólo de generación, es decir, con la restricción *r*="RL" (de derecha a izquierda). En la figura 3.13 puede verse un ejemplo de entrada con este elemento.

Diccionarios bilingües

Como ya hemos explicado, en nuestro sistema no se utilizan paradigmas en los diccionarios bilingües; éstos se construyen mediante entradas genéricas en las que casi nunca se especifica más que su categoría gramatical y no hay información de flexión. Por ejemplo, en el diccionario `es-ca` la entrada para la palabra española *pan*, *panes*, traducida al catalán por *pa*, *pans*, quedaría como muestra la figura 3.14.

Como vemos, en la entrada únicamente se especifica el primero símbolo gramatical <**s** *n*="..."/> de ambas palabras, ya que los símbolos no

```
<e>
  <p>
    <l>pan<s n="n"/></l>
    <r>pa<s n="n"/></r>
  </p>
</e>
```

Figura 3.14: Entrada en el diccionario bilingüe para la traducción *pan* (es)–*pa* (ca)

especificados que siguen a los especificados en el diccionario bilingüe son copiados de la forma léxica original a la forma léxica de la lengua meta; así, esta entrada vale tanto para *pan* (singular) como para *panes* (plural), ya que el módulo analizador entrega el lema (*pan*) seguido de los símbolos gramaticales correspondientes a la forma superficial analizada (*n m sg* o *n m pl* según el caso); por lo tanto, los símbolos no especificados en la entrada (*m sg* o *m pl*) se copian a la lengua meta. Lo mismo vale para los dos sentidos de traducción. La idea es que se especifica la información imprescindible para diferenciar las entradas y el resto de información se *sobreentiende* (se copia). Es importante tener en cuenta este hecho ya que, si existen diferencias entre los símbolos gramaticales de una forma léxica entre la LO y la LM, estas diferencias deben quedar especificadas en el diccionario bilingüe. Así, por ejemplo, cuando hay un cambio de género o de número entre la forma original y la forma traducida, hay que especificar los símbolos gramaticales por orden (el orden en que aparecen estos símbolos en los diccionarios morfológicos)[3] hasta que lleguemos al símbolo divergente entre LO y LM.

Por ejemplo, para traducir la palabra española *cama*, nombre femenino, por la palabra catalana *llit*, nombre masculino, la entrada en el diccionario bilingüe debe ser como se muestra en la figura 3.15. Debe especificarse el género (*f*, *m*) ya que, de no hacerlo, se copiarían los símbolos de género y número de la forma léxica en LO a la forma léxica en LM. Por lo tanto, al traducir de `es` a `ca` se obtendría la forma léxica *llit* con los símbolos `n f sg` o bien `n f pl`. En ambos casos, el generador se encontraría con una palabra imposible de generar, ya que los diccionarios morfológicos del catalán no contienen ninguna entrada con lema *llit* y género femenino.

En este ejemplo el número no se especifica y la entrada sirve tanto para la correspondencia *cama–llit* como para la correspondencia *camas–llits*. Sin embargo, si hay que especificar un cambio de número, no hay más re-

[3]Para saber qué símbolos gramaticales se han utilizado en los diccionarios morfológicos y en qué orden, consúltese el Apéndice B.

```
<e>
  <p>
    <l>cama<s n="n"/><s n="f"/></l>
    <r>llit<s n="n"/><s n="m"/></r>
  </p>
</e>
```

Figura 3.15: Entrada en el diccionario bilingüe para la traducción *cama* (es)–*llit* (ca)

medio que especificar también el género si en todo el diccionario el orden usado es *género, número*.

Mediante la restricción de dirección r podemos indicar qué traducciones tienen lugar en un sentido y no en el otro (véase la descripción de las restricciones LR y RL en la página 26). Esto es necesario cuando la correspondencia entre dos formas léxicas no es simétrica, en cuyo caso deben realizarse varias entradas en el diccionario bilingüe con alguna restricción de dirección, como en el ejemplo de la figura 3.16. En este ejemplo, cuando traducimos de español a catalán (LR), necesitamos que solo se generen formas en plural porque la palabra *postres* en catalán no tiene singular. Pero en cambio, sólo traduciremos al español en plural porque no hay forma de determinar si el número sería singular o plural.

```
<e r="LR">
  <p>
    <l>postre<s n="n"/><s n="m"/><s n="sg"/></l>
    <r>postres<s n="n"/><s n="m"/><s n="pl"/></r>
  </p>
</e>

<e>
  <p>
    <l>postre<s n="n"/><s n="m"/><s n="pl"/></l>
    <r>postres<s n="n"/><s n="m"/><s n="pl"/></r>
  </p>
</e>
```

Figura 3.16: Entradas en el diccionario bilingüe español–catalán/valenciano para la correspondencia *postre–postres*

Existe otro problema debido a las diferencias gramaticales entre dos lenguas determinadas que se soluciona con dos símbolos especiales, GD

(de *género por determinar*) y `ND` (de *número por determinar*), símbolos que deberán estar definidos en la sección de símbolos del diccionario bilingüe. Este problema surge cuando la información gramatical de una forma léxica en LO no es suficiente para determinar el género (masculino o femenino) o el número (singular o plural) de la forma léxica en LM. Por poner un ejemplo, el adjetivo *común* en español es tanto masculino como femenino (y por tanto masculino/femenino, `mf`), pero en catalán hay formas diferentes para el masculino, *comú/comuns*, y para el femenino, *comuna/comunes*. En el diccionario bilingüe la entrada quedaría como se ve en la figura 3.17: en la dirección `LR` (de español a catalán), la información de género no es `m`, `f` ni `mf` sino `GD`; este *género por determinar* lo determinará a continuación el módulo de transferencia estructural mediante la aplicación de las reglas de transferencia apropiadas (normalmente reglas de concordancia entre formas léxicas de un patrón; véase el apartado 3.4 para obtener una descripción detallada del funcionamiento de las reglas de este módulo). Análogamente, se puede definir un mecanismo similar, mediante el símbolo `ND`, para el caso del número singular–plural (por ejemplo, en español *análisis* es singular y plural, mientras que en catalán el singular es *anàlisi* y el plural *anàlisis*).

Diccionarios de postgeneración

En el diccionario morfológico, las formas léxicas que, una vez generadas, pueden sufrir operaciones de contracción, apostrofación, etc., según las palabras que entren en contacto con ellas en el texto resultante, deben llevar la marca de activación del postgenerador (<**a**/>, véase la página 35) en la entrada de generación (dirección `LR`). Es fundamental que las formas superficiales marcadas con la marca de activación del postgenerador coincidan en los diccionarios morfológicos y de postgeneración del mismo traductor. En el diccionario de postgeneración, todas las entradas comienzan por esta marca de activación.

En la figura 3.18 se muestra un ejemplo de fragmento del postgenerador para el español; en este ejemplo se puede ver cómo se realiza la contracción de *de* y *el* para formar *del*. En el ejemplo, el paradigma no definido, `puntuación`, es un paradigma que se limita a definir los caracteres no alfabéticos que pueden formar parte de un texto. Vemos en el ejemplo que la entrada para la preposición *de* lleva la marca <**a**/>. El paradigma asignado a esta entrada, "`el`", es el que encontramos definido arriba. De este modo, cuando el sistema se encuentra con la cadena izquierda de la entrada (la parte entre <**l**>) concatenada con la cadena izquierda del paradigma (es decir, cuando se encuentra con las cadenas de entra-

```
<e r="LR">
  <p>
    <l>común<s n="adj"/><s n="mf"/></l>
    <r>comú<s n="adj"/><s n="GD"/></r>
  </p>
</e>

<e r="RL">
  <p>
    <l>común<s n="adj"/><s n="mf"/></l>
    <r>comú<s n="adj"/><s n="m"/></r>
  </p>
</e>

<e r="RL">
  <p>
    <l>común<s n="adj"/><s n="mf"/></l>
    <r>comú<s n="adj"/><s n="f"/></r>
  </p>
</e>
```

Figura 3.17: Entradas en el diccionario bilingüe español–catalán para la correspondencia *común–comú*, la primera para la traducción del español al catalán y las dos últimas para la traducción del catalán al español

da "<a/>**de**<b/>**el**<b/>" o bien "<a/>**de**<b/>**el**[puntuación]"), el módulo ofrece como cadena de salida (la parte entre <r>) la cadena **"del"** seguida de los blancos representados por <b/> o de los símbolos representados por [puntuación]. Nótese que a la salida del módulo todas las marcas <**a**/> han sido eliminadas.

Unidades léxicas multipalabra

Con el formato propuesto para los diccionarios es posible introducir *unidades léxicas multipalabra* —simplificando, *multipalabras*— de formas diferentes dependiendo del problema que haya que resolver.

En este proyecto se consideran tres tipos básicos de unidades multipalabra:

1. El caso más sencillo es el de las *multipalabras sin flexión*, que están formadas por una sola forma léxica, ya que su lema consta de varias palabras ortográficas invariables pero se etiqueta como una unidad. En la figura 3.19 se muestra un ejemplo de unidad multipalabra invariable (la expresión *hoy en día*). Esta multipalabra consta de tres palabras separadas por un blanco (<**b**/>) y, aunque estrictamente esté formada por un adverbio, una preposición y un nombre, se etiqueta globalmente como adverbio, pues cumple una función equivalente.

2. Un caso algo más complejo es el de las *multipalabras compuestas*, que están formadas por más de una forma léxica, cada una con sus símbolos gramaticales. Se considera que las palabras de las que se componen no conforman una unidad semántica como en el caso anterior, y que sólo aparecen juntas formando una unidad por razones de contacto entre sí (razones fonéticas u ortográficas). En esta categoría entran las *contracciones* y los *pronombres enclíticos* que acompañanan a los verbos. Para marcar este fenómeno se usa la etiqueta <**j**> descrita en la página 36. Puede verse un ejemplo en la figura 3.20, en la que el análisis de *del* entrega una multiforma léxica compuesta por dos formas léxicas: *de*, preposición, y *el*, determinante definido masculino singular, enlazadas mediante el elemento <**j**/>. El analizador y el desambiguador léxico categorial tratan estas multipalabras como una unidad; sin embargo, antes de entrar al módulo de transferencia son procesadas por un módulo auxiliar llamado `pretransfer` (véase el apartado 3.3) que se encarga de separar las formas léxicas de las que se componen. De este modo llegan al módulo de transferencia como formas independientes; es el lingüista quien debe deci-

```
<dictionary>
<pardefs>
  ...
  <pardef n="el">
    <e>
      <p>
        <l>el<b/></l>
        <r>l<b/></r>
      </p>
    </e>
    <e>
      <p>
        <l>el</l>
        <r>l</r>
      </p>
      <par n="puntuación"/>
    </e>
  </pardef>
  ...
</pardefs>
<section id="main" type="standard">
  ...
  <e>
    <p>
      <l><a/>de<b/></l>
      <r>de</r>
    </p>
    <par n="el"/>
  </e>
  ...
</section/>
</ditionary>
```

Figura 3.18: Diccionario de postgeneración en el que se observa la información necesaria para la contracción española *de* + *el* = *del* .

```
<e lm="hoy en día">
  <p>
    <l>hoy<b/>en<b/>día</l>
    <r>hoy<b/>en<b/>día<s n="adv"/></r>
  </p>
</e>
```

Figura 3.19: Ejemplo de multipalabra sin flexión en el diccionario morfológico.

```
<e lm="del" r="LR">
  <p>
    <l>del</l>
    <r>de<s n="pr"/><j/>
     el<s n="det"/><s n="def"/>
     <s n="m"/><s n="sg"/></r>
  </p>
</e>
```

Figura 3.20: Ejemplo de entrada para analizar una contracción (la contracción española *del*) en el diccionario morfológico.

dir si hay que volver a unirlas (tarea que debe realizarse en el módulo de transferencia estructural) o si deben continuar como formas independientes a través de los módulos posteriores.

En nuestro sistema, los elementos que forman cada contracción prosiguen como formas independientes, y es tarea del postgenerador realizar la contracción en la lengua meta en caso de que sea necesaria. En cambio, los pronombres enclíticos vuelven a unirse al verbo mediante una regla de transferencia estructural (véase el apartado 3.4), de modo que el verbo más los enclíticos llegan como una sola multiforma léxica al módulo de generación, unidos entre sí mediante el elemento <**j**/>. Por lo tanto, las entradas que contienen pronombres enclíticos no deberán tener ninguna restricción de dirección, como puede verse en el ejemplo de la figura 3.21, que muestra un fragmento del paradigma del verbo ”dar”, concretamente la entrada correspondiente a la forma de infinitivo unida a un pronombre enclítico.

3. El caso más complejo de los que se especifican en este documento es el de las *multipalabras con flexión intercalada* en medio del lema (o formas de ”lema partido”), que se muestra en la figura 3.22. Este tipo

```
<e>
  <p>
    <l>ar</l>
    <r>ar<s n="vblex"/><s n="inf"/><j/></r>
  </p>
  <par n="S__cantar"/>
</e>
```

Figura 3.21: Fragmento del paradigma de flexión del verbo *dar*, que muestra la entrada correspondiente a la forma de infinitivo seguida de un pronombre enclítico. Los pronombres enclíticos están contenidos en el paradigma `S__cantar`. Obsérvese que, a diferencia de la figura 3.20, la entrada es tanto de análisis como de generación.

de multipalabras se caracterizan por tener lemas con una parte afectada por la flexión (a la que llamamos *cabeza del lema*) seguida de una parte invariable (a la que llamamos *cola del lema*). La parte invariable se colocará entre elementos <**g**>, para poder moverla a la posición inmediatamente posterior a la cabeza del lema con el objetivo de obtener el lema de la multipalabra. Es decir, conviene considerar que el lema de multipalabras como *echó de menos*, *echándole de menos*, etc. sea *echar de menos*, ya que será esta forma la que se buscará en el diccionario bilingüe para encontrar su traducción. Esto supondrá un adelantamiento de la cola del lema (*de menos*) detrás de la forma no flexionada de la cabeza del lema (*echar*). Este adelantamiento se realizará en el módulo auxiliar `pretransfer` (véase el apartado 3.3) que se ejecuta antes del módulo de transferencia estructural.

Para entender el ejemplo de la figura 3.22, hay que tener en cuenta que el paradigma que define el verbo *echar* incluye, además de la flexión del verbo, los pronombres enclíticos que van al final de las formas flexionadas del verbo, los cuales aparecen enlazados, en la multiforma léxica resultante, usando el elemento vacío <**j**/>.

Cuando la traducción es también un *lema partido* (por ejemplo, en catalán *trobar a faltar*, con formas como *trobem a faltar*, *trobar-lo a faltar*, etc.), es necesario recolocar la cola en su sitio, tras la forma flexionada más los pronombres enclíticos, en caso de haberlos, e indicar la correspondencia de estos fragmentos invariables del lema (*de menos*, *a faltar*) a ambos lados de la traducción. Así, en el ejemplo de la figura 3.22, el elemento <**g**> sirve para diferenciar el grupo '`<b/>de<b/>menos`' en el diccionario morfológico. En el diccionario

```
<e lm="echar de menos">
  <i>ech</i>
  <par n="aspir/ar__vblex"/> <!-- incluye pronombres enclíticos -->
  <p>
    <l><b/>de<b/>menos</l>
    <r><g><b/>de<b/>menos</g></r>
  </p>
</e>
```

Figura 3.22: Ejemplo de entrada en el diccionario morfológico que contiene un grupo <**g**>.

```
<e>
  <p>
    <l>echar<g><b/>de<b/>menos</g><s n="vblex"/></l>
    <r>trobar<g><b/>a<b/>faltar</g><s n="vblex"/></r>
  </p>
</e>
```

Figura 3.23: Ejemplo de entrada en el diccionario bilingüe que contiene dos grupos <**g**> en correspondencia.

rio bilingüe (véase la figura 3.23), el elemento <**g**> sirve para establecer una correspondencia entre los grupos "<b/>de<b/>menos" y "<b/>a<b/>faltar".

Cuando la traducción no es ningún *lema partido*, no hace falta introducir ningún elemento <**g**> en la cadena correspondiente a la lengua meta.

3.1.3. Generación automática de los módulos de procesamiento léxico

Los cuatro módulos de procesamiento léxico (analizador morfológico, transferencia léxica, generador morfológico y postgenerador) se compilan a partir de los diccionarios mediante un único compilador basado en transductores de letras [14]. Este compilador es mucho más rápido que los utilizados en los sistemas interNOSTRUM [2, 5, 4] y Traductor Universia [7, 17], gracias a la utilización de nuevas estrategias de construcción de transductores y a la minimización de transductores parciales durante la construcción [11].

La división de entradas del diccionario en lema y paradigma permite construir eficazmente transductores de letras mínimos. El compilador aprovecha completamente la factorización que permiten los paradigmas para acelerar la construcción. Como en la mayoría de lenguas europeas las modificaciones de las palabras tienen lugar al final o al principio de las mismas, hemos explotado este hecho para mejorar la velocidad de construcción del transductor mínimo.

Los paradigmas se minimizan antes de ser insertados en el transductor para reducir el tamaño del transductor completo final antes de minimizar. Como antes de minimizar, los paradigmas de los diccionarios de las lenguas con las que hemos trabajado suelen tener unos pocos cientos de estados, la minimización de estos paradigmas es un proceso muy rápido.

Si suponemos que una entrada puede presentar una referencia a un paradigma en cualquier punto de la misma, podríamos pensar en copiar en ese punto el transductor que se calcula en la definición del paradigma. El procedimiento utilizado en *Apertium* se basa en que no siempre es necesario copiar, sino que en determinadas ocasiones es posible reutilizar un paradigma que ya haya sido copiado. En particular, dos o más entradas que comparten un paradigma como sufijo pueden reutilizar la misma copia de ese paradigma y lo mismo sucede cuando ocurre como prefijo. Sin embargo, en general, no es posible reutilizar paradigmas si se encuentran en posiciones intermedias de entradas diferentes, ya que se pueden introducir nuevos sufijos (prefijos) a entradas existentes, lo que hace que la información que se introduce en el transductor no sea consistente con el diccionario, y el transductor generado sería incorrecto (añadiría pares de cadenas que no se encuentran en el lenguaje formal que definen los diccionarios).

La construcción de transductores de letras mínimos se realiza como se explica a continuación. A partir de una transducción de cadenas se puede construir una *secuencia de transducciones de letras* $S(s:t)$ de longitud $N = \max(|s|, |t|)$ que se define como sigue para cada elemento $1 \leq i \leq N$:

$$S_i(s:t) = \begin{cases} (s_i : \theta) & \text{si } i \leq |s| \wedge i > |t| \\ (\theta : t_i) & \text{si } i \leq |t| \wedge i > |s| \\ (s_i : t_i) & \text{en otro caso} \end{cases} \tag{3.1}$$

Hay que destacar que, por construcción, se asegura que no puede existir ningún $(s:t)$ que sea igual a $(\epsilon : \epsilon)$, lo que es crucial para la consistencia del método de construcción.

El método de construcción usa dos procedimientos, el procedimiento de *montaje* que se infiere de la ecuación 3.1 y el de minimización, que se realiza por un algoritmo convencional de minimización [16] de autómatas

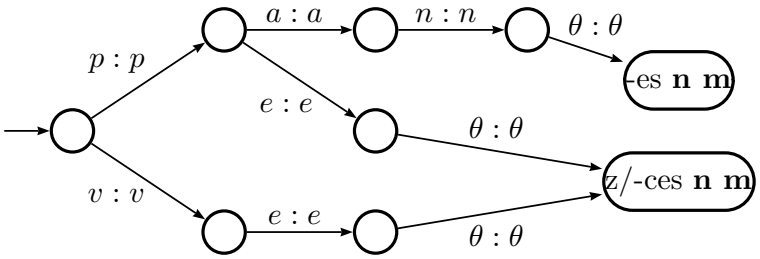

Figura 3.24: Construcción del diccionario como aceptor de prefijos y enlace con paradigmas mediante transiciones $(\theta : \theta)$.

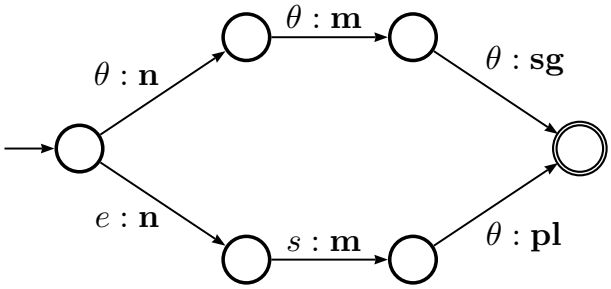

Figura 3.25: Paradigma «-es **n m**» minimizado que se usa en la figura 3.24.

finitos que consiste en invertir, determinizar, volver a invertir y volver a determinizar, tomando como alfabeto del autómata que hay que minimizar el producto cartesiano L y como transición vacía la $(\theta : \theta)$.

En la figura 3.24 se observa un ejemplo simplificado de este montaje. Se introduce transducción por transducción compuesta como en la ecuación 3.1 en un transductor en forma de *aceptor de prefijos* o *trie*, es decir, de manera en la que haya sólo un nodo para cada prefijo común del conjunto de transducciones que forman el diccionario. Con los sufijos de las transducciones (que no se comparten) se crean estados nuevos. En el punto en el que se haga referencia a un paradigma, se crea una réplica de ese paradigma y se enlaza a la entrada del diccionario que se está insertando en el transductor mediante una transducción nula $(\theta : \theta)$.

Cada uno de los paradigmas, en tanto que pueden ser vistos como pequeños diccionarios, se han construido por este mismo procedimiento a su vez y se han minimizado para reducir el tamaño del problema en la construcción del diccionario grande. En las figuras 3.25 y 3.26 se muestra el estado de los dos paradigmas utilizados en la figura 3.24 después de la minimización.

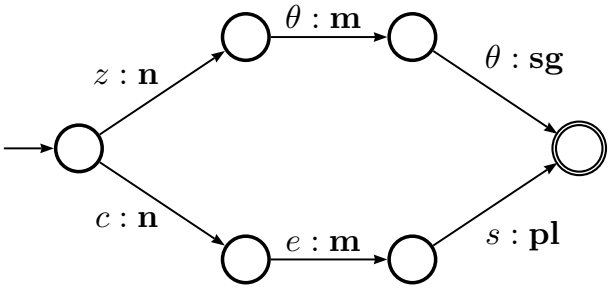

Figura 3.26: Paradigma «z/-ces **n m**» minimizado que se usa en la figura 3.24.

3.2. Desambiguador léxico categorial

3.2.1. Descripción del funcionamiento

El desambiguador léxico categorial está basado en modelos ocultos de Markov [13] de primer orden, es decir, en información de naturaleza estadística. Los estados del modelo de Markov representan categorías gramaticales y los observables son clases de ambigüedad [3], esto es, conjuntos de categorías gramaticales.

A pesar de trabajar con información estadística, el entrenamiento y el comportamiento actual del desambiguador mejoran si se introducen restricciones que impiden determinadas secuencias de categorías (en los modelos de primer orden, estas secuencias solo pueden implicar a dos categorías); por ejemplo, en español o catalán una preposición nunca puede ir seguida de un verbo conjugado en forma personal, restricción que es de gran ayuda cuando la forma que sigue a la preposición es ambigua y uno de sus análisis morfológicos es verbo en forma personal (p.e., *de trabajo*, *en libertad*, etc.). En el fichero de definición del desambiguador se declaran explícitamente estas restricciones, a veces en forma de *prohibiciones* y otras en forma de *obligaciones*.

Las etiquetas gramaticales con las que trabaja el desambiguador no son las mismas que se utilizan en el analizador morfológico. Normalmente, la información que entrega el analizador morfológico es demasiado detallada para la desambiguación léxica categorial (por ejemplo, para la mayoría de los propósitos, es suficiente agrupar en una misma categoría todos los substantivos comunes, independientemente de su género y su número). El empleo de etiquetas más finas no mejora los resultados, al tiempo que incrementa considerablemente el número de parámetros a estimar y hace más acusado el problema de la escasez de recursos lingüísticos tales

como textos desambiguados a mano. Por este motivo, en el fichero del desambiguador se especifica cómo agrupar las etiquetas *finas* (o específicas) proporcionadas por el analizador morfológico en etiquetas *gruesas* —que llamaremos *categorías*— más generales que se emplearán en la desambiguación léxica categorial. Además de las categorías gruesas, también pueden definirse etiquetas lexicalizadas. En la bibliografía se describen básicamente dos tipos de lexicalizaciones, las que añaden nuevos observables y las que además añaden nuevos estados al modelo de Markov [12]; el desambiguador léxico categorial de Apertium utiliza este último tipo de lexicalización.

Cabe recordar que, pese a trabajar con categorías *gruesas*, el desambiguador proporciona a su salida etiquetas finas como las del analizador morfológico. De hecho, en ocasiones, puede suceder que el analizador morfológico entregue, para una palabra dada, dos o más etiquetas finas que pueden agruparse bajo una misma categoría: p. e., en español *cante* puede ser la 1ª o la 3ª persona del presente de subjuntivo del verbo *cantar*; las dos etiquetas finas, `<vblex><prs><p1><sg>` y `<vblex><prs><p1><sg>`, se agrupan bajo la categoría `VLEXSUBJ` (*verbo subjuntivo*). En este caso, una de las dos etiquetas finas es descartada; en el fichero de definición del desambiguador se puede definir qué etiqueta fina de las que componen una etiqueta gruesa será la entregada tras la desambiguación.

3.2.2. Datos para el desambiguador léxico categorial

Introducción

A continuación se describe el formato de los ficheros en los que se especifica cómo agrupar las etiquetas *finas* proporcionadas por el analizador morfológico en etiquetas *gruesas* más generales. En dichos ficheros, además, se pueden especificar *restricciones* que ayuden a la estimación del modelo estadístico encargado del proceso de desambiguación léxica, así como reglas de preferencia para el caso de que a dos etiquetas finas les corresponda la misma categoría.

El desambiguador asume que a su entrada las palabras se encontrarán convenientemente delimitadas, tal como se especifica en el formato del flujo de datos entre módulos (capítulo 2). El formato de los datos entregados

por el analizador morfológico, a grandes rasgos, es como sigue:

$$\begin{aligned} \text{formaanalizada} &\rightarrow \text{multiformaléxica} \; [\; \text{multiformaléxica} \;]^* \\ \text{multiformaléxica} &\rightarrow \text{formaléxica} \; [\; \text{formaléxica} \;]^* \; \text{cola-lema?} \\ \text{formaléxica} &\rightarrow \text{lema etiquetafina} \\ \text{cola-lema} &\rightarrow \text{lema} \\ \text{etiquetafina} &\rightarrow \text{símbolograma} \; [\; \text{símbolograma} \;]^* \end{aligned} \tag{3.2}$$

donde:

- *formaanalizada* es toda la información que se entrega para cada forma superficial a la salida del analizador morfológico
- *multiformaléxica* es una secuencia de una o más formas léxicas seguida, opcionalmente, por una cola invariable como en el caso de algunas multipalabras (*cántale las cuarenta*).
- *formasléxicas*[4] son unidades compuestas por un lema y una o más etiquetas con la información de la salida del analizador
- *cola-lema* está compuesta por uno o más lemas[5] que forman la parte invariable de una multipalabra. La cola de una multipalabra está formada por el lema o los lemas sin flexión que siguen a los lemas con flexión. Por ejemplo, la multipalabra *cantar las cuarenta* puede tomar las formas *cántale las cuarenta*, *(le) cantaré las cuarenta*, *cantándole las cuarenta*, etc. La cola sería en este caso *las cuarenta* (véase la página 42 para más información).
- *etiquetafina* corresponde a uno o más símbolos gramaticales (*símbolograma*).

Por ejemplo, la entrada correspondiente a la forma superficial ambigua española *correos* tendría dos multiformas léxicas; la primera multiforma léxica tendría una única forma léxica, con lema *correo* y una etiqueta fina compuesta de los símbolos gramaticales *nombre común*, *masculino*, *plural*; la segunda multiforma léxica sería una secuencia de dos formas léxicas, una con lema *correr* y etiqueta fina compuesta de los símbolos gramaticales *verbo léxico*, *imperativo*, *segunda persona*, *plural* y la otra con lema *vosotros*, y etiqueta fina compuesta por los símbolos *pronombre*, *enclítico*, *segunda persona*, *masculino-femenino*, *plural*.

[4] Separadas entre ellas por un delimitador que se corresponde con el elemento `<j/>` (ver página 36).

[5] Separados entre ellos mediante el elemento `<b/>` (ver página 35).

Especificación del formato

El formato del fichero (codificado en XML) se especifica mediante la DTD que se observa en el apéndice A.2.

El significado de las etiquetas es como sigue:

`tagger` : es el elemento raíz; su atributo obligatorio `name` sirve para especificar el nombre del etiquetador generado a partir del fichero.

`tagset` : define el etiquetario *grueso* de las categorías con el que trabaja el desambiguador. Las categorías se definen a partir de las etiquetas finas que proporciona el analizador morfológico a su salida.

`def-label` : define una categoría o etiqueta gruesa (cuyo nombre se indica en el atributo obligatorio `name`) en términos de listas de etiquetas finas definidas mediante uno o más elementos `tags-item`; un atributo opcional `closed` indica si la categoría es cerrada; de ser cerrada se entiende que una palabra desconocida nunca puede pertenecer a esta categoría.[6]

Las categorías más específicas *deben* definirse antes de las más generales. Cuando la definición de una categoría general incluye implícitamente la de una categoría específica definida previamente, se entiende que se refiere a todos los casos *excepto* los definidos por las categorías más específicas.

`tags-item` : permite definir una etiqueta fina en términos de una secuencia de símbolos gramaticales. Mediante el atributo obligatorio `tags` se define la secuencia de símbolos gramaticales que conforman la etiqueta fina. En dicha secuencia los símbolos gramaticales se separan por puntos y el asterisco "`*`" se utiliza para indicar que en determinada posición puede aparecer cualquier secuencia de símbolos. También se pueden especificar categorías lexicalizadas indicando en el atributo `lemma` el lema deseado.

`def-mult` : define categorías especiales (*multicategorías*) formadas por varias categorías, para tratar el caso de que una entrada presente multiformas léxicas como las definidas en la sección anterior. Cada categoría se define como un conjunto de secuencias (`sequence`) válidas de categorías definidas previamente o de etiquetas finas. Su uso es apropiado para contracciones, verbos con enclíticos, etc.

[6] Las categorías cerradas son aquellas que no crecen cuando se crea nuevo léxico: preposiciones, determinantes, conjunciones, etc.

sequence : define una secuencia de elementos que pueden ser categorías (`label-item`) o etiquetas finas (`tags-item`). El utilizar directamente etiquetas finas es útil si se desea utilizar una secuencia de símbolos gramaticales que no forma parte de una etiqueta fina ya definida o que supone una especialización mayor de una etiqueta fina ya creada.

label-item : permite hacer referencia a una categoría o etiqueta gruesa ya definida con anterioridad y que se especifica en el atributo obligatorio `label`.

forbid : esta sección (opcional) permite definir restricciones en forma de secuencias de categorías `label-sequence` que no pueden darse en el idioma en cuestión. En la versión actual, al estar el desambiguador basado en modelos ocultos de Markov de primer orden, las secuencias sólo pueden estar formadas por *dos* `label-items`.

label-sequence : define una secuencia de categorías (`label-item`).

enforce-rules : esta sección (opcional) permite definir restricciones en forma de obligaciones.

enforce-after : define una restricción que obliga a que a una determinada categoría sólo puedan seguirle categorías pertenecientes al conjunto (`label-set`) de categorías que se define. Nótese que este tipo de restricciones es equivalente a definir varias secuencias (`label-sequence`) prohibidas (`forbid`) con la categoría definida mediante el atributo obligatorio `label` y el resto de categorías no pertenecientes al conjunto definido en `label-set`. Por esta razón, este tipo de restricción debe usarse con mucha cautela.

label-set : define un conjunto de categorías (`label-items`).

preferences : permite definir prioridades en cuanto a qué etiquetas finas se deben producir a la salida del desambiguador cuando una o más etiquetas finas sean asignadas a la misma categoría.

prefer : especifica que en caso de conflicto entre varias etiquetas finas asignadas a la misma categoría, el desambiguador debe proporcionar a la salida la etiqueta definida mediante el atributo obligatorio `tags`. Si la categoría contiene más de una de las etiquetas finas definidas en estos elementos `prefer`, se entregará la que esté definida antes.

Las figuras 3.27 y 3.28 muestran con un ejemplo las partes más importantes de un fichero de especificación de desambiguador definido por la DTD que acabamos de describir.

3.2.3. Consideraciones sobre el entrenamiento del desambiguador léxico categorial

El entrenamiento del desambiguador léxico categorial podrá hacerse tanto de forma supervisada, utilizando textos desambiguados a mano, como de forma no supervisada, con textos ambiguos.

Cuando el entrenamiento se realice a partir de textos ambiguos (de forma no supervisada) el formato de dicho texto se podrá obtener de forma automática a partir de un corpus de texto llano de la lengua deseada utilizando el analizador morfológico del sistema; en este caso el formato de las formas del texto será como el definido en el esquema 3.3 (puede leerse su descripción en la página 51). Como muestra el esquema, puede haber más de un análisis para cada forma superficial analizada (una *formaanalizada* puede dar como resultado más de una *multiformaléxica*).

$$\begin{array}{rcl} \text{formaanalizada} & \rightarrow & \text{multiformaléxica [multiformaléxica]}^* \\ \text{multiformaléxica} & \rightarrow & \text{formaléxica [formaléxica]}^* \text{ cola-lema?} \\ \text{formaléxica} & \rightarrow & \text{lema etiquetafina} \\ \text{cola-lema} & \rightarrow & \text{lema} \\ \text{etiquetafina} & \rightarrow & \text{símbolograma [símbolograma]}^* \end{array} \tag{3.3}$$

Para el entrenamiento supervisado se necesita texto desambiguado a mano. El formato de las formas de estos textos será como el proporcionado por el analizador morfológico (véase el capítulo 2) con la salvedad de que, al tratarse de texto ya desambiguado, nunca habrá más de una forma léxica asociada a cada forma superficial como se muestra en 3.4 (una *formadesambiguada* consistirá siempre en una única *multiformaléxica*).

$$\begin{array}{rcl} \text{formadesambiguada} & \rightarrow & \text{multiformaléxica} \\ \text{multiformaléxica} & \rightarrow & \text{formaléxica [formaléxica]}^* \text{ cola-lema?} \\ \text{formaléxica} & \rightarrow & \text{lema etiquetafina} \\ \text{cola-lema} & \rightarrow & \text{lema} \\ \text{etiquetafina} & \rightarrow & \text{símbolograma [símbolograma]}^* \end{array} \tag{3.4}$$

Por último, para el entrenamiento también se precisa del diccionario de la lengua implicada. Este diccionario se emplea para determinar conjun-

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE tagger SYSTEM "tagger.dtd">
<tagger name="es-ca">
<tagset>
   <def-label name="adv">
      <tags-item tags="adv"/>
   </def-label>
   <def-label name="detnt" closed="true">
      <tags-item tags="detnt"/>
   </def-label>
   <def-label name="detm" closed="true">
      <tags-item tags="det.*.m"/>
   </def-label>
   <def-label name="vlexpfci">
      <tags-item tags="vblex.pri"/>
      <tags-item tags="vblex.fti"/>
      <tags-item tags="vblex.cni"/>
   </def-label>
   <def-mult name="infserprnenc" closed="true">
      <sequence>
         <label-item label="vserinf"/>
         <label-item label="prnenc"/>
      </sequence>
      <sequence>
         <label-item label="vserinf"/>
         <label-item label="prnenc"/>
         <label-item label="prnenc"/>
      </sequence>
   </def-mult>
   <def-mult name="prepdet" closed="true">
      <sequence>
         <label-item label="prep"/>
         <tags-item tags="det.def.m.sg"/>
      </sequence>
   </def-mult>
</tagset>
<!-- ... -->
```

Figura 3.27: Ejemplo de fichero de definición del desambiguador (continúa en la figura 3.28).

```
<!-- ... -->
<forbid>
   <label-sequence>
      <label-item label=="prep"/>
      <label-item label=="vlexpfci"/>
   </label-sequence>
   <!-- ... -->
</forbid>
<enforce-rules>
   <enforce-after label=="prnpro">
      <label-set>
         <label-item label=="prnpro"/>
         <label-item label=="vlexpfci"/>
         <!-- ... -->
      </label-set>
   </enforce-after>
   <!-- ... -->
</enforce-rules>
<preferences>
   <prefer tags="vblex.pii.3.sg"/>
   <prefer tags="vbser.pii.3.sg"/>
   <!-- ... -->
</preferences>
</tagger>
```

Figura 3.28: Ejemplo de fichero de definición del desambiguador (viene de la figura 3.27).

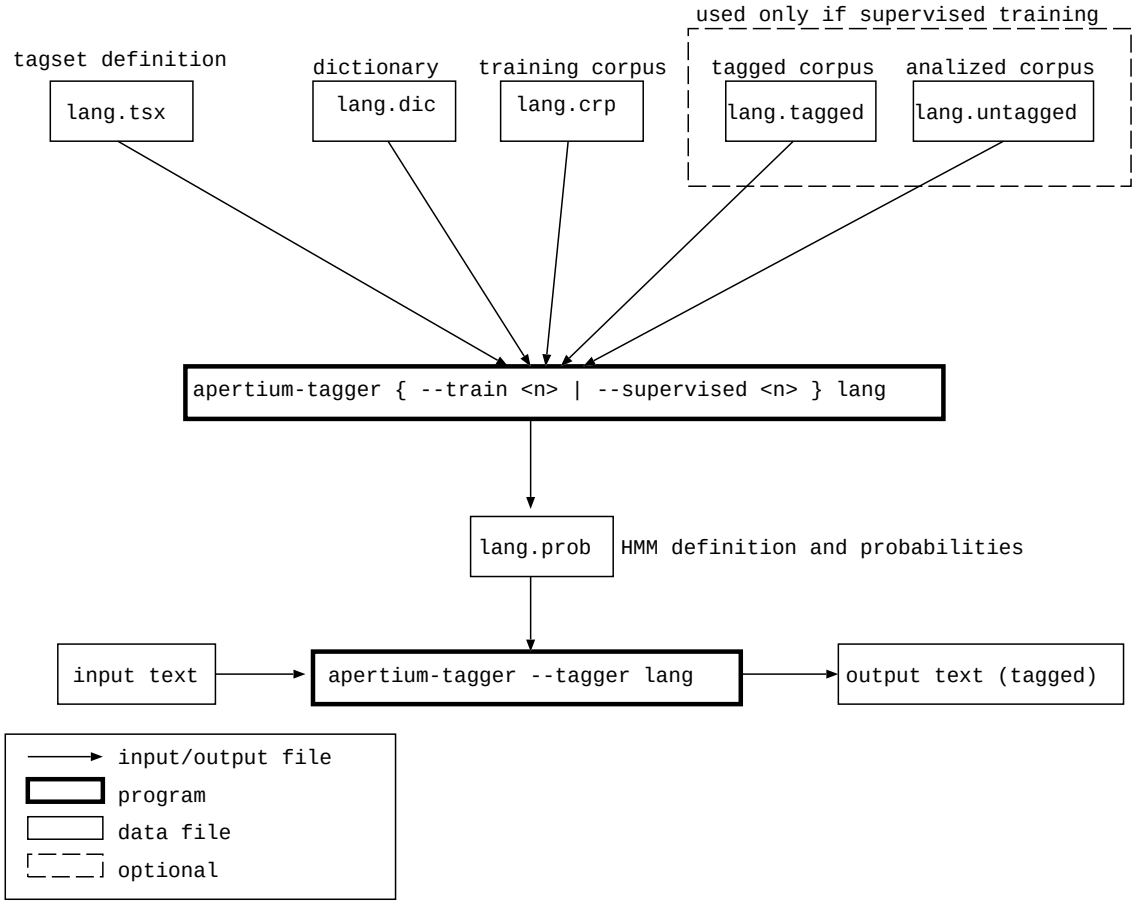

Figura 3.29: Esquema de dependencias del desambiguador léxico categorial.

tamente con la especificación del etiquetario a emplear las distintas clases de ambigüedad con las que trabaja el desambiguador.

La figura 3.29 muestra el esquema de dependencias para el entrenamiento y el uso del desambiguador.

3.3. Módulo auxiliar: módulo de preproceso de la transferencia

3.3.1. Motivación

El módulo de preproceso de la transferencia `pretransfer`, se ocupa de fragmentar las unidades multipalabra compuestas (ver la página 42) y mover ciertas partículas de lugar en las multipalabras con flexión intercalada o de *lema partido*. Este módulo procesa la salida del etiquetador y genera una entrada apta para el módulo de transferencia. Este procesamiento es necesario por varias razones:

- Para que el módulo de transferencia pueda tratar estas unidades por separado para tratar, por ejemplo, fenómenos de cambio de pronombres enclíticos por pronombres proclíticos o viceversa.

- Para que en el diccionario bilingüe sólo haya que almacenar información de los lemas que se traducen. Si las partículas que componen una unidad multipalabra se presentan conjuntamente en el diccionario bilingüe, este diccionario tiene que tener una entrada para cada una de estas composiciones. Mediante la separación se consigue no tener entradas para multipalabras que incluyan la flexión en el diccionario bilingüe.

3.3.2. Comportamiento y ejemplo

El funcionamiento del programa consiste en sustituir cada `<j/>` en el diccionario, es decir, cada `+` en el flujo de datos, por un carácter de fin de palabra, un espacio en blanco y un carácter de comienzo de palabra. Además, si se trata de una multipalabra de lema partido, se mueve la cola a la posición que queda entre el final de la primera palabra de la multipalabra y la primera etiqueta morfológica que se corresponde con esta palabra.

La responsabilidad de generar una salida de este módulo que tenga el orden original que acepta el generador se deja a las reglas correspondientes del módulo de transferencia, así como la de crear las multipalabras compuestas que sean necesarias en la lengua meta. El generador trabaja, en general, con las mismas multipalabras que el analizador morfológico, y con los elementos en el mismo orden, y es por eso que esta tarea hay que realizarla en el módulo de transferencia.

A continuación se muestra el resultado de aplicar este proceso a la multipalabra compuesta *darlo*:

```
$ pretransfer
^dar<vblex><inf>+lo<prn><enc><p3><m><sg>$      ⟵ entrada
^dar<vblex><inf>$ ^lo<prn><enc><p3><m><sg>$    ⟵ salida
```

Como se puede ver, sencillamente se trata de dividir las formas léxicas de una unidad multipalabra compuesta en formas léxicas individuales.

Cuando se trata de una multipalabra con lema partido, se opera como se ve en el siguiente ejemplo para la multipalabra *echarte de menos*:

```
$ pretransfer
^echar<vblex><inf>+te<prn><enc><p2><m><sg># de menos$
^echar# de menos<vblex><inf>$ ^te<prn><enc><p2><m><sg>$
```

Aquí, además de dividir las formas léxicas, se produce un movimiento de la cola invariable del lema a la posición anunciada. Como vemos, con el movimiento de esta cola invariable, las unidades semánticas se mantienen, ya que se puede considerar *echar de menos* como una unidad verbal con significado propio.

3.4. Módulo de transferencia estructural

3.4.1. Descripción del funcionamiento

El diseño del lenguaje y del compilador que se usan para generar el módulo encargado de la transferencia estructural está fuertemente inspirado en el lenguaje MorphTrans descrito en [5] y usado por los sistemas de TA español–catalán interNOSTRUM [2, 5, 4] y español–portugués Traductor Universia [7, 17], desarrollados por el grupo Transducens de la Universitat d'Alacant.

La tarea de transferencia se realiza a partir de patrones que representan secuencias de longitud fija de formas léxicas (véase página 7) de la lengua origen (FLLO); una secuencia sigue un cierto patrón cuando contiene la secuencia de categorías léxicas correspondiente. Los patrones no tienen por qué ser constituyentes o sintagmas en un sentido sintáctico estricto, sino que son meras concatenaciones de formas léxicas que pueden necesitar un procesamiento conjunto adicional a la simple traducción palabra por palabra, debido a las divergencias gramaticales existentes entre la LO y la LM (cambios de género y número, reordenamientos, cambios preposicionales, etc). Los patrones que forman el conjunto definido para una lengua determinada se escogen de manera que definan las transformaciones estructurales más comunes. Cuando las lenguas origen y meta son sintácticamente similares, como es el caso del español, el catalán y el gallego, reglas simples basadas en secuencias de categorías léxicas permiten obtener una calidad razonable en la traducción.

El módulo de transferencia detecta en la LO secuencias de formas léxicas que concuerden con algun patrón de los previamente definidos en un catálogo de patrones y los procesa aplicando unas reglas de transferencia estructural al mismo tiempo que realiza la transferencia léxica consultando el diccionario bilingüe.

La *detección de patrones* funciona como sigue: si el módulo de transferencia empieza a procesar la $i$ésima FLLO del texto, l_i, éste intenta comparar la secuencia de FLLO $l_i, l_{i+1}, \ldots$ con todos los patrones que hay en su catálogo: escoge el patrón más largo que coincida, procesa la secuencia detectada (mirar más abajo), y el proceso continúa en la FLLO l_{i+k}, donde k es la longitud del patrón que se acaba de procesar. Si ningún patrón coincide con la secuencia que empieza con la FLLO l_i, ésta se traduce como una palabra aislada y el proceso comienza de nuevo con la FLLO l_{i+1} (es decir, cuando ningún patrón es aplicable, el sistema recurre a la traducción palabra por palabra). Nótese que cada FLLO es procesada una sola vez: los patrones no se solapan; por tanto, el procesamiento se realiza de izquierda

a derecha y en fragmentos bien definidos.

Para el *procesamiento de patrones*, el sistema toma la secuencia de FLLO detectada y construye (utilizando un programa para la consulta del diccionario bilingüe) una secuencia de formas léxicas en la lengua meta (FLLM) obtenidas por aplicación de las operaciones descritas en la regla que afecta al patrón (reordenación, adición, sustitución o eliminación de palabras, cambios en la flexión, etc.). La información que no cambia se copia automáticamente de la lengua origen a la lengua meta. Los datos resultantes, es decir, los lemas más todas las etiquetas morfológicas correspondientes, se envían al generador, el cual se encarga de crear las formas flexionadas.

Como ejemplo, la secuencia en español *una señal inequívoca*, que llegaría desde el desambiguador al módulo de transferencia en el siguiente formato [7]:

```
^uno<det><ind><f><sg>$
^señal<n><f><sg>$
^inequívoco<adj><f><sg>$
```

sería detectado como patrón por una regla para determinante–nombre–adjetivo. El módulo de transferencia consultaría el diccionario bilingüe para establecer las correspondencias en catalán y, como detectaría un cambio de género en la palabra *señal* (su equivalente catalán *señal* es masculino), propagaría este cambio al determinante y al adjetivo para ofrecer la secuencia de salida:

```
^un<det><ind><m><sg>$
^senyal<n><m><sg>$
^inequívoc<adj><m><sg>$
```

que el módulo de generación convertiría en la forma catalana flexionada: *un senyal inequívoc*.

La mayoría de reglas se encargan de garantizar la concordancia de género y número en varios sintagmas nominales (SN) sencillos (determinante–nombre, determinante–nombre–adjetivo, determinante–adjetivo–

[7] El ejemplo se ha elegido de manera que no tenga superblancos con información de formato, para que quede más clara la parte lingüística de la transformación. Véase el capítulo 2

nombre, determinante–adjetivo, etc.), siempre que haya concordancia entre las FLLO del patrón detectado. Estas reglas son necesarias bien porque el nombre cambia de género o número entre la LO y la LM (como en el ejemplo que acabamos de describir) o bien porque hay que determinar el género o el número en la LM debido a que en la LO era ambiguo para alguna de las palabras (por ejemplo, el determinante *cap* en catalán se puede traducir al español por *ningún* (masc.) o *ninguna* (fem.) según el nombre al que acompañe: *cap cotxe* (`ca`) → *ningún coche* (`es`) y *cap casa* (`ca`) → *ninguna casa* (`es`). Además, hay definidas otras reglas para solucionar problemas frecuentes de transferencia entre el español, el catalán y el gallego, como, entre otras:

- reglas para cambiar preposiciones en determinadas construcciones: *en Barcelona* (`es`) → *a Barcelona* (`ca`); *consiste en hacer* (`es`) → *consisteix a fer* (`ca`);
- reglas para añadir/quitar la preposición *a* en determinadas construcciones modales del gallego con *ir* y *vir*: *vai comprar* (`gl`) → *va a comprar* (`es`);
- reglas para los artículos delante de nombres propios: *ve la Marta* (`ca`) → *viene Marta* (`es`);
- reglas léxicas, por ejemplo, para decidir la traducción correcta del adverbio *molt* (`ca`) al español (*muy, mucho*) o la del adjetivo *primeiro* (`gl`) o *primer* (`ca`) al español (*primer, primero*);
- reglas para reposicionar los pronombres átonos o clíticos, cuya distribución en gallego es diferente que en español (proclítico en gallego y enclítico en español o viceversa): *envioume* (`gl`) → *me envió* (`es`); *para nos dicir* (`gl`) → *para decirnos* (`es`).

Las *multipalabras* (sus diferentes tipos están explicados en la página 42) son procesadas de una manera especial en este módulo:

- Las *multipalabras sin flexión*, formadas por una sola forma léxica, no necesitan ningún procesamiento específico, ya que son tratadas como las demás FL.
- Cuando se trata de *multipalabras compuestas*, es decir, multipalabras formadas por varias *formas léxicas*, cada una con un símbolo gramatical propio y unidas mediante el elemento `<j>` en la entrada del diccionario (que se corresponde con el símbolo '+' en el flujo de datos),

el módulo auxiliar `pretransfer` (véase 3.3), situado antes del presente módulo, separa las distintas formas léxicas para que lleguen aquí como FL independientes. Si se desea volver a unirlas para que lleguen al generador como multipalabras (como es el caso de los pronombres enclíticos en nuestro sistema), hay que hacerlo mediante una regla de transferencia, utilizando el elemento <**mlu**> (descrito más adelante, en el apartado **??**). En la página 120 puede verse un ejemplo de regla para la unión de los pronombres enclíticos al verbo.

- En el caso de las *multipalabras con flexión intercalada*, el módulo `pretransfer` adelanta la cola del lema (la parte invariable) y la coloca detrás de la cabeza del lema (la forma que se flexiona), para que sea posible buscar la multipalabra en el diccionario bilingüe. Este tipo de multipalabras deben procesarse mediante una regla de transferencia estructural, que recoloque la cola del lema en la posición adecuada. Esto se realiza utilizando los atributos `lemh` (*lemma head* o "cabeza del lema" y `lemq` (*lemma queue* o "cola del lema") del elemento <**clip**> a la salida de la regla. Véase la página 77 para obtener una explicación más detallada del uso de este elemento, y la página 120 para ver dos reglas en que se utilizan estos atributos.

3.4.2. Datos: especificación del formato de las reglas de transferencia estructural

En esta sección se explica el formato en el que se escriben las reglas de transferencia estructural. En el apéndice A.3 se ofrece la descripción formal (DTD).

El archivo de reglas de transferencia estructural tiene dos partes bien diferenciadas: una de declaración de los elementos que se utilizarán en las reglas y otra de reglas propiamente dichas.

En la parte de **declaración** encontramos:

- Una serie de declaraciones de *categorías léxicas* que especifican aquellas formas léxicas que serán tratadas como una categoría particular y que serán detectadas en los patrones. Hemos de destacar que el lingüista puede incluir cualquier información de la forma léxica para definir una categoría; las categorías pueden ser muy genéricas (p.e. todos los nombres) o muy específicas (p.e. sólo aquellos determinantes que son demostrativos femeninos plurales).

- Una serie de declaraciones de los *atributos* que queremos detectar en las formas léxicas (como el *género*, el *número*, la *persona* o el *tiempo*), para realizar con ellos las operaciones de transformación pertinentes y enviar la información resultante a la salida de las reglas. En la declaración de los atributos se incluye el nombre del atributo y los posibles valores que puede tomar este atributo en una forma léxica (por lo general se corresponden con los símbolos morfológicos que la caracterizan): por ejemplo, el atributo de *número* puede tomar los valores de *singular*, *plural*, *singular-plural* (para formas léxicas invariables, como *crisis* en español) y *número por determinar* (para formas léxicas de la LM con distinción *singular–plural* cuyo número no se puede determinar en la traducción al ser la forma léxica de la LO invariable en género, véase la explicación de la página 39). Si dentro de la regla, fuera del patrón, debe hacerse referencia a alguna de las categorías léxicas definidas en el punto anterior (para someterlas a acciones o comprobaciones), deberán definirse también atributos para las mismas.

- Una serie de declaraciones de *variables globales*, que se usan para transferir valores de atributos activos dentro de cada regla o de una regla a las que se apliquen posteriormente.

- Una sección de *definición de listas de cadenas*, generalmente listas de lemas, que serán utilizados para buscar en ellas un elemento que corresponda para realizar una transformación determinada.

- Una serie de declaraciones de *macroinstrucciones*; las macroinstrucciones contienen secuencias de instrucciones de operaciones frecuentes y pueden incluirse en varias reglas (por ejemplo, una macroinstrucción para asegurar la concordancia de género y número entre dos formas léxicas de un mismo patrón).

En las **reglas de transferencia estructural** encontramos:

- La definición del patrón que será detectado, especificado como una secuencia de categorías léxicas tal como se han definido en la parte de declaración. Cabe resaltar que, si la secuencia de formas léxicas concuerda con dos reglas diferentes, primero, prevalece la más larga y, segundo, para reglas de la misma longitud, prevalece la regla definida en primer lugar.

- La parte de proceso de las reglas, en la que especifican las acciones a realizar sobre las FLLO y se construye el patrón en la LM.

A continuación se explican detalladamente las características de cada uno de los elementos utilizados.

Elemento <transfer>

Es el elemento raíz y contiene todos los demás elementos del archivo de reglas de transferencia estructural.

Elemento de sección de definición de categorías <section-def-cats>

En esta sección se definen las categorías léxicas que se usarán para crear los patrones utilizados en las reglas. Cada definición se realiza con un <**def-cat**>.

Elemento de definición de categorías <def-cat>

Cada definición de categoría tiene un nombre `n` obligatorio como por ejemplo `det`, `adv`, `prep`, etc. y una lista de categorías (<**cat-item**>) que la definen. El nombre de la categoría no puede contener acentos.

Elemento de categoría <cat-item>

Mediante este elemento se definen las categorías léxicas que se utilizarán en los patrones, es decir, que se desea detectar en el texto origen. Estas categorías se definen a partir de una subsecuencia de las etiquetas finas (ver definición en la página 50) que entregan tanto el analizador morfológico como el desambiguador[8].

Cada elemento <**cat-item**> tiene un atributo obligatorio `tags` cuyo valor es una secuencia de símbolos gramaticales separados por puntos; esta secuencia es una subsecuencia de la etiqueta fina, es decir, de la secuencia de símbolos gramaticales que define cada posible forma léxica

[8]Hay que tener en cuenta que a lo largo de los distintos módulos de procesamiento lingüístico se utilizan distintas categorizaciones léxicas: en los diccionarios morfológicos, los lemas se acompañan de una etiqueta fina (por ejemplo, *<n><m><pl>* para los nombres masculinos en plural); el desambiguador categorial agrupa estas etiquetas finas en etiquetas más generales (por ejemplo, la categoría `NOM` para todos los nombres), aunque a su salida vuelve a entregar la etiqueta fina completa de cada FL; finalmente, en el módulo de transferencia, las etiquetas finas de las FL se agrupan otra vez en categorías más generales (aunque pueden definirse también categorías detalladas) según el tipo de formas léxicas que se quiera detectar en los patrones.

```
<def-cat n="nom"/>
  <cat-item tags="n.*"/>
</def-cat>

<def-cat n="que"/>
  <cat-item lemma="que" tags="cnjsub"/>
  <cat-item lemma="que" tags="rel.an.mf.sp"/>
</def-cat>
```

Figura 3.30: Uso del elemento <**cat-item**> para definir dos categorías, una para nombres sin especificar ningún lema (*nom*) en la que se incluyen todas las formas léxicas cuyo primer símbolo morfológico sea *n*, y otra con lema asociado (*que*), que tiene más de un símbolo gramatical asociado, para incluir el *que* conjunción y el *que* relativo.

entregada por el desambiguador léxico categorial. De este modo, una categoría representa un determinado conjunto de formas léxicas. Tenemos que definir tantas categorías distintas como tipos de formas léxicas queramos detectar en los patrones. Así, si nos interesa detectar todos los nombres para realizar operaciones con ellos, crearemos una categoría definida a partir del símbolo gramatical `n`. Por otro lado, si nos interesa detectar todos los nombres femeninos en plural, deberemos definir una categoría mediante los símbolos `n f` y `pl`.

Cuando un símbolo gramatical utilizado para definir una categoría viene seguido por otros símbolos gramaticales en el grupo de lemas que se quiere incluir, se utiliza el carácter `"*"`. Por ejemplo, `tags="n.*"` abarca todas las formas léxicas que contengan este símbolo, como `casa<n><f><pl>` o `coche<n><m><sg>`. En cambio, cuando detrás del símbolo utilizado no puede venir ningún otro símbolo, no se incluye el asterisco: por ejemplo, `tags="adv"` abarcará todos los adverbios. También puede utilizarse el asterisco para indicar la existencia de símbolos precedentes: `tags="*.f.*"` incluye a todas las formas léxicas femeninas, sean de la categoría que sean. Además, puede utilizarse un atributo opcional, `lemma`, para definir formas léxicas a partir de su lema (ver figura 3.30).

Elemento de sección de definición de atributos de categoría <section-def-attrs>

En esta sección se definen los atributos que se extraerán de las categorías detectadas por el patrón y que se utilizarán en la parte de acción de las reglas. Cada atributo se define mediante una etiqueta <**def-attr**>.

Elemento de definición de atributos de categoría <def-attr>

Cada <**def-attr**> define un atributo con información morfológica (tanto información flexiva –género, nombre, persona, etc.–, como categorial –verbo, adjetivo, etc–) mediante una lista de elementos de atributo de categoría (<**attr-item**>) y tiene un nombre único obligatorio `n`. Un atributo se define, por lo tanto, a partir de los símbolos morfológicos que pueden encontrarse en una forma léxica dada. Cada atributo extrae, de las formas léxicas del patrón, los símbolos que éstas contienen de entre los posibles valores dados.

Elemento de atributo de categoría <attr-item>

Cada elemento de atributo de categoría representa uno de los valores posibles que puede tomar el atributo (por ejemplo, el atributo de número `nbr` puede tomar los valores singular `sg`, plural `pl`, singular–plural `sp` y número por determinar `ND`). Estos valores son una subsecuencia de las etiquetas morfológicas que caracterizan a las formas léxicas y se indican en el atributo `tags` del elemento, separadas por punto si hay más de una. En la figura 3.31 puede verse un ejemplo de utilización de este elemento para el atributo de *número* y el atributo de *nombre*. Compárese la definición del atributo de nombre en esta figura (con todos los valores posibles y sin asteriscos) con la definición de la categoría de nombre de la figura 3.30.

Elemento de sección de definición de variables <section-def-vars>

En esta sección se definen (mediante <**def-var**>) las variables globales de tipo cadena que se utilizarán para transferir información dentro de una regla y de una regla a otra (por ejemplo, para poder transmitir información de género o número entre dos patrones).

Elemento de definición de variables <def-var>

La definición de una variable global de tipo cadena tiene un nombre único obligatorio `n` que se utilizará para referirse a la misma dentro de las reglas. Las variables transmiten cadenas que describen información de estado, como la existencia de concordancia entre dos elementos, la detección de un signo de interrogación en la LO que haya que eliminar en LM, etc.

```
<def-attr n="nbr"/>
  <attr-item tags="sg"/>
  <attr-item tags="pl"/>
  <attr-item tags="sp"/>
  <attr-item tags="ND"/>
</def-attr>

<def-attr n="a_nom"/>
  <attr-item tags="n"/>
  <attr-item tags="n.acr"/>
</def-attr>
```

Figura 3.31: Definición del atributo de categoría *número* `nbr`, que puede tomar los valores *singular*, *plural*, *singular-plural* o *número por determinar*, así como del atributo de categoría *nombre*, que puede tomar los valores de los símbolos *n* o *n acr*.

Elemento de sección de definición de listas de cadenas `<section-def-lists>`

En esta sección se definen (mediante <**def-list**>) las listas que se utilizarán para realizar búsquedas de cadenas. Estas listas se pueden utilizar para agrupar lemas de palabras que tengan alguna característica común (por ejemplo, verbos que expresen movimiento, adjetivos que expresen estados de ánimo, etc.). Esta sección es opcional.

Elemento de definición de listas de cadenas `<def-list>`

Este elemento sirve para ponerle nombre a la lista de cadenas mediante el atributo `n` y para encapsular la lista que se define mediante uno o más elementos <**list-item**>. Puede verse un ejemplo de uso en la figura 3.32.

Elemento de miembro de lista de cadenas `<list-item>`

Este elemento define, mediante el valor del atributo `v`, la cadena concreta que se incluye en la definición de la lista que engloba a este elemento. Puede verse un ejemplo de uso en la figura 3.32.

```
<def-list n="verbos_estado">
  <list-item v="actuar"/>
  <list-item v="buscar"/>
  <list-item v="estudiar"/>
  <list-item v="existir"/>
  <list-item v="ingressar"/>
  <list-item v="introduir"/>
  <list-item v="penetrar"/>
  <list-item v="publicar"/>
  <list-item v="treballar"/>
  <list-item v="viure"/>
</def-list>
```

Figura 3.32: Definición de una lista de lemas en catalán. Estos lemas se utilizan en una regla que puede verse en la figura 3.37.

Elemento de sección de definición de macroinstrucciones `<section-def-macros>`

En esta sección se definen las macroinstrucciones que contienen fragmentos de código utilizado con frecuencia en la parte de acción de las reglas.

Elemento de definición de macroinstrucciones `<def-macro>`

Cada definición de una macroinstrucción tiene un nombre obligatorio (el valor del atributo `n`), el número de argumentos que se le pasan como referencia (atributo `npar`) y un cuerpo con instrucciones.

Elemento de sección de reglas `<section-rules>`

Esta sección contiene las reglas de transferencia estructural, cada una en un elemento <**rule**>.

Elemento de regla `<rule>`

Cada regla tiene un patrón (<**pattern**>) y la acción (<**action**>) asociada a él que se ejecuta cuando es detectado.

Elemento de patrón <pattern>

Un patrón se especifica utilizando componentes de patrón (<**pattern-item**>), cada uno de los cuales corresponde a una forma léxica en el patrón detectado, en orden de aparición.

Elemento de componente de patrón <pattern-item>

En cada componente de un patrón se indica, mediante un atributo de nombre obligatorio `n`, qué tipo de forma léxica se quiere detectar. Para ello se utiliza una de las categorías definidas en <**section-def-cats**> (véase un ejemplo de cómo se detecta el patrón determinante–nombre en la figura 3.39).

Elemento de acción <action>

En este elemento se incluyen las "instrucciones" que se tienen que ejecutar para llevar a cabo el procesamiento que se desee para cada detección de patrones.

La parte de procesamiento del patron detectado es un bloque de cero o más instrucciones de tipo: <**choose**> (procesamiento condicional), <**let**> (asignación de valores), <**out**> (escritura de formas léxicas de la LM), <**modify-case**> (modificación del estado de las mayúsculas y minúsculas de una forma léxica) y <**call-macro**> (llamadas a macroinstrucciones). Durante el procesamiento, según se cumplan o no una serie de opciones condicionales, se realizan distintas operaciones, como por ejemplo la concordancia de los elementos de un patrón ya que éstos están sujetos a posibles cambios de género o número durante el proceso de transferencia léxica. Para ello, a pesar de trabajar con FL en LM también se tiene en cuenta la información de la LO ya que si, por ejemplo, en LO los elementos de un patrón no concuerdan, tal vez tampoco deben hacerlo en LM. Durante la aplicación de las distintas operaciones realizadas dentro un patrón, se asignan valores a atributos del mismo y, en su caso, a variables globales o de estado y se envía la información del patrón en LM resultante al siguiente módulo (el generador morfológico).

Elemento de llamada a macroinstrucción <call-macro>

Dentro de una regla puede invocarse cualquiera de las macroinstrucciones definidas en <**section-def-macros**>. Para ello, hay que especi-

```
<call-macro n="f_concord2">
  <with-param pos="3"/>
  <with-param pos="1"/>
</call-macro>
```

Figura 3.33: Llamada a la macroinstrución `f-concord2` para hacer concordar los elementos de un patrón como por ejemplo determinante–adverbio–nombre. La propagación de género y número se hace a partir de uno de los elementos, en este caso el nombre que aparece como tercer elemento del patrón (3). Por eso, la posición del nombre se pasa como primer parámetro y después se pasan el resto de elementos. Como el adverbio (en posición 2) no necesita información de concordancia, sólo se pasa la posición del determinante (1).

ficar su nombre en el atributo `n` y uno o más argumentos en el elemento de parámetros <**with-param**> (ver más abajo).

Elemento de parámetros <with-param>

Este elemento se utiliza dentro de las llamadas a macroinstrucciones <**call-macro**>. El atributo `pos` de cada argumento se utiliza para referirse a una forma léxica de la regla que invoca la macroinstrucción. Por ejemplo, si se ha definido una macroinstrucción de 2 parámetros que realiza operaciones de concordancia nombre–adjetivo, puede utilizarse con los argumentos 1 y 2 en una regla de nombre–adjetivo, con los argumentos 2 y 3 en una regla de determinante–nombre–adjetivo, con los argumentos 1 y 3 en una regla de nombre–adverbio–adjetivo y con los argumentos 2 y 1 en una regla de adjetivo–nombre. Véase un ejemplo de llamada a macroinstrucción en la figura 3.33.

Elemento de selección <choose>

La instrucción de selección se compone de una o más opciones condicionales (<**when**>) y una opción alternativa <**otherwise**>, de inclusión opcional.

Elemento de condición <when>

Con este elemento se describe una opción condicional (véase la página 71). Se compone de la condición a verificar <**test**> y de un bloque de cero o más instrucciones de tipo <**choose**>, <**let**>, <**out**>, <**modify-case**> y <**call-macro**>, que se ejecutarán si se cumple la citada condición.

Elemento de opción alternativa <otherwise>

El elemento <**otherwise**> contiene un bloque de una o más instrucciones (de tipo <**choose**>, <**let**>, <**out**>, <**modify-case**> y <**call-macro**>) que deben realizarse si no se cumple la condición descrita en ninguno de los <**when**> de un <**choose**>.

Elemento de evaluación <test>

El elemento de evaluación <**test**> dentro de un elemento de condición <**when**> puede estar compuesto por una conjunción (<**and**>), una disyunción (<**or**>) o una negación (<**not**>) de condiciones a evaluar, así como por una simple condición de igualdad (<**equal**>).

Elementos de operadores condicionales o booleanos: <equal>, <and>, <or>, <not>, <in>

- El elemento de conjunción <**and**> representa una condición, compuesta de dos o más condiciones, que se cumple cuando todas las condiciones en él incluidas son ciertas. Se puede ver un ejemplo de su uso en la figura 3.39

- El elemento de disyunción <**or**> representa una condición, compuesta de dos o más condiciones, que se cumple cuando al menos una de ellas es cierta. En la figura 3.36 hay un ejemplo de este tipo de condición para evaluar la concordancia de género de un patrón en LO.

- El elemento de negación <**not**> representa una condición que se cumple cuando la condición incluida no se cumple y viceversa. Se puede ver un ejemplo de negación de una igualdad en la figura 3.36.

- El operador condicional más simple es el de igualdad (<**equal**>). Es una instrucción que comprueba si dos argumentos (dos cadenas) son o no idénticos. Véase un ejemplo del uso de este elemento en las figuras 3.34 y 3.35. En este operador, además, se puede especificar el atributo `caseless`, el cual, si tiene el valor `yes`, hace que la comparación entre cadenas se realice sin tener en cuenta las diferencias de mayúsculas y minúsculas.

- El operador de búsqueda en listas, <**in**>, que sirve para buscar cualquier valor (que corresponde al primer parámetro de la condición en una lista definida en la sección correspondiente (<**section-def-lists**), a la que se hace referencia mediante el atributo `n` del

elemento <**list**>. La búsqueda es cierta si el valor se encuentra en la lista. En esta comparación se puede utilizar también el atributo `caseless`: si se le asigna el valor `yes`, se realiza la búsqueda en la lista sin tener en cuenta las diferencias de mayúsculas y minúsculas. En la figura 3.37 puede verse un ejemplo de su uso.

Elemento <clip>

El elemento <**clip**> representa una subcadena de una forma léxica de la lengua de origen o de la lengua de destino, definida por el valor de varios atributos (véase un ejemplo de uso en la figura 3.34):

- `pos` es un índice (1, 2, 3, etc.) utilizado para seleccionar una forma léxica dentro de la regla: se corresponde con el lugar que ocupa la forma léxica en el patrón.
- `side` especifica si se selecciona un *clip* de la lengua de origen (`sl`, por *source language*) o de la lengua de destino (`tl`, por *target language*).
- en `part` se indica qué parte de la forma léxica se procesa; generalmente su valor es uno de los atributos definidos en <**section-def-attrs**> (`gen`, `nbr`, etc.), aunque también puede tomar cuatro valores predefinidos que son: `lem` (para hacer referencia al lema de la forma léxica), `lemh` (para hacer referencia a la primera parte de un lema partido), `lemq` (la cola del lema partido), y `whole` (la forma léxica completa, con el lema y todos los símbolos gramaticales, con sus valores actuales en el caso de que hayan sido modificados en la parte precedente de la regla).

Elemento de cadena literal <lit>

Este elemento sirve para especificar el valor de una cadena literal mediante el atributo `v`. Por ejemplo, <**lit** `v="andar"/>` representa la cadena *andar*.

Elemento de valor de etiqueta <lit-tag>

Es similar al elemento <**lit**> pero no especifica el valor de una cadena literal sino el de una etiqueta morfológica, mediante el atributo `v`. Véase un ejemplo de su uso en la figura 3.35.

```
<test>
  <not>
    <equal>
      <clip pos="2" side="tl" part="gen"/>
      <clip pos="2" side="sl" part="gen"/>
    </equal>
  </not>
</test>
```

Figura 3.34: Fragmento de una regla en la que se comprueba si el género (`gen`) en LM (`tl`) de la segunda unidad léxica identificada en un patrón es distinto del género de la misma unidad léxica en LO (`sl`)

```
<equal>
  <clip pos="2" side="tl" part="nbr"/>
  <lit-tag v="ND"/>
</equal>
```

Figura 3.35: Uso del elemento <**lit-tag**>: se comprueba si la etiqueta (o símbolo) de número (`nbr`) de la segunda unidad léxica de la LM (`tl`) es `ND` (número por determinar)

```
<test>
  <or>
    <not>
      <equal>
        <clip pos="1" side="sl" part="gen"/>
        <clip pos="3" side="sl" part="gen"/>
      </equal>
    </not>
    <not>
      <equal>
        <clip pos="2" side="sl" part="gen"/>
        <clip pos="3" side="sl" part="gen"/>
      </equal>
    </not>
  </or>
</test>
```

Figura 3.36: Fragmento de una regla en la que se comprueba si el género en LO de la primera o de la segunda unidad léxica identificada en un patrón (podría ser determinante–adjetivo–nombre) es distinto del género de la tercera unidad léxica también en LO.

```
<rule>
  <pattern>
    <pattern-item n="verb"/>
    <pattern-item n="a"/>
  </pattern>
  <action>
  <choose>
    <when>
      <test>
        <in caseless="yes"/>
          <clip pos="1" side="sl" part="lem"/>
          <list n="verbos_estado"/>
        </in>
      </test>
      <let>
        <clip pos="2" side="tl" part="lem"/>
        <lit v="en"/>
      </let>
    </when>
    <!-- ... -->
```

Figura 3.37: Fragmento de una regla que detecta un patrón formado por un verbo más la preposición *a* y a continuación comprueba si el verbo (el lema indicado en `lem`) de la lengua origen (`sl`) es uno de los que se han incluido en la lista de verbos de estado (definida en la figura 3.32). En caso afirmativo, el lema de la segunda palabra de la lengua meta (`tl`) se cambia por *en*.

Elemento de variable <var>

Cada <**var**> es un identificador de variable: el atributo obligatorio `n` indica su nombre tal como se ha definido en <**section-def-vars**>. Cuando se encuentra en un <**out**>, un <**test**>, o la parte derecha de un <**let**>, representa el valor de la variable; cuando se encuentra en la parte izquierda de un <**let**> o en un <**modify-case**>, representa la referencia de la variable y se puede cambiar su valor.

Elemento de referencia a lista de cadenas <list>

Este elemento sólo se usa como segundo parámetro de una búsqueda <**in**>. El atributo `n` hace referencia a la lista concreta definida en la sección de definición de listas de cadenas `<section-def-lists>`. Véase un ejemplo de uso en la figura 3.37.

Elemento de información de mayúsculas/minúsculas <get-case-from>

El elemento <**get-case-from**> representa la cadena resultante de aplicar el esquema de mayúsculas que presenta el lema de una unidad léxica en LO a una cadena (*clip*, *lit* o *var*). Para hacer referencia a la unidad léxica de la que se tomará la información, se utiliza el atributo `pos` que indica la posición de dicha unidad léxica en LO. Se utiliza cuando se reordenan las unidades léxicas de un patrón o cuando se añade o se suprime una unidad léxica. Puede verse un ejemplo de uso en la figura 3.38, que contiene una regla para transformar el pretérito perfecto simple español (*dije*) por el pretérito perfecto perifrástico catalán (*vaig dir*). En esta regla se añade una FL con lema *anar* y símbolo morfológico *vaux* ("verbo auxiliar"), el cual debe tomar la información de mayúsculas del verbo en español (que tiene la posición "1.^{en} el patrón), para que se traduzca *Dije* por *Vaig dir*, *dije* por *vaig dir* y *DIJE* por *VAIG DIR*.

Elemento de obtención del patrón de mayúsculas/minúsculas <case-of>

Sirve para obtener el patrón de mayúsculas/minúsculas, es decir, uno de los valores "`aa`", "`Aa`" o "`AA`", de una cadena, que se obtiene como en el caso de la etiqueta <**clip**>, pues tiene los mismos atributos: `pos`, la posición que ocupa la palabra en el patrón detectado; `side`, el lado de la traducción, `sl` para la LO y `tl` para la LM, y `part`, el atributo concreto al que se hace referencia (normalmente el lema), con los atributos predefinidos que ya se mencionan en la página 73. En la figura 3.38 puede verse un ejemplo de utilización de este elemento, y en el apartado siguiente (la explicación de <**modify-case**>) se encuentra una explicación más detallada del ejemplo.

Elemento de modificación de estado de mayúsculas/minúsculas <modify-case>

Esta instrucción sirve para modificar las mayúsculas o minúsculas de la primera expresión (el primer parámetro, típicamente un lema) mediante un literal o una variable (el segundo parámetro). El primer parámetro puede ser una <**var**>, un <**clip**> o un <**case-of**>, mientras que el segundo puede ser cualquier cosa que devuelva un valor, pero en principio será <**var**> o <**lit**>. Los valores que toma este valor suelen ser "`Aa`' para expresar que la "parte izquierda" de esta modificación de las mayúsculas

debe tener la primera letra en mayúsculas y el resto en minúsculas, "aa" para poner todo en minúsculas y "AA" para poner todo en mayúsculas. En la figura 3.38 puede verse un ejemplo de su utilización. En esta regla se modifica el estado de mayúsculas del lema en LM en posición "1", que corresponde a *dir*, puesto que, aunque a la salida de la regla sea la segunda forma léxica (*vaig dir*), es la traducción de la FL que tiene la posición 1 en la LO, y por lo tanto sigue teniendo asignada esta posición en la LM. A este lema se le asigna el valor "aa" en caso de que el lema en LO tenga el estado "Aa". En los demás casos no hay que especificar nada, pues el estado de mayúsculas/minúsculas en la FL con posición 1 coincidirá en la LO y en la LM y, por lo tanto, se transfiere automáticamente (véase la página 19 para más información sobre la gestión de mayúsculas y minúsculas en los diccionarios).

Elemento de asignación <let>

La instrucción de asignación <**let**> asigna el valor de la parte derecha de la asignación (una cadena literal, un clip, una variable, etc.) a la parte izquierda (un clip, una variable, etc.). Véase un ejemplo de uso en la figura 3.39.

Elemento de salida <out>

En la instrucción de salida se especifican las formas léxicas que se envían a la salida del módulo tras haber sido sometidas a las operaciones de transferencia estructural pertinentes. La instrucción envía cada forma léxica dentro de un conjunto <**lu**>, que a su vez puede estar contenido dentro de un elemento <**mlu**> si lo que se envía es una multipalabra compuesta de dos o más FL. Además, se envían también los espacios en blanco o superblancos (<**b**>) que haya que colocar entre FL y FL.

En las figuras 3.38 y 3.40 puede verse un ejemplo de uso.

Elemento de unidad léxica <lu>

Su nombre proviene de *lexical unit* o "unidad léxica" y es el elemento mediante el que se envía cada forma léxica del patrón en lengua meta a la salida de la regla, dentro del elemento <**out**>. Mediante este elemento puede enviarse la forma léxica completa, con el atributo whole de un <**clip**>, o bien, en caso necesario, indicar explícitamente cada una de sus partes (lema más etiquetas, especificadas mediante cadenas <**clip**>, cade-

```
<rule>
  <pattern>
    <pattern-item n="pretind"/>
  </pattern>
  <action>
    <out>
      <lu>
         <get-case-from pos ="1">
           <lit v="anar"/>
         </get-case-from>
         <lit-tag v="vaux"/>
         <clip pos="1" side="sl" part="persona"/>
         <clip pos="1" side="sl" part="nbr"/>
       </lu>
       <b/>
     </out>
     <choose>
       <when>
         <test>
           <equal>
              <case-of pos="1" side="sl" part="lemh"/>
              <lit v="Aa"/>
           </equal>
         </test>
         <modify-case>
             <case-of pos="1" side="tl" part="lemh"/>
             <lit v="aa"/>
         </modify-case>
       </when>
     </choose>
     <out>
       <lu>
          <clip pos="1" side="tl" part="lemh"/>
          <clip pos="1" side="tl" part="a_verb"/>
          <lit-tag v="inf"/>
          <clip pos="1" side="tl" part="lemq"/>
       </lu>
    </out>
  </action>
</rule>
```

Figura 3.38: Regla para `es-ca` que transforma los verbos en pretérito perfecto simple o indefinido (*dije*) en la forma de pretérito perfecto perifrástico usual en catalán (*vaig dir*), al mismo tiempo que se asigna la información correcta de mayúsculas/minúsculas a las dos palabras resultantes.

```
<rule>
  <pattern>
    <pattern-item n="det"/>
    <pattern-item n="nom"/>
  </pattern>
  <action>
      <choose>
        <when>
          <test>
            <and>
              <not>
                <equal>
                  <clip pos="2" side="tl" part="gen"/>
                  <clip pos="2" side="sl" part="gen"/>
                </equal>
              </not>
              <not>
                <equal>
                  <clip pos="2" side="tl" part="gen"/>
                  <lit-tag v="mf"/>
                </equal>
              </not>
              <not>
                <equal>
                  <clip pos="2" side="tl" part="gen"/>
                  <lit-tag v="GD"/>
                </equal>
              </not>
            </and>
          </test>
          <let>
            <clip pos="1" side="tl" part="gen"/>
            <clip pos="2" side="tl" part="gen"/>
          </let>
        </when>
      </choose>
      <!-- Otras operaciones de concordancia de género y número -->
```

Figura 3.39: Fragmento de una regla en la que el patrón detectado es determinante–nombre (continúa en la fig. 3.40): en esta parte de la regla, se asigna el género del nombre al determinante en caso de que el nombre cambie de género entre la LO (`sl`) y la LM (`tl`) durante el proceso de transferencia léxica entre ambas lenguas.

```
    <!-- ... -->
    <out>
      <lu>
        <clip pos="1" side="tl" part="whole"/>
      </lu>
      <lu>
        <clip pos="2" side="tl" part="whole"/>
      </lu>
    </out>
  </process>
 </action>
</rule>
```

Figura 3.40: Fragmento de una regla (viene de la fig. 3.39). Al final de la regla, y tras varias operaciones, se da salida a la información resultante mediante el atributo `whole`, que contiene el lema y los símbolos morfológicos de cada FL (posiciones 1 y 2 del patrón).

nas literales <**lit**>, etiquetas <**lit-tag**>, variables <**var**>, así como información de mayúsculas y minúsculas [<**get-case-from**>, <**case-of**>]).

Hay que tener en cuenta que, como se ha explicado antes, en el caso de las multipalabras con *lema partido* hay que resituar la cola de un lema multipalabra detrás de los símbolos gramaticales de la palabra flexionada (o cabeza del lema), ya que el módulo `pretransfer` ha adelantado la cola y la ha colocado antes de los símbolos gramaticales de la cabeza. Esta recolocación se realiza aquí, dentro del elemento <**lu**>, mediante los valores `lemh` y `lemq` del atributo `part` de un <**clip**>. El atributo `lemh` corresponde a la cabeza de lema, y el atributo `lemq`, a la cola del lema. Como se ve en el ejemplo 3.38, la parte `lemq` del <**clip**> se coloca detrás de la cabeza del lema y de los símbolos gramaticales que la acompañan. Esta regla serviría, por ejemplo, para la forma en español *eché de menos*, que debe traducirse al catalán por *vaig trobar a faltar*. El atributo `a_verb` que aparece detrás de `lemh` contiene el símbolo gramatical que describe la categoría del verbo (*vblex*, *vbser*, *vbhaver* o *vbmod* según el caso). Así, la última forma léxica enviada por esta regla, para el caso de *vaig trobar a faltar*, sería en el flujo de datos:

```
^trobar<vblex><inf># a faltar$
```

El símbolo almohadilla del flujo de datos se corresponde con el elemento <**g**> de los diccionarios, utilizado para indicar la posición de las partes invariables de una unidad multipalabra con lema partido.

Es importante tener en cuenta que los atributos que se incluyen dentro de <**lu**> pueden estar vacíos. Así, un verbo que entre por la regla de la figura 3.38 y que no sea una multipalabra con lema partido, será enviado con el atributo `lemq` vacío, puesto que no tiene cola de lema. De este modo no hay que definir reglas diferentes para formas léxicas con cola y sin cola. Puede verse también otro ejemplo en la página 120, en que la regla para verbo envía con <**lu**> los atributos `gen` (*género*) y `nbr` (*número*). De este modo se incluyen los participios (con género y número) y las demás formas verbales (que llevarán estos atributos vacíos).

En la misma página puede verse una regla para verbo seguido de pronombre enclítico. Aquí, la cola del lema se coloca detrás del pronombre enclítico; así, las formas léxicas enviadas en el caso de una multipalabra unida a un pronombre enclítico, como *echándote de menos*, serían, en el flujo de datos y traduciendo al catalán:

```
^trobar<vblex><ger>+et<prn><enc><p2><mf><sg># a faltar$
```

Por supuesto, esta regla sirve también para verbos que no sigan este patrón de multipalabra, de modo que la forma *explicándote* sería así enviada por la regla en la traducción de español a catalán:

```
^explicar<vblex><ger>+et<prn><enc><p2><mf><sg>$
```

En cuanto al atributo `whole` de un <**clip**>, es importante tener en cuenta que sólo puede utilizarse para enviar la forma léxica completa en caso de que la palabra enviada no pueda ser una multipalabra, es decir, no pueda contener un lema partido. Compárense las figuras 3.38 y 3.40. El atributo `whole` puede utilizarse en el segundo ejemplo porque contiene el lema `lem` más todas las etiquetas morfológicas de las formas léxicas en posición 1 y 2 (determinante y nombre). En cambio, en el primer ejemplo, la forma léxica enviada dentro de <**lu**> se envía por partes, con un `lemh` (cabeza del lema) y un `lemq` (cola del lema), puesto que puede ser que el verbo detectado en el patrón sea una multipalabra con lema partido. En la práctica, esto significa que, en nuestro sistema, el atributo `whole` puede utilizarse para enviar cualquier tipo de forma léxica excepto los verbos, puesto que sólo hemos definido multipalabras con flexión para formas verbales.

Elemento de unidad léxica <mlu>

Su nombre proviene de *multilexical unit* o "multiforma léxica" y se utiliza dentro del elemento <**out**> para enviar multipalabras compuestas por

más de una forma léxica. Cada forma léxica de una <**mlu**> se envía dentro de un elemento <**lu**>. A la salida del módulo, las formas léxicas contenidas dentro de este elemento aparecerán unidas entre sí mediante el símbolo '+' del flujo de datos. Esto significa que se convertirán en una multipalabra compuesta por varias formas léxicas y que serán tratadas como una unidad por los módulos subsiguientes; por lo tanto, en el diccionario de generación deberá existir una entrada para esta multipalabra para que sea posible generarla.

En nuestro sistema, este elemento se utiliza para unir los pronombres enclíticos a los verbos conjugados.

Elemento de blanco <b>

El elemento <**b**> hace referencia a los [super]blancos, y se indexa mediante el atributo `pos`; por ejemplo, un <**b**> con `pos="2"` se refiere a los [super]blancos (incluidos los datos de formato encapsulados por el desformatador) entre la 2ª FLLO y la 3ª FLLO. La gestión explícita de [super]blancos permite la colocación correcta del formato cuando el resultado de la transferencia estructural tiene más o menos elementos léxicos que el original o bien ha sido sometido a algún tipo de reordenación.

3.4.3. Preproceso del módulo de transferencia estructural

Los ficheros de especificación de los módulos de transferencia estructural, también llamados *ficheros de reglas de transferencia*, son preprocesados por el programa *apertium-preprocess-transfer* que calcula los patrones para casar precondiciones de las reglas y además indexa las reglas para acelerar su procesamiento en tiempo de ejecución. Esta información se almacena en un archivo en formato binario que se lee junto con el diccionario bilingüe correspondiente y el propio fichero de reglas de transferencia para la ejecución conjunta de los módulos de transferencia léxica y estructural.

3.5. Desformateador y reformateador

3.5.1. Tratamiento del formato de los documentos

En esta sección se describe cómo tratan el formato de los documentos los desformateadores y reformateadores de este sistema, creados a partir de las reglas de especificación de formato en XML que se describen más adelante, en la sección 3.5.2.

Los requisitos del proyecto indican que el traductor debe ser capaz de procesar documentos en formato XML, HTML, RTF y texto llano. En todos estos tipos de documento es posible realizar una *encapsulación* del formato, tal como se describe a continuación.

El procedimiento consiste en encapsular las cadenas de texto —en adelante *bloques de formato* o *superblancos*— que se identifiquen como parte del formato entre delimitadores que dependerán de la especificación del flujo de datos entre módulos (que se discute en detalle en el capítulo 2); así, en el formato de flujo (secciones 2.2.1 y 2.3), los *superblancos* van entre corchetes `'['` y `']'`. Cada una de estas cadenas encapsuladas será tratada como si de un blanco <**b**/> (página 35) se tratara —de ahí lo de *superblancos*— y serán restituidas en el orden correcto a la salida del traductor.

Como ya se ha indicado en el capítulo 2, cuando los bloques de formato sean grandes (como ocurre a veces en HTML con segmentos de código en Javascript, o en RTF con imágenes en formato de mapa de bits), se utilizará un mecanismo de almacenamiento en ficheros temporales de estos bloques de formato grandes para eliminarlos del flujo de datos de la traducción.

Un requisito adicional que aparece es que muchas veces el formato de una parte del documento marca implícitamente la división del texto en oraciones (véase la página 13 del capítulo 2). Por ejemplo, los títulos de secciones o de documentos pueden estar en frases que no acaben en punto. Si sabemos que una marca de formato nos está indicando esta división debemos aprovechar esta información para poder traducir mejor estos casos. Ejemplos de este fenómeno pueden ser: dos saltos de línea seguidos en el formato de texto llano, una etiqueta `</h1>` en HTML, etcétera. El módulo desformateador debe generar en estos casos una marca de final de oración equivalente a un punto.

Sistema de encapsulación del formato y ejemplo

Los tipos de bloques de formato o *superblancos* que se generan como resultado del procesamiento del formato se presentan a continuación:

- *Bloques de formato* o *superblancos no vacíos*. Son los que contienen exclusivamente marcas de formato del documento original. En el flujo de datos descrito en el capítulo 2 se abren con un corchete izquierdo `'['` y se cierran con el derecho `']'`.

- *Bloques de formato con referencia a archivo externo* o *superblancos extensos*. Sirven para encapsular segmentos de formato largos y así mejorar el rendimiento del traductor. En el flujo de datos descrito en el

capítulo 2 comienzan por la cadena '`[@`', siguen con el nombre del archivo en el que se almacena el segmento de formato extraído del documento original, y finalmente acaban en un corchete derecho '`]`'.

- *Bloques de formato vacíos*. Sirven para incorporar información artificial de segmentación del texto obtenida a partir del formato. Antes del bloque de formato vacío se coloca el signo de puntuación artificial que se tenga que incluir. Cuando sea necesario restituir el formato original al documento, la presencia de un bloque de formato de estas características obligará a la supresión del carácter inmediatamente anterior que se encuentre en el flujo de datos.

El criterio que se aplica para crear estos bloques de formato se basa en los siguientes criterios generales:

- Se debe encapsular todo lo que no se considere parte del texto que hay que traducir para formar bloques de formato.
- No debe haber dos o más bloques de formato no vacíos estrictamente consecutivos. Dos bloques de formato consecutivos siempre deben unirse en un solo bloque.
- Los bloques de formato vacíos deben preceder a un bloque de formato no vacío o al fin de fichero.

En la figura 3.41 se muestra un documento de ejemplo cuyo formato debe ser tratado previamente a la traducción; en el ejemplo, el encapsulamiento se corresponde con el formato de flujo no basado en XML. La figura 3.42 muestra el resultado de procesar el anterior documento.

De este ejemplo conviene resaltar los siguientes fenómenos:

- No se generan bloques de formato con contenido (no vacíos) consecutivos.
- Las etiquetas como </**title**> y </**p**> inducen la introducción de signos de puntuación artificiales, y esta introducción se realiza de manera sistemática (incluso cuando no hace falta, porque no molesta y es eficiente).
- Los superblancos extensos salen literalmente del proceso de la traducción. En este caso, el fichero temporal `temp35345` contiene las etiquetas desde </**title**> hasta <**p**>

```
<html>
<head>
<title>Esto es el título</title>
<script>
<!-- ... (un bloque de código extenso) -->
</script>
</head>
<body>
<p>Esto
es un párrafo escrito en dos líneas</p>
</body>
</html>
```

Figura 3.41: Ejemplo de documento HTML

```
[<html>
<head>
<title>]Esto es el título.[][@/tmp/temp35345]Esto[
]es un párrafo escrito en dos líneas.[][</p>
</body>
</html>]
```

Figura 3.42: Ejemplo de documento HTML con los bloques de formato encapsulados por el desformateador

- Los blancos sencillos entre palabras no se encapsulan. No ocurre lo mismo con los blancos múltiples (dos o más blancos consecutivos), tabuladores, etc. que son encapsulados. Los saltos de línea también se encapsulan.

3.5.2. Datos: reglas de especificación de formato

En esta sección se describe el proceso de generación de formateadores y desformateadores a partir de una especificación de formato en XML.

Las reglas para los formatos se especifican, como los datos lingüísticos, en XML, y dentro de ellas se declaran expresiones regulares con sintaxis de `flex`. La especificación se estructura en tres partes (ver DTD de la definición formal en el apéndice A.4):

- **Opciones de configuración**. En este apartado se especifica el valor de la longitud máxima de un superblanco no extenso, las codificaciones de entrada y de salida, la necesidad de la distinción o no entre mayúsculas y minúsculas y las expresiones regulares para los caracteres de escape y para los caracteres de espaciado.

- **Reglas de formato**. Se describe el conjunto de etiquetas propias de un formato determinado que han de ser encerradas en un bloque de formato por el desformateador. Dichas etiquetas de formato podrán, opcionalmente, indicar el fin de frase para que el desformateador inserte el signo de puntuación artificial (seguido de un bloque de formato vacío, como se especifica en la sección anterior). También se ha de especificar la prioridad de aplicación de las reglas aunque, en caso de no ser necesario, se puede dar a todas reglas la misma prioridad mediante la asignación del mismo valor (cualquier número) a todas ellas.

  Se entiende que todo lo que no sea especificado como formato quedará sin encapsular y será texto para traducir.

- **Reglas de sustitución**. Permiten sustituir caracteres especiales del texto. Una expresión regular recogerá un conjunto de caracteres especiales, y los sustituirá por los caracteres que se especifique. Por ejemplo, en HTML los caracteres especificados en hexadecimal deben ser sustituidos por su correspondiente entidad o carácter ASCII. Por ejemplo, `camión` se corresponde con `camión`.

A continuación se describen las reglas con más detalle.

- Raiz del documento de especificación. En el atributo `name` está el nombre del formato.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<format name="html">
  <options>
  ...
  </options>

  <rules>
  ...
  </rules>
</format>
```

Dentro deben estar las opciones y las reglas de las que se presenta un ejemplo a continuación:

- Opciones.

```
<options>
  <largeblocks size="8192"/>
  <input encoding="ISO-8859-1"/>
  <output encoding="ISO-8859-1"/>
  <escape-chars regexp='[\[\]^$\\]'/>
  <space-chars regexp='[ \n\t\r]'/>
  <case-sensitive value="no"/>
</options>
```

El elemento `<largeblocks>` sirve para especificar la longitud máxima de un superblanco no extenso, mediante el valor del atributo `size`. Los elementos `<input>` y `<output>` sirven para especificar la codificación de entrada y de salida del texto mediante el atributo `encoding`.

El elemento `escape-chars` permite especificar, mediante una expresión regular que se da en el valor del atributo `regexp`, qué caracteres deben protegerse mediante una barra invertida. El elemento `<space-chars>` especifica el conjunto de caracteres que tienen que ser tomados como si fueran espacios en blanco.

Finalmente, el elemento `case-sensitive` sirve para especificar si, en todas las especificaciones de atributos del formato en las que intervengan expresiones regulares es relevante si se especifican mayúsculas o minúsculas.

- Reglas. Hay reglas de formato y de sustitución.

```
<rules>
  <format-rule ... >
    ...
  </format-rule>
  ...

  <replacement-rule>
    ...
  </replacement-rule>
  ...
</rules>
```

Se detallan en los dos puntos siguientes.

- Reglas de formato. El desformateador encerrará en bloques de formato las etiquetas que estas reglas indiquen en su campo `regexp`. Si se trata de etiquetas de inicio y fin, y todo lo que encierran es formato, se especifica una `regexp` tanto para `begin` como para `end`:

  ```
  <format-rule eos="no" priority="1">
    <begin regexp='"\&lt;!--"'/>
    <end regexp='"--\&gt;"'/>
  </format-rule>
  ```

  De lo contrario se utiliza sólo un elemento `begin-end`:

  ```
  <format-rule eos="yes" priority="3">
    <begin-end regexp='"&lt;"[/]?"li"[^&gt;]*"&gt;"'/>
  </format-rule>
  ```

  Además, en `priority` se especifica una prioridad que sirve para saber en qué orden se aplican las reglas de formato (el valor absoluto no es relevante, sólo se utiliza el orden que se establece entre los valores usados en las reglas). En "`eos`" se indica mediante `yes` o `no` si el bloque de formato del que forme parte el patrón detectado deberá ir precedido de un signo de puntuación artificial o no.[9]

- Reglas de sustitución. Sirve para sustituir caracteres especiales del texto. La expresión regular del atributo `regexp` recogerá un conjunto de caracteres especiales y los sustituirá en el texto a traducir por los caracteres que se especifiquen. La correspondencia entre caracteres originales y sustituidos se establece en los atributos `source` y `target` de los elementos `replace`, que pueden ser múltiples:

[9] En todos estos casos, se usan las típicas entidades `<` y `>` para representar los caracteres < y > respectivamente.

```
<replacement-rule regexp='"&amp;"[^;]+;'>
  <replace source="&amp;Agrave;" target="À"/>
  <replace source="&amp;#192;" target="À"/>
  <replace source="&amp;#xC0;" target="À"/>
  <replace source="&amp;#xc0;" target="À"/>
  <replace source="&amp;Aacute;" target="Á"/>
  <replace source="&amp;#193;" target="Á"/>
  <replace source="&amp;#xC1;" target="Á"/>
  <replace source="&amp;#xc1;" target="Á"/>
  ...
</replacement-rule>
```

- Expresiones regulares de los atributos `regexp`. Tienen la sintaxis utilizada en `flex` [9].

Como ejemplo de especificación de formato vamos a utilizar el de HTML, del que se puede seguir la explicación que ofrecemos a continuación mirando la figura 3.43.

Encontramos, en primer lugar, la regla de formato que especifica de forma general todas las etiquetas de HTML, es decir, considera etiqueta HTML todo aquello que empiece por el signo **<** y termine por el signo **>**. Esta regla tiene la prioridad más baja (4) para permitir que las reglas más específicas se apliquen preferentemente. Pero antes de considerar una etiqueta como general, es decir, antes de aplicar esta regla, se aplicará alguna de las reglas de mayor prioridad. En el caso de HTML, la mayor prioridad la tienen los comentarios `<!-- ... -->`. Las marcas de inicio y fin `<script></script>` y `<style></style>`, donde se considera formato todo que lo éstas encierren, tienen prioridad 2. Prioridad 3 tienen las etiquetas que implican fin de frase (puntuación artificial), que son `</br>`, `</hr>`, `</p>`, etc.

Finalmente se especifican las reglas de sustitución, para sustituir todos los códigos que empiecen por `&`, y así se especifica en la expresión regular. Después se define cada una de las sustituciones: tanto `&Agrave`, como `&#192`, `&#xC0` y `&#xc0` se sustituyen por `À`. De igual manera se declaran los demás caracteres especiales.

3.5.3. Generación de formateadores y desformateadores

Para generar el formateador y el desformateador de un formato dado, se aplica a las reglas en XML que declaran este formato (que se describen a continuación) una hoja de estilo generadora de formateadores y desformateadores Esta transformación XSLT da como resultado un fichero `lex`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<format name="html">
  <options>
    <largeblocks size="8192"/>
    <input encoding="ISO-8859-1"/>
    <output encoding="ISO-8859-1"/>
    <escape-chars regexp='[\[\]^$\\]'/>
    <space-chars regexp='[ \ n\ t\ r]'/>
    <case-sensitive value="no"/>
  </options>

  <rules>
   <format-rule eos="no" priority="1">
      <begin regexp='"&lt;!--"'/>
     <end regexp='"--&gt;"'/>
   </format-rule>

   <format-rule eos="no" priority="2">
     <begin regexp='"&lt;script"[^&gt;]*"&gt;"'/>
     <end regexp='"&lt;/script"[^&gt;]*"&gt;"'/>
   </format-rule>
   <format-rule eos="no" priority="2">
     <begin regexp='"&lt;style"[^&gt;]*"&gt;"'/>
     <end regexp='"&lt;/style"[^&gt;]*"&gt;"'/>
   </format-rule>

   <format-rule eos="yes" priority="3">
     <begin-end regexp='"&lt;"[/]?"br"[^&gt;]*"&gt;"'/>
   </format-rule>
   <!-- Aquí faltan más declaraciones format-rule eos="yes"-->
   <!-- ...                                                 -->

   <format-rule eos="no" priority="4">
     <begin-end regexp='"&lt;"[a-zA-Z][^&gt;]*"&gt;"'/>
   </format-rule>

   <replacement-rule regexp='"&amp;"[^;]+;'>
     <replace source="&amp;Agrave;" target="À"/>
     <replace source="&amp;#192;" target="À"/>
     <replace source="&amp;#xC0;" target="À"/>
     <replace source="&amp;#xc0;" target="À"/>
     <!-- Aquí faltan más elementos replace              -->
     <!-- ...                                            -->
   </replacement-rule>
  </rules>
</format>
```

Figura 3.43: Parte de la definición de reglas del formato HTML

[9] que, una vez compilado, es el ejecutable del formateador o desformateador para el formato especificado.

Gracias a la especificación general de formatos descrita en este capítulo, se han podido definir las especificaciones para los formatos HTML, RTF y texto llano. Estas especificaciones se hallan en el paquete `apertium`, en los ficheros respectivos `html-format.xml`, `rtf-format.xml`, `txt-format.xml`. En particular, es relativamente sencillo definir desformateadores y reformateadores de cualquier formato XML.

Capítulo 4

Instalación y ejecución del sistema

4.1. Requisitos del sistema

El sistema en el que se desee instalar y ejecutar Apertium debe tener los siguientes programas instalados:

- `libxml2` versión 2.6.17 o superior
- programa `xmllint` (generalmente viene incluido en `libxml2`, pero puede ser un paquete independiente en el sistema, por ejemplo en Debian GNU-Linux)
- programa `xsltproc` (para no usuarios de PowerPC); también viene con `libxml2` pero puede ser igualmente un paquete independiente en el sistema, como en el caso de `xmllint`
- programa `sabcmd` (usuarios de PowerPC), suministrado con el paquete `sablotron`
- GNU `make`, `gcc` (`g++`), intérprete de comandos `bash`

4.2. Instalación de los paquetes de programas

Para instalar los programas y librerías del sistema de traducción automática Apertium, deberá descargarse (de `http://sourceforge.net/projects/apertium`), compilar e instalar la última versión de los siguientes paquetes, en el orden especificado:

1. `lttoolbox`
2. `apertium`

La manera más sencilla de compilar los paquetes es la siguiente:

1. Vaya al directorio que contiene el código fuente del paquete y escriba `./configure`; se configurará el paquete en su sistema. Si utiliza csh en una versión antigua de System V, probablemente deberá escribir `sh ./configure` para evitar que `csh` (el intérprete de comandos por defecto en el antiguo System V) intente ejecutar `configure` por sí solo. La ejecución de `configure` tarda un tiempo; mientras se ejecuta, muestra en pantalla algunos mensajes que indican qué comprobaciones está realizando.
2. Escriba `make` para compilar el paquete
3. Escriba `make install` (probablemente necesitará privilegios de administrador) para instalar los programas, los ficheros de datos y la documentación.
4. Si desea eliminar del directorio de código fuente los archivos binarios y los archivos de objetos del programa, escriba `make clean`. Para eliminar también los ficheros que `configure` ha creado (de modo que pueda compilar el paquete para un ordenador de diferente tipo), escriba `make distclean`. También hay una opcion `maintainer-clean` en el Makefile, pero está prevista principalmente para los desarrolladores del paquete. Si lo utiliza, es probable que después tenga que buscar varios programas que necesitará para volver a generar los archivos que estaban incluidos en la distribución.

Si no dispone de privilegios de administrador para instalar los programas en su sistema, puede utilizar la opción `-prefix` con el script de configuración para que se instalen en su cuenta de usuario. Por ejemplo:

```
$ pwd
/home/me/lttoolbox-0.9.1
$ ./configure --prefix=/home/me/myinstall
```

Las librerías se instalarán en el directorio `LIBDIR=$prefix/lib`. Si no se especifica ninguna opción `-prefix` con el script de configuración, el directorio LIBDIR será `/usr/local/lib`.

Si encuentra algún error de enlace con las librerías instaladas en el directorio `LIBDIR`, deberá usar libtool y especificar la ruta de acceso completa de la librería, o bien utilizar la opción `LIBDIR` durante el enlace y realizar al menos una de las siguientes acciones:

- añadir `LIBDIR` a la variable de entorno `LD_LIBRARY_PATH` durante la ejecución
- añadir `LIBDIR` a la variable de entorno `LD_RUN_PATH` durante el enlace
- utilizar `-Wl,--rpath -Wl,` opción de enlazado `LIBDIR`
- pedir al administrador del sistema que añada `LIBDIR` a `/etc/ld.so.conf` y ejecute `ldconfig`

Consulte la documentación del sistema operativo relativa a las librerías compartidas para obtener más información; por ejemplo, las páginas de manual `ld(1)` y `ld.so(8)`.

4.3. Instalación de los paquetes de datos

Para instalar los paquetes de datos lingüísticos, siga los pasos siguientes:

1. Descargue un paquete de datos (`apertium-`L_1`-`L_2`-VERSION.tar.gz`) de la página web de Apertium en Sourceforge (`http://apertium.sourceforge.net/`). Por ejemplo, para obtener la versión 0.9 de los datos lingüísticos del traductor español–catalán, deberá descargarse el paquete `apertium-es-ca-0.9.tar.gz`.
2. Descomprima el archivo en cualquier directorio, vaya al directorio en cuestión y escriba `make` en el terminal. Espere mientras se compilan los datos lingüísticos.

4.4. Ejecución del traductor

Hay versiones del traductor Apertium que funcionan tanto en sistemas operativos Linux (siempre más actualizadas) como en Windows. La información de este capítulo está básicamente dirigida a los usuarios de Linux.

Para utilizar el traductor, debe ejecutarse la herramienta `apertium-translator` haciendo referencia al directorio en el que se encuentran los datos lingüísticos, indicando además la dirección de traducción (`es-ca`, `ca-es`, `es-gl`, etc.), el formato del fichero (`txt`, `html`, `rtf`), el nombre del fichero a traducir y el nombre del fichero resultante. La estructura del comando es por tanto la siguiente:

```
$ apertium-translator <directorio> <traducción> <formato>

                < fichero_original > fichero_traducido
```

Por ejemplo, si el directorio es `/home/maria/apertium-es-ca`, hay que escribir lo siguiente para traducir un fichero en formato `txt` de español a catalán:

```
$ apertium-translator /home/maria/apertium-es-ca es-ca
txt <fichero_esp >fichero_cat
```

Es aconsejable situarse en el directorio en el que están almacenados los datos lingüísticos, porque de este modo es suficiente con escribir un punto para hacer referencia al directorio presente:

```
$ apertium-translator . es-ca txt <fichero_esp >fichero_cat
```

Si no se indica ningún formato, el formato por defecto es `txt`. Con los formatos `txt`, `html` y `rtf`, las palabras desconocidas se marcan con un asterisco (*); si se desea que no se marquen las desconocidas, se añade una `u` al nombre del formato. Así, las opciones de formato son las siguientes:

- `txt` : Opción predeterminada, texto con marca de desconocidas
- `txtu` : texto sin marca de desconocidas
- `html` : HTML con marca de desconocidas
- `htmlu` : HTML sin marca de desconocidas
- `rtf` : RTF con marca de desconocidas
- `rtfu` : RTF sin marca de desconocidas

Si no se desea traducir un fichero sino traducir directamente una frase en pantalla, puede ejecutarse la herramienta `apertium-translator` sin indicar ningún nombre de archivo. El comando, si estamos situados en el directorio de datos lingüísticos, sería como sigue:

```
$ apertium-translator . es-ca
```

A continuación, debe escribirse el texto que se desea traducir (pueden incluirse saltos de línea). Para obtener la versión traducida, debe pulsarse Ctrl + D, con lo que la traducción aparecerá en pantalla.

Una tercera manera de traducir con Apertium es utilizando el comando `echo` para pasar texto a través del traductor:

```
$ echo "texto que quiero traducir" | apertium-translator . es-ca
```

Capítulo 5

Mantenimiento de los datos lingüísticos del traductor

5.1. Descripción de los datos lingüísticos existentes

Actualmente, Apertium posee datos lingüísticos para dos pares de lenguas: español–catalán y español–gallego. Los archivos que contienen los datos lingüísticos están contenidos en un solo directorio: `apertium-es-ca` para el par español–catalán y `apertium-es-gl` para el par español–gallego. Los nombres de los ficheros contenidos en este directorio siguen la siguiente estructura:

- `apertium-`L_1`-`L_2`.`L_1`.dix` : diccionario monolingüe del idioma L_1.
- `apertium-`L_1`-`L_2`.`L_2`.dix` : diccionario monolingüe del idioma L_2.
- `apertium-`L_1`-`L_2`.`L_1`-`L_2`.dix` : diccionario bilingüe del par de idiomas L_1–L_2 (en ambas direcciones).
- `apertium-`L_1`-`L_2`.trules-`L_1`-`L_2`.dix` : reglas de transferencia estructural para la traducción de L_1 a L_2.
- `apertium-`L_1`-`L_2`.trules-`L_2`-`L_2`.dix` : reglas de transferencia estructural para la traducción de L_2 a L_1.
- `apertium-`L_1`-`L_2`.`L_1`.tsx` : fichero de definición del desambiguador para L_1.
- `apertium-`L_1`-`L_2`.`L_2`.tsx` : fichero de definición del desambiguador para L_2.
- `apertium-`L_1`-`L_2`.post-`L_1`.dix` : diccionario de postgeneración para L_1 (se aplica al traducir hacia la lengua L_1).

- `apertium-`L_1`-`L_2`.post-`L_2`.dix` : diccionario de postgeneración para L_2 (se aplica al traducir hacia la lengua L_2).
- directorio L_1`-tagger-data` : contiene datos necesarios para el tagger de L_1 (corpus, etc.)
- directorio L_2`-tagger-data` : contiene datos necesarios para el tagger de L_2 (corpus, etc.)

`apertium-`L_1`-`L_2 hace referencia a la combinación lingüística del traductor. Sus dos valores posibles actualmente son `apertium-es-ca` y `apertium-es-gl`. De acuerdo con esta estructura, el fichero monolingüe de catalán se llama `apertium-es-ca.ca.dix`, el fichero bilingüe español–gallego se llama `apertium-es-gl.es-gl.dix` y el fichero de reglas de transferencia estructural para la traducción de catalán a español se llama `apertium-es-ca.trules-ca-es.xml`.

Los datos lingüísticos de que se dispone (datos de enero del 2006) para los distintos pares de lenguas se resumen en el siguiente cuadro.

Traductor Apertium-es-ca	
Diccionario monolingüe de español	11.800 entradas
Diccionario monolingüe de catalán	11.800 entradas
Diccionario bilingüe español–catalán	12.800 entradas (correspondencias `es-ca`)
Reglas de transferencia estructural de español a catalán	44 reglas
Reglas de transferencia estructural de catalán a español	58 reglas
Diccionario de postgeneración para el español	25 entradas y 5 paradigmas
Diccionario de postgeneración para el catalán	16 entradas y 57 paradigmas
Traductor Apertium-es-gl	
Diccionario monolingüe de español	9.000 entradas
Diccionario monolingüe de gallego	8.600 entradas
Diccionario bilingüe español–gallego	8.500 entradas (correspondencias `es-gl`)
Reglas de transferencia estructural de español a gallego	46 reglas
Reglas de transferencia estructural de gallego a español	38 reglas
Diccionario de postgeneración para el español	36 entradas y 12 paradigmas
Diccionario de postgeneración para el gallego	74 entradas y 48 paradigmas

5.2. Introducir palabras en los diccionarios monolingües y bilingües

La adición de léxico a los diccionarios es seguramente la operación que con más probabilidad realizarán quienes quieran ampliar o adaptar Apertium, más que añadir reglas de transferencia o de postgeneración.

A continuación se explican los aspectos más importantes que hay que tener en cuenta a la hora de introducir palabras. Esta información es más general que la ofrecida en el apartado de descripción de los diccionarios (punto 3.1.2), aunque contiene información práctica que puede ser de gran utilidad para quienes decidan llevar a cabo esta tarea.

Para añadir una palabra a Apertium hay que realizar tres entradas en los diccionarios. Supongamos que está trabajando con el par español–catalán. En este caso, deberá añadir:

1. una entrada en el diccionario monolingüe español: para que el traductor pueda analizar (.[en]tender") la palabra cuando la encuentre en un texto, y la pueda generar al traducir esta palabra al español.

2. una entrada en el diccionario bilingüe: para que Apertium pueda traducir esta palabra de una lengua a la otra.

3. una entrada en el diccionario monolingüe de catalán: para que el traductor pueda analizar (.[en]tender") la palabra cuando lo encuentre en un texto, y la pueda generar al traducir esta palabra al catalán.

Deberá ir al directorio que contiene los diccionarios en XML (para el par español–catalán, por ejemplo, es `apertium-es-ca`) y abrir con un editor de texto o un editor de XML especializado los tres diccionarios mencionados: `apertium-es-ca.es.dix`, `apertium-es-ca.es-ca.dix` y `apertium-es-ca.ca.dix`. Las entradas que deberá añadir en estos tres diccionarios tienen una estructura similar.

Diccionario monolingüe (español)

Supongamos que desea añadir el adjetivo español *cósmico*, cuyo equivalente en catalán es *còsmic*. El primer paso es añadir esta palabra al diccionario monolingüe español.

Verá que un diccionario monolingüe tiene principalmente dos tipos de datos: **paradigmas** (en la sección "`<pardefs>`"del diccionario, cada uno en un elemento `<pardef>`) y **entradas léxicas** (en la sección principal del

diccionario (`<section>`), cada una dentro de un elemento `<e>`. Las entradas léxicas se componen de un lema (es decir, la palabra que encontraríamos como entrada en un diccionario en papel) y de información gramatical; los paradigmas contienen los datos de flexión de todos los lemas del diccionario. Para buscar una palabra concreta, puede buscar la cadena `lm="palabra"` (`lm` de *lema*). Tenga en cuenta, sin embargo, que el elemento `lm` es opcional en los diccionarios y que es posible que algunos no incluyan esta información.

Observe cómo son las entradas léxicas del diccionario monolingüe español, por ejemplo la entrada para el adjetivo *bonito*. Puede encontrarla buscando la cadena `lm="bonito"`:

```
<e lm="bonito">
    <i>bonit</i>
  <par n="absolut/o__n"/>
</e>
```

Para añadir una palabra, deberá crear una entrada con la misma estructura. La parte entre `<i>` e `</i>` contiene la parte inicial de la palabra que es común a todas las formas flexionadas, y el elemento `<par>` indica cuál es su paradigma de flexión. Por lo tanto, esta entrada significa que el adjetivo *bonito* se flexiona igual que el adjetivo *absoluto* con el mismo análisis morfológico: las formas *bonit**o***, *bonit**a***, *bonit**os***, *bonit**as*** son equivalentes a las formas *absolut**o***, *absolut**a***, *absolut**os***, *absolut**as*** y tienen el mismo análisis morfológico: `adj m sg`, `adj f sg`, `adj m pl` y `adj f pl` respectivamente.

Primero debe decidir cuál es la categoria léxica de la palabra que desea añadir: la palabra *cósmico* es un adjetivo, como *bonito*. A continuación, debe encontrar el paradigma apropiado para este adjetivo. ¿Es el mismo que el de *bonito* o *absoluto*? ¿Puede decir *cósmic**o***, *cósmic**a***, *cósmic**os***, *cósmic**as***? Como la respuesta es afirmativa, ya tiene la información necesaria para crear la entrada:

```
<e lm="cósmico">
    <i>cósmic</i>
  <par n="absolut/o__adj"/>
</e>
```

Si la palabra que desea añadir tiene un paradigma diferente, deberá buscarlo en el diccionario y asignarlo a la entrada. Hay dos maneras de buscar un paradigma: mirando en la sección `<pardefs>` del diccionario, donde están definidos todos los paradigmas, cada uno dentro de un elemento

<pardef>, o bien buscando una palabra que pensemos que pueda existir en el diccionario y que tenga el mismo paradigma de inflexión que la que queremos añadir. Por ejemplo, si desea añadir la palabra *genoma*, deberá encontrar un paradigma para un **nombre** de género masculino que forme el plural añadiendo una **-s**. En los diccionarios actuales, se trata del paradigma "abismo__n". Por lo tanto, la entrada para esta nueva palabra sería:

```
<e lm="genoma">
    <i>genoma</i>
  <par n="abismo__n"/>
</e>
```

En casos excepcionales será necesario crear un nuevo paradigma para una palabra determinada. Para ello, observe la estructura de los demás paradigma y cree uno nuevo a partir de alguno que sea similar. Para obtener una descripción más detallada de los paradigmas y de las entradas léxicas de los diccionarios, consulte la sección 3.1.2.

Diccionario monolingüe (catalán)

Una vez añadida la palabra a un diccionario monolingüe, hay que hacer lo mismo en el otro diccionario monolingüe del par de traducción (en el ejemplo actual, el diccionario monolingüe de catalán) aplicando la misma estructura. El resultado sería:

```
<e lm="còsmic">
    <i>còsmi</i>
  <par n="acadèmi/c__adj"/>
</e>
```

Diccionario monolingüe (gallego)

Si lo que quiere es mejorar los diccionarios XML para el par español–gallego, deberá ir al directorio apertium-es-gl y abrir, con un editor de texto o un editor XML especializado, los tres ficheros de diccionarios apertium-es-gl.es.dix, apertium-es-gl.es-gl.dix y apertium-es-gl.gl.dix. En este caso, una vez ha añadido la nueva palabra *genoma* al diccionario monolingüe español (apertium-es-gl.es.dix), deberá añadir la palabra gallega equivalente *xenoma* al diccionario monolingüe gallego (apertium-es-gl.gl.dix), es decir:

```
<e lm="xenoma">
    <i>xenoma</i>
  <par n="Xulio__n"/>
</e>
```

Diccionario bilingüe

El último paso es añadir la traducción al diccionario bilingüe.

Generalmente, un diccionario bilingüe no tiene paradigmas, sino sólo lemas. Una entrada contiene únicamente el lema en ambos idiomas y el primer símbolo gramatical (la categoría léxica) de cada uno. Las entradas tienen una parte izquierda (`<l>`) y una parte derecha (`<r>`), y un idioma tiene que ocupar siempre la misma posición: en nuestro sistema, se ha convenido que el español ocupe la parte izquierda, y el catalán y el gallego la parte derecha.

Añadiendo el lema de ambas palabras, el sistema traducirá todas sus formas flexionadas (puesto que los símbolos gramaticales se copian desde la palabra original a la palabra en lengua meta). Esto sólo dará resultado si la palabra de origen y la palabra de destino son gramaticalmente equivalentes, es decir, si comparten exactamente los mismos símbolos gramaticales para todas sus formas flexionadas. Como este es el caso del ejemplo que nos ocupa, la entrada que hay que añadir al diccionario bilingüe es:

```
<e>
  <p>
    <l>cósmico<s n="adj"/></l>
    <r>còsmic<s n="adj"/></r>
  </p>
</e>
```

Esta entrada servirá para traducir todas las formas flexionadas, es decir, `adj m sg`, `adj f sg`, `adj m pl` y `adj f pl`. Sirve para ambas direcciones: de español a catalán y de catalán a español.

En el caso del par español–gallego, la siguiente entrada en el diccionario bilingüe español–gallego (`apertium-es-gl.es-gl.dix`) traducirá todas las formas flexionadas de las palabras equivalentes *genoma*/*xenoma* en ambas direcciones:

```
<e>
  <p>
    <l>genoma<s n="n"/></l>
    <r>xenoma<s n="n"/></r>
  </p>
</e>
```

En caso de que el par de traducción no sea gramaticalmente equivalente (sus símbolos gramaticales no sean exactamente los mismos), es necesario especificar todos los símbolos gramaticales (en el mismo orden en que

están especificados en los diccionarios monolingües) hasta llegar al símbolo que cambia entre la forma léxica origen y la forma léxica destino. Por ejemplo, el nombre español *limón*, de género masculino, se traduce al catalán por un nombre de género femenino, *llimona*. Por lo tanto, la entrada en el diccionario bilingüe debe ser:

```
<e>
  <p>
    <l>limón<s n="n"/><s n="m"/></l>
    <r>llimona<s n="n"/><s n="f"/></r>
  </p>
</e>
```

Un caso más complicado es el que surge cuando, teniendo dos formas léxicas con símbolos gramaticales diferentes, la información gramatical de la forma de la lengua origen no es suficiente para determinar el género (masculino o femenino) o el número (singular o plural) de la forma de la lengua destino. Así sucede, por ejemplo, con el adjetivo español *canadiense*. Su género es masculino–femenino puesto que es de invariable en género, es decir, puede acompañar a nombres tanto masculinos como femeninos (*hombre canadiense*, *mujer canadiense*. En catalán, en cambio, el adjetivo se flexiona de manera diferente para el masculino y para el femenino (*home canadenc*, *dona canadenca*). Así, al traducir de español a catalán no es posible saber, sin mirar el nombre al que acompaña, si el adjetivo español (*mf*) tiene que traducirse como un adjetivo femenino o masculino en catalán. En este caso se utiliza el símbolo `GD` (de "género por determinar") en lugar del símbolo de género. Será en el módulo de transferencia estructural donde se determinará el género de la palabra mediante una regla de transferencia (en este caso concreto, una regla que detecte el género del nombre precedente). El símbolo `GD` debe utilizarse sólo para la traducción de español a catalán, y no viceversa, ya que en español el género será siempre `mf` independientemente del género de la palabra original. En el diccionario bilingüe, habrá que añadir más de una entrada con restricciones de dirección, puesto que es necesario indicar en qué dirección de traducción el género queda sin determinar. Las entradas para este adjetivo deben ser como sigue:

```
<e r="LR">
  <p>
    <l>canadiense<s n="adj"/><s n="mf"/></l>
    <r>canadenc<s n="adj"/><s n="GD"/></r>
  </p>
```

```
</e>
<e r="RL">
  <p>
    <l>canadiense<s n="adj"/><s n="mf"/></l>
    <r>canadenc<s n="adj"/><s n="f"/></r>
  </p>
</e>
<e r="RL">
  <p>
    <l>canadiense<s n="adj"/><s n="mf"/></l>
    <r>canadenc<s n="adj"/><s n="m"/></r>
  </p>
</e>
```

"LR"significa *left to right* ("de izquierda a derecha") y "RL", *right to left* ("de derecha a izquierda"). Puesto que el español está situado a la izquierda y el catalán a la derecha, el adjetivo será GD sólo en la traducción de español a catalán (LR). Para la traducción RL hay que crear dos entradas, una para el adjetivo en femenino y otra para el adjetivo en masculino.[1]

Este mismo principio se aplica cuando no es posible determinar el número de la palabra destino por los mismos motivos que se acaban de exponer. Por ejemplo, el nombre español *rascacielos* es invariable en número, es decir, tiene la misma forma para el singular y para el plural (*un rascacielos*, *dos rascacielos*). En catalán, el nombre tiene diferente flexión para el singular y para el plurar (*un gratacel*, *dos gratacels*). En este caso, el símbolo utilizado es "ND"(de "número por determinar") y las entradas para estos lemas deben ser como sigue:

```
<e r="LR">
  <p>
    <l>rascacielos<s n="n"/><s n="m"/><s n="sp"/></l>
    <r>gratacel<s n="n"/><s n="m"/><s n="ND"/></r>
  </p>
</e>
<e r="RL">
  <p>
    <l>rascacielos<s n="n"/><s n="m"/><s n="sp"/></l>
    <r>gratacel<s n="n"/><s n="m"/><s n="pl"/></r>
  </p>
</e>
<e r="RL">
```

[1]También se podrían agrupar usando un pequeño paradigma

```
  <p>
    <l>rascacielos<s n="n"/><s n="m"/><s n="sp"/></l>
    <r>gratacel<s n="n"/><s n="m"/><s n="sg"/></r>
  </p>
</e>
```

Para obtener una descripción más detallada de este tipo de entradas, consulte lapágina 37.

5.2.1. Añadir restricciones de dirección

En el ejemplo anterior hemos visto el uso de restricciones de dirección en las entradas con género o número por determinar (`GD` o `ND`). Estas restricciones también pueden utilizarse en otros casos.

Hay que tener en cuenta que la versión actual de *Apertium* sólo puede proporcionar un único equivalente para cada forma léxica de la lengua origen, es decir, no realiza ningún tipo de desambiguación semántica.[2] Cuando una forma léxica puede traducirse a la lengua meta de dos o más maneras diferentes, hay que escoger una (las más general, la más frecuente, etc.). Puede indicarse al sistema que una forma sea analizada (.$^{\text{en}}$tendida") pero no generada, puesto que no es la traducción de ningún lema de la otra lengua.

Veámoslo con un ejemplo. El nombre español *muñeca* puede traducirse al catalán por dos palabras diferentes dependiendo de su significado: *canell* o *nina*. El contexto es el que decide cuál es la traducción correcta, pero el sistema *Apertium* actual no puede tomar este tipo de decisiones.[3] Por lo tanto, debe decidirse qué palabra será el equivalente para la traducción de español a catalán. Para la traducción catalán–español, ambos lemas pueden traducirse como *muñeca* y no surge ningún problema. Todas estas circunstancias deben quedar reflejadas en las entradas bilingües utilizando las restricciones de dirección (`LR`, "de izquierda a derecha", es decir, `es-ca`, y `RL`, "de derecha a izquierda", es decir, `ca-es`). Si se decide traducir *muñeca* como *canell* en todos los casos, las entradas que deben crearse en el diccionario bilingüe son:

[2] El sistema realiza únicamente desambiguación categorial para las palabras homógrafas, es decir, para formas que pueden tener como análisis más de una forma léxica, como ocurre con la forma *vino* en español, la cual puede ser tanto un sustantivo masculino como la forma pretérita del verbo *venir*. El desambiguador léxico categorial es el encargado de deshacer este tipo de ambigüedad.

[3] En el apartado 5.2.2 dedicado a la adición de multipalabras se explican maneras de sortear este problema.

```
<e>
  <p>
    <l>muñeca<s n="n"/><s n="f"/></l>
    <r>canell<s n="n"/><s n="m"/></r>
  </p>
</e>

<e r="RL">
  <p>
    <l>muñeca<s n="n"/></l>
    <r>nina<s n="n"/></r>
  </p>
</e>
```

Con estas entradas, las direcciones de traducción serán las siguientes:

```
muñeca --> canell
muñeca <-- canell
muñeca <-- nina
```

(Observe también que hay un cambio de género entre *muñeca* y *canell*).

Es muy importante que una forma léxica no tenga dos traducciones en la lengua meta, puesto que, de lo contrario, se produciría un error en el proceso de traducción (véase la sección 5.5 sobre cómo pueden detectarse y corregirse este y otros tipos de errores). Cuando una palabra puede traducirse a la lengua meta de dos maneras diferentes en todos los contextos, hay que elegir una como equivalente de traducción y dejar la otra como forma léxica analizable pero no generable, valiéndose de restricciones de dirección como en el ejemplo anterior. Por ejemplo, como los lemas catalanes *mot* y *paraula* pueden traducirse ambos al español como *palabra*, las entradas del diccionario bilingüe deberían ser como las siguientes:

```
<e>
  <p>
    <l>palabra<s n="n"/></l>
    <r>paraula<s n="n"/></r>
  </p>
</e>

<e r="RL">
  <p>
    <l>palabra<s n="n"/><s n="f"/></l>
    <r>mot<s n="n"/><s n="m"/></r>
  </p>
</e>
```

de las que resultan las siguientes direcciones de traducción:

```
palabra --> paraula
palabra <-- paraula
palabra <-- mot
```

En algunos casos puede ser necesario especificar restricciones de dirección también en los diccionarios monolingües. Por ejemplo, las dos formas verbales españolas *cantaran* y *cantasen* deben ser analizadas como lema *cantar*, verbo, pretérito imperfecto de subjuntivo, 3ª persona plural, pero al traducir al español, debe decidirse e indicarse qué forma será la que se generará. Los diccionarios monolingües se leen en dos direcciones según su finalidad: para el análisis, la dirección de lectura es de izquierda a derecha; para la generación, de derecha a izquierda. Por lo tanto, una palabra que debe analizarse pero no generarse debe contener la restricción `LR`, mientras que una que deba generarse pero no analizarse debe contener la restricción `RL`.

El caso de *cantaran* o *cantasen* debe haberse tratado en los paradigmas de flexión verbal y es poco probable que constituya un problema para alguien que esté ampliando un diccionario. En algunos otros casos puede ser necesario introducir una restricción en las entradas léxicas del diccionario monolingüe.

5.2.2. Añadir multipalabras

Es posible crear entradas formadas por dos o más palabras si se considera que éstas forman una sola ünidad de traducción". Las unidades multipalabra pueden ser útiles también cuando se trata de seleccionar el equivalente correcto de una palabra dentro de una expresión fija. Por ejemplo, la palabra española *dirección* puede traducirse al catalán de dos maneras: *direcció* (el sentido más general) y *adreça* (en el sentido de "dirección postal"); si se añaden, por ejemplo, unidades multipalabra frecuentes como *dirección general* → *direcció general* y *dirección postal* → *adreça postal* se pueden conseguir mejores traducciones en determinadas situaciones.

Las unidades multipalabra pueden clasificarse básicamente en dos categorías: multipalabras con flexión intercada y multipalabras sin flexión intercalada.

Multipalabras sin flexión intercalada

Son iguales a las entradas léxicas de una sola palabra, con la única diferencia de que debe inserirse el elemento `<b>` (que representa un espa-

cio en blanco) entre las palabras que forman la unidad. Así, si se quiere añadir, por ejemplo, la multipalabra española *hoy en día*, cuyo equivalente en catalán es *avui dia*, las entradas que habrá que crear en los diferentes diccionarios son:

- Diccionario monolingüe de español:

```
<e lm="hoy en día">
    <i>hoy<b/>en<b/>día</i>
  <par n="ahora__adv"/>
</e>
```

- Diccionario monolingüe de catalán:

```
<e lm="avui dia">
    <i>avui<b/>dia</i>
  <par n="ahir__adv"/>
</e>
```

- Diccionario bilingüe español–catalán:

```
<e>
  <p>
    <l>hoy<b/>en<b/>día<s n="adv"/></l>
    <r>avui<b/>dia<s n="adv"/></r>
  </p>
</e>
```

Para el par español–gallego, si desea añadir, por ejemplo, la multipalabra española *manga por hombro*, cuyo equivalente en gallego es *sen xeito nin modo*, las entradas que hay que crear son las siguientes:

- Diccionario monolingüe de español:

```
<e lm="manga por hombro">
    <i>manga<b/>por<b/>hombro</i>
  <par n="ahora__adv"/>
</e>
```

- Diccionario monolingüe de gallego:

```
<e lm="sen xeito nin modo">
    <i>sen<b/>xeito<b/>nin<b/>modo</i>
  <par n="Deo_gratias__adv"/>
</e>
```

- Diccionario bilingüe español–gallego:

```
<e>
  <p>
    <l>manga<b/>por<b/>hombro<s n="adv"/></l>
    <r>sen<b/>xeito<b/>nin<b/>modo<s n="adv"/></r>
  </p>
</e>
```

Breve introducción a los paradigmas

Los paradigmas de los ejemplos anteriores, como los adverbios, no tienen flexión, contienen sólo el símbolo gramatical de la forma léxica, como se puede ver a continuación:

```
<pardef n="ahora__adv">
  <e>
    <p>
      <l/>
      <r><s n="adv"/></r>
    </p>
  </e>
</pardef>
```

Los paradigmas se construyen del mismo modo que una entrada léxica. Hasta ahora, hemos visto entradas léxicas en que la parte invariable del lema se coloca entre :

```
<e lm="cósmico">
    <i>cósmic</i>
  <par n="absolut/o__adj"/>
</e>
```

Ahora bien, es posible expresar lo mismo mediante un par de cadenas: una cadena izquierda <l> y una cadena derecha <r> dentro de un elemento <p>:

```
<e lm="cósmico">
  <p>
    <l>cósmic</l>
    <r>cósmic</r>
  </p>
  <par n="absolut/o__adj"/>
</e>
```

Las dos entradas son equivalentes. El elemento `<i>` se utiliza porque permite obtener entradas más simples y compactas, y puede emplearse cuando la parte izquierda y la parte derecha del par de cadenas son idénticas. Como ya se ha indicado, los diccionarios monolingües se leen `LR` para el análisis de un texto y `RL` para la generación. Por ello, cuando hay alguna diferencia entre la cadena analizada y la cadena generada (cosa que no sucede normalmente) la entrada no puede escribirse utilizando la `<i>`. Esto es lo que sucede con los paradigmas, en los cuales las cadenas izquierda y derecha no son nunca idénticas, puesto que la parte derecha debe contener los símbolos gramaticales que circularán a través de todos los módulos del sistema.

Multipalabras con flexión intercalada

Consisten en una palabra con flexión (en general un verbo) seguida de una o más palabras invariables. En las entradas correspondientes debe especificarse el paradigma de flexión justo detrás de la palabra que se flexiona, mientras que la parte invariable debe quedar incluida en un elemento `<g>` (de *grupo*) en la cadena derecha. Los blancos entre palabras se indican, como en el caso anterior, mediante la colocación del elemento `<b>`. Su estructura puede observarse en el siguiente ejemplo, correspondiente a la multipalabra española *echar de menos*, que se traduce al catalán por *trobar a faltar*:

- Diccionario monolingüe de español:

```
<e lm="echar de menos">
    <i>ech</i>
    <par n="aspir/ar__vblex"/>
    <p>
      <l><b/>de<b/>menos</l>
      <r><g><b/>de<b/>menos</g></r>
    </p>
</e>
```

- Diccionario monolingüe de catalán:

```
<e lm="trobar a faltar">
    <i>trob</i>
    <par n="abander/ar__vblex"/>
    <p>
      <l><b/>a<b/>faltar</l>
```

```
      <r><g><b/>a<b/>faltar</g></r>
    </p>
</e>
```

- Diccionario bilingüe español–catalán:

```
<e>
  <p>
    <l>echar<g><b/>de<b/>menos</g><s n="vblex"/></l>
    <r>trobar<g><b/>a<b/>faltar</g><s n="vblex"/></r>
  </p>
</e>
```

Obsérvese cómo el símbolo gramatical se agrega al final, tras el grupo marcado con <g>.

Puede suceder también que un lema sea una multipalabra de este tipo en un idioma, mientras que en el otro idioma sea una única palabra. En este caso, en el diccionario bilingüe, la multipalabra contendrá el elemento <g> y la palabra simple no. En los diccionarios monolingües, se creará cada entrada de acuerdo con el tipo de lema. Esto se ilustra con el siguiente ejemplo, correspondiente a la multipalabra española *darse cuenta*, cuya equivalencia en catalán es el verbo *adonar-se*:[4]

- Diccionario monolingüe de español:

```
<e lm="darse cuenta">
    <i>d</i>
    <par n="d/ar__vblex"/>
    <p>
      <l><b/>cuenta</l>
      <r><g><b/>cuenta</g></r>
    </p>
</e>
```

- Diccionario monolingüe de catalán:

[4]El verbo *adonar-se* es considerado una palabra simple, pues la colocación de los pronombres enclíticos es tratada dentro del paradigma de flexión de los verbos (para todos los idiomas de *Apertium*); así, no es necesario especificarlos en las entradas léxicas. La correcta colocación de los pronombres clíticos es una de las principales razones de la utilización de las etiquetas <g>... </g> alrededor de la parte invariable de los verbos multipalabra.

```
<e lm="adonar-se">
    <i>adon</i>
    <par n="abander/ar__vblex"/>
</e>
```

- Diccionario bilingüe español–catalán:

```
<e>
  <p>
    <l>dar<g><b/>cuenta</g><s n="vblex"/></l>
    <r>adonar<s n="vblex"/></r>
  </p>
</e>
```

Los mismos principios y acciones descritos para las entradas básicas (cambio de género y número, restricciones de dirección, etc.) se aplican también a todos los tipos de multipalabras. Para obtener una descripción más detallada de las unidades multipalabra, consulte la página 42.

5.2.3. Colaborar en la mejora de datos léxicos

Si ha añadido satisfactoriamente datos léxicos de propósito general para cualquiera de los pares lingüísticos de *Apertium*, considere la opción de facilitarlos a los desarrolladores del proyecto para que podamos ofrecer una mejor herramienta a los usuarios. Puede enviar los datos léxicos por correo electrónico (en tres ficheros XML, uno para cada diccionario monolingüe y uno para el diccionario bilingüe) a las direcciones siguientes:

Datos español–catalán	Mireia Ginestí: `mginesti@dlsi.ua.es`
Datos español–portugués	Carme Armentano: `carmentano@dlsi.ua.es`[5]
Datos español–gallego	Xavier Gómez-Guinovart: `xgg@uvigo.es`

Si cree que va a contribuir de manera más decisiva al proyecto, puede unirse al grupo de desarrollo a través de www.sourceforge.net. Si no dispone de cuenta en Sourceforge, primero deberá crearse una; a continuación, escriba a Mikel L. Forcada (`mlf@ua.es`) o a Sergio Ortiz (`sortiz@dlsi.ua.es`), o bien a Xavier Gómez Guinovart si está interesado en el par español–gallego, explicando brevemente sus motivaciones y experiencia para unirse al proyecto. La forma usual de aportar datos es mediante CVS; como miembro del proyecto, podrá hacer `commit` para subir los cambios a los diccionarios directamente.

La adición de contribuciones léxicas se ha simplificado recientemente a través de formularios web en `http://xixona.dlsi.ua.es/prototype/webform/`, de manera que los colaboradores ocasionales no tienen que manejar ficheros XML.

Tenga en cuenta que los datos que aporte al proyecto serán distribuidos gratuitamente bajo la licencia que corresponda al producto (licencia GNU GPL o Creative Commons 2.5 Atribución-NoComercial-LicenciarIgual, según se indique). Asegúrese de que los datos que entregue no estén afectados por ningún tipo de licencia que pueda ser incompatible con las licencias utilizadas en este proyecto. No se creará ningún tipo de acuerdo o contrato entre usted y los desarrolladores. Si tiene alguna duda, o si tiene previsto hacer una contribución de gran envergadura, póngase en contacto con Mikel L. Forcada.

5.3. Introducir reglas de transferencia estructural

El contenido de este capítulo repite parcialmente información ya expuesta en el apartado de descripción del módulo de transferencia estructural (punto 3.4), aunque aquí se describen las reglas de una manera más general y práctica, orientada a quienes deseen una primera aproximación a su funcionamiento.

Las reglas de transferencia estructural llevan a cabo transformaciones en el texto analizado y desambiguado, que son necesarias debido a divergencias gramaticales, sintácticas y léxicas entre los dos idiomas implicados (cambios de género y de número para garantizar la concordancia en la lengua meta, reordenamientos de palabras, cambios de preposiciones, etc.). Las reglas detectan patrones (secuencias) de formas léxicas de la lengua origen y les aplican las transformaciones correspondientes. El módulo detecta los patrones de izquerda a derecha y recortando el segmento concordante más largo (*left-to-right, longest-match*): Por ejemplo, el sintagma *el gato verde* será detectado y procesado por la regla para *determinante–nombre–adjetivo* y no por la regla para *determinante–nombre*, puesto que el primer patrón es más largo. Si dos patrones tienen la misma longitud, la regla que se aplica es la definida en primer lugar.

El módulo de transferencia estructural (generado a partir del fichero de reglas de transferencia estructural) llama, durante el procesamiento, al módulo de transferencia léxica (generado a partir del diccionario bilingüe) para determinar los equivalentes en lengua de meta de las formas léxicas

en lengua origen.

Las reglas de transferencia estructural están guardadas en un fichero XML, uno para cada dirección de traducción (por ejemplo, para la traducción de español a catalán, el fichero se llama `apertium-es-ca.trules-es-ca.xml`). Deberá editar este fichero para añadir o modificar las reglas de transferencia.

Las reglas tienen un **patrón** y una **acción**. El patrón indica qué secuencias de formas léxicas tienen que ser detectadas y procesadas. La acción describe las verificaciones y transformaciones que deben realizarse en ellas. Algunas operaciones de transformación usuales (como las operaciones de concordancia de género y número) están definidas dentro de una macroinstrucción que se llama dentro de la regla. Al final de la parte de acción de cada regla, se envían las formas léxicas resultantes en la lengua meta para que puedan ser procesadas por los módulos posteriores del sistema de traducción.

Un fichero de reglas de transferencia contiene cuatro secciones con definiciones de elementos utilizados en las reglas, y una quinta sección en la que se definen las reglas propiamente dichas. Las secciones son las siguientes:

- `<section-def-cats>`: Esta sección contiene la definición de las categorías que se utilizarán en los patrones de las reglas (es decir, los tipos de formas léxicas que serán detectados por una regla determinada). Para la regla que se presenta más abajo, es necesario haber definido aquí las categorías `det` y `nom` (determinante y nombre). Las categorías se definen especificando los símbolos gramaticales que tienen las formas léxicas que se quiere incluir. Un asterisco indica que los símbolos gramaticales especificados van seguidos de uno o más símbolos en las formas léxicas correspondientes. A continuación se presenta la definición de la categoría `det`, que agrupa determinantes y predeterminantes[6] en una misma categoría pues para el módulo de transferencia desempeñan un papel similar:

```
<def-cat n="det">
    <cat-item tags="det.*"/>
    <cat-item tags="predet.*"/>
</def-cat>
```

  También es posible definir definir una categoría para un lema determinado, como la siguiente para la preposición `en`:

[6] como el español *todo*, *toda*, *todos*, *todas*

```
<def-cat n="en">
    <cat-item lemma="en" tags="pr"/>
</def-cat>
```

- <section-def-attrs>: Esta sección contiene la definición de los atributos que se utilizarán dentro de las reglas, en la parte de acción. Se necesitan atributos para las categorías definidas en la sección anterior si van a utilizarse en la parte de acción de una regla (para realizar verificaciones en ellas o para enviarlas al final de la regla), así como para otros atributos necesarios en la regla (como el género o el número). Los atributos se definen especificando los símbolos gramaticales correspondientes y no pueden llevar asteriscos; su nombre tiene que ser unívoco. A continuación se muestran las definiciones para los atributos a_det (para determinantes) y gen (para el género):

```
<def-attr n="a_det">
    <attr-item tags="det.def"/>
    <attr-item tags="det.ind"/>
    <attr-item tags="det.dem"/>
    <attr-item tags="det.pos"/>
    <attr-item tags="predet"/>
</def-attr>

<def-attr n="gen">
    <attr-item tags="m"/>
    <attr-item tags="f"/>
    <attr-item tags="mf"/>
    <attr-item tags="nt"/>
    <attr-item tags="GD"/>
</def-attr>
```

- <section-def-vars>: Esta sección contiene la definición de las variables utilizadas en las reglas.

```
<def-var n="interrogativa"/>
```

- <section-def-macros>: Aquí se definen las macroinstrucciones, que contienen secuencias de código utilizado con frecuencia dentro de las reglas; esto evita tener que escribir repetidamente las mismas acciones al crear las reglas. Hay macroinstrucciones, por ejemplo, para operaciones de concordancia de género y número.

- `<section-def-rules>`: Esta es la sección en la que se escriben las reglas de transferencia estructural.

A continuación se presenta como ejemplo una regla que detecta la secuencia *determinante–nombre*:

```
<rule>
  <pattern>
    <pattern-item n="det"/>
    <pattern-item n="nom"/>
  </pattern>
  <action>
    <call-macro n="f_concord2">
      <with-param pos="2"/>
      <with-param pos="1"/>
    </call-macro>
    <out>
      <lu>
        <clip pos="1" side="tl" part="whole"/>
      </lu>
      <b pos="1"/>
      <lu>
        <clip pos="2" side="tl" part="whole"/>
      </lu>
    </out>
  </action>
</rule>
```

Una parte de la acción ejecutada en este patrón está especificada dentro de la macroinstrucción `f_concord2`, que está definida en la sección `<section-def-macros>`. Esta macroinstrucción realiza operaciones de concordancia de género y de número: si se produce un cambio de género o número entre la lengua origen y la lengua meta (en el nombre), se cambia el género o el número del determinante de acuerdo con ello; además, si el género o el número son indeterminados (`GD` o `ND`[7]), el nombre recibe el valor correcto de género o número aportado por el determinante que le precede. En Apertium es–ca, es–gl y es–pt, se han definido macroinstrucciones de concordancia para una, dos, tres o cuatro formas léxicas (`f_concord1`, `f_concord2`, `f_concord3`, `f_concord4`). Cuando se llama una macroinstrucción en una regla, debe indicarse cuál es la forma léxica principal (la que tiene más peso en la determinación de género o

[7] Véanse las páginas 39 o 105

número de las demás formas léxicas) y qué otras formas léxicas del patrón tienen que incluirse en las operaciones de concordancia, en orden de importancia. Esto se realiza con el elemento `<with-param pos=/>`. En la regla precedente, la forma léxica principal es el nombre (posición "2.[en] el patrón) y la segunda es el determinante (posición "1.[en] el patrón).

Tras las acciones pertinentes, se envían las formas léxicas resultantes, dentro del elemento `<out>`. Cada forma léxica se define mediante un `<clip>`, cuyos atributos significan lo siguiente:

- `pos`: hace referencia a la posición de la forma léxica en el patrón; `1` representa la primera forma léxica (el determinante) y `2` la seguna (el nombre).

- `side`: indica si la forma léxica está en lengua origen (`sl`) o en lengua meta (`tl`). Naturalmente, al final de la regla las palabras se envían siempre en la lengua meta; las formas léxicas en lengua origen pueden necesitarse dentro de una regla para evaluar sus atributos o características.

- `part`: indica qué parte de la forma léxica incluye el `clip`. Pueden utilizarse algunos valores predefinidos:

    - `whole`: la forma léxica completa (lema más símbolos gramaticales). Se utiliza sólo al enviar la forma léxica (dentro de un elemento `<out>`).
    - `lem`: el lema de la forma léxica
    - `lemh`: la cabeza del lema de una multipalabra con flexión intercalada (véase el apartado 5.2.2 de este capítulo, o bien la página 42 si desea una explicación más detallada)
    - `lemq`: la cola del lema de una multipalabra con flexión intercalada

  A parte de estos valores predefinidos, puede utilizarse cualquiera de los atributos definidos en la sección `<section-def-attrs>` (como `gen` o `a_det`).

  Los valores `lemh` y `lemq` se utilizan al enviar multipalabras con flexión intercalada para poder colocar la cabeza y la cola del lema en la posición correcta, puesto que el módulo previo desplazó la cola justo después de la cabeza del lema por varios motivos. En la práctica, en nuestro sistema, esto significa que para enviar verbos hay que utilizar estos valores en lugar de `whole`. Esto se debe a que, en nuestros

diccionarios, las multipalabras con flexión intercalada son siempre verbos y, si se utiliza el valor `whole` al enviarlos al final de una regla, la multipalabra quedaría mal formada (la cabeza y la cola del lema no tendrían la posición correcta y la multipalabra no podría generarse en el generador).

Según lo dicho, pues, una regla que contenga un verbo en el patrón debe enviar las formas léxicas como en estos dos ejemplos:

```
<rule>
  <pattern>
    <pattern-item n="verb"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lemh"/>
        <clip pos="1" side="tl" part="a_verb"/>
        <clip pos="1" side="tl" part="temps"/>
        <clip pos="1" side="tl" part="persona"/>
        <clip pos="1" side="tl" part="gen"/>
        <clip pos="1" side="tl" part="nbr"/>
        <clip pos="1" side="tl" part="lemq"/>
      </lu>
    </out>
  </action>
</rule>


<rule>
  <pattern>
    <pattern-item n="verb"/>
    <pattern-item n="prnenc"/>
  </pattern>
  <action>
    <out>
      <mlu>
        <lu>
          <clip pos="1" side="tl" part="lemh"/>
          <clip pos="1" side="tl" part="a_verb"/>
          <clip pos="1" side="tl" part="temps"/>
          <clip pos="1" side="tl" part="persona"/>
          <clip pos="1" side="tl" part="nbr"/>
        </lu>
```

```
        <lu>
          <clip pos="2" side="tl" part="lem"/>
          <clip pos="2" side="tl" part="a_prnenc"/>
          <clip pos="2" side="tl" part="persona"/>
          <clip pos="2" side="tl" part="gen"/>
          <clip pos="2" side="tl" part="nbr"/>
          <clip pos="1" side="tl" part="lemq"/>
        </lu>
      </mlu>
    </out>
  </action>
</rule>
```

La primera regla detecta un verbo y coloca la cola en la posición correcta, después de todos los símbolos gramaticales. La unidad léxica se envía especificando sus atributos por separado: cabeza del lema, categoría léxica (verbo), tiempo, persona, género (para los participios), número y cola del lema.

La segunda regla detecta un verbo seguido de un pronombre enclítico y envía dos formas léxicas especificando también sus atributos por separado; la primera unidad léxica se compone de: cabeza del lema, categoría léxica (verbo), tiempo, persona y número; la segunda unidad léxica se compone de: lema, categoría léxica (pronombre enclítico) , persona, género, número y cola del lema (de la primera unidad léxica). De este modo, la cola se coloca después del pronombre enclítico. Las dos unidades léxicas (verbo y pronombre enclítico) se envían dentro de un elemento `<mlu>`, puesto que tienen que llegar al generador morfológico como unidad multiléxica.

Teniendo en cuenta todo lo que se ha explicado, para **añadir una regla de transferencia** deben seguirse los pasos siguientes:

1. Especificar el patrón que se quiere detectar. Téngase presente que las palabras sólo son procesadas una vez por las reglas, y que las reglas se aplican de izquierda a derecha y escogiendo el segmento concordante más largo. Por ejemplo, supongamos que tenemos un fichero de reglas de transferencia con sólo dos reglas, una para el patrón *determinante–nombre* y otra para el patrón *nombre–adjetivo*. El sintagma español *el valle verde* sería detectado y procesado por la primera regla, no por la segunda. Si queremos que las tres palabras sean procesadas por una misma regla, deberemos añadir una regla para el patrón *determinante - nombre - adjetivo*

2. Describir las operaciones que deben realizarse sobre el patrón. En los sistemas Apertium `es-ca`, `es-gl` y `es-pt`, las operaciones normales de concordancia (de género y número) son fáciles de especificar utilizando una macroinstrucción. Para realizar otras operaciones hay que utilizar elementos del lenguaje más complicados, que están explicados con detalle en la sección 3.4.2.

3. Enviar las unidades léxicas del patrón en la lengua meta dentro de un elemento `<out>`. Cada unidad léxica debe estar dentro de un elemento `<lu>`. Si dos o más unidades léxicas deben ser generadas como una unidad multiléxica (sólo para pronombres enclíticos en los pares de lenguas actuales), deben estar agrupadas dentro de un elemento `<mlu>`.

   Todas las palabras que sean detectadas por la regla (que son parte del patrón) deben enviarse al final de la regla para que el módulo siguiente (el generador) las reciba. Si una unidad léxica es detectada por un patrón y no se incluye dentro del elemento `<out>`, no se generará.

5.4. Añadir datos para el desambiguador léxico categorial

La información lingüística que necesita el desambiguador léxico categorial para desambiguar un texto proviene básicamente de dos fuentes: del fichero de definición del desambiguador y de corpus. Los ficheros de definición se encuentran en el directorio de datos lingüísticos y su nombre tiene la estructura `apertium-`L_1`-`L_2`.`L_1`.tsx` y `apertium-`L_1`-`L_2`.`L_2`.tsx`, mientras que los corpus se encuentran en los directorios L_1`-tagger-data` y L_1`-tagger-data` que están contenidos en el directorio anterior. En este directorio se encuentran los ficheros siguientes (ejemplos de `es-tagger-data`):

- `es.tagged.txt`: Corpus españo en formato de texto llano.
- `es.untagged`: Corpus analizado morfológicamente (procesado por el desformateador de texto y el analizador morfológico).
- `es.tagged`: El corpus anterior desambiguado manualmente.
- `es.crp.txt`: Un corpus grande (cientos de miles de palabras) utilizado cuando se entrena el desambiguador de forma sno supervisada mediante el algoritmo de maximización de la esperanza matemática.

- `es.crp`: El corpus anterior procesado consecutivamente por el desformateador de texto y el analizador morfológico.

- `es.dic`: Calculado a partir del diccionario `*.es.dix`, y mediante el uso de las herramientas `lt-expand` y `apertium-filter-ambiguity` contiene todas las formas distintas que el etiquetador definido mediante el archivo `*.es.tsx` identifica como clases de ambigüedad diferentes (es decir, qué categorías léxicas pueden ser homógrafas). Si se añaden al diccionario palabras que comporten la aparición de una nueva clase de ambigüedad, será necesario volver a generar este fichero.

Al bajarse el traductor Apertium de Sourceforge (`http://apertium.sourceforge.net/` se obtendrán sólo los tres primeros ficheros, de modo que deberán generarse los últimos tres, expandiendo los diccionarios y creando un corpus de gran tamaño, aunque este gran corpus no es necesario si no se va a entrenar el desambiguador con reestimación de Baum-Welch.

Rl entrenamiento del desambiguador puede ser supervisado (con texto desambiguado a mano) o no supervisado (sin texto desambiguado a mano), de modo que:

- Para entrenar el desambiguador utilizando texto desambiguado a mano se necesitan los ficheros (para español): `es.untagged`, `es.tagged`, `es.dic`.

- Para entrenar el desambiguador de forma no supervisada mediante el algoritmo de Baum y Welch se necesitan los ficheros (para español): `es.dic`, `es.crp`

Los ficheros `*.crp`, `*.untagged`, `*.dic` son generados automáticamente mediante `make` a partir del resto de los ficheros.

Aparte de estos ficheros contenidos en los directorios L_1`-tagger-data` y L_1`-tagger-data`, se necesitan los mencionados ficheros de definición `apertium-`L_1`-`L_2`.`L_1`.tsx` y `apertium-`L_1`-`L_2`.`L_2`.tsx`. Este fichero contiene la definición de las etiquetas gruesas (o categorías) que utiliza el desambiguador durante su entrenamiento y al desambiguar un texto, así como restricciones de ocurrencia de etiquetas que ayudan a obtener mejores probabilidades para la desambiguación.

De acuerdo con todo lo expuesto, se pueden modificar los datos del desambiguador de dos maneras:

1. Modificar el fichero de definición del desambiguador. Se pueden añadir, modificar o suprimir etiquetas gruesas si se considera que una nueva categoría sería útil para la desambiguación o que la modificación de una categoría comportaría mejoras. También pueden añadirse restricciones (por ejemplo, se puede prohibir la secuencia determinante–determinante en un idioma si se trata de una combinación imposible o poco probable y con ello puede mejorarse la desambiguación de ciertas palabras homógrafas).

2. Modificar los corpus utilizados para entrenar el desambiguador. Puede modificarse el texto desambiguado a mano (`es.tagged` para español) para corregir algunas etiquetas mal elegidas. También pueden añadirse frases a este texto (así como el corpus `es.tagged.txt`, usado para generar automáticamente el corpus `es.untagged`) para dar más información al desambiguador, puesto que es posible que la mala desambiguación de ciertas palabras se deba a su inexistencia en los corpus de entrenamiento.

   Si se modifican los datos utilizados por el desambiguador (los corpus o el fichero de definición) es necesario reentrenarlo; para ello, debe escribirse `make tagger_supervised` o `make tagger_unsupervised` en el directorio en el que se guardan los datos lingüísticos, dependiendo de si el entrenamiento es supervisado o no supervisado, respectivamente.

5.5. Detección de errores

Es fácil cometer errores al añadir palabras o reglas de transferencia al sistema Apertium.

Por un lado es posible que, al compilar los ficheros nuevos, el sistema muestre un mensaje de error. En este caso se tratará muy probablemente de un error formal (una etiqueta XML que falta, una etiqueta no permitida en un determinado contexto, etc.). Sólo hay que ir al número de línea indicado en el mensaje de error, corregirlo y compilar de nuevo. Por otro lado, hay otros tipos de errores que no se detectan al compilar pero que pueden provocar una mala traducción o la traducción de una palabra por una cadena de texto incomprensible. Éstos son errores lingüísticos, que pueden detectarse y corregirse con la ayuda de la información ofrecida en este capítulo. La siguiente información está dirigida a los usuarios de Linux, puesto que por el momento Apertium funciona únicamente con este

sistema operativo.[8]

Como todos los datos procesados por el sistema, desde el texto original al texto traducido, circulan entre los ocho módulos del sistema en formato de texto, es posible detener el flujo de texto en cualquier punto para examinar la entrada o la salida de un módulo determinado. Para ello sólo hay que escribir los comandos adecuados en el terminal utilizando una estructura de tubería, de modo que la salida estándar de un módulo se utilice como la entrada estándar del siguiente.

Mediante los comandos `echo` o `cat` se puede enviar texto a través de uno o varios módulos para analizar la salida y detectar el origen del error. Debe irse al directorio en que se guardan los datos lingüísticos y escribir los comandos explicados a continuación.

5.5.1. La salida del analizador morfológico

Para saber cómo el traductor analiza una palabra, escriba lo siguiente en el terminal (ejemplo para la palabra catalana *sabates*):

```
echo "sabates" | apertium-destxt | lt-proc ca-es.automorf.bin
```

Puede sustituirse `ca-es` por la dirección de traducción que se desea comprobar.

La salida de Apertium debería ser:

```
^sabates/sabata<n><f><pl>$^./.<sent>$[][]
```

La estructura de la cadena es `^palabra/lema<`*análisis morfológico*`>$`. La etiqueta `<sent>` corresponde al análisis del punto, ya que el sistema representa cada fin de frase como un punto, tanto si aparece explícitamente en el texto como si no.

El análisis de una palabra desconocida es (prescindiendo de la información del final de frase):

```
^genoma/*genoma$
```

y el análisis de una palabra ambigua:

```
^casa/casa<n><f><sg>/casar<vblex><pri><p3><sg>/casar<vblex><imp><p2><sg>$
```

Cada forma léxica (lema más análisis morfológico) es presentada como análisis posible de la palabra *casa*.

[8] Existen en `http://apertium.org` paquetes experimentales para Windows con los datos lingüísticos fijos (binarios no modificables)

5.5.2. La salida del desambiguador

Para saber cuál es la salida que da el desambiguador de un texto, escriba lo siguiente en el terminal (ejemplo para la dirección catalán – español):

```
echo "sabates" | apertium-destxt | lt-proc ca-es.automorf.bin
|./ca-es.tagger --tagger ca-es
```

La salida será:

```
^sabata<n><f><pl>$^./.<sent>$[][]
```

La salida de una palabra ambigua será igual que esta, puesto que el desambiguador elige una forma léxica de entre todas las posibles. Por lo tanto, la salida para la palabra *casa* en catalán será, por ejemplo (según el contexto):

```
^casa<n><f><sg>$^.<sent>$[][]
```

5.5.3. La salida del módulo `pretransfer`

Este módulo aplica algunos cambios a las multipalabras (desplaza la cola del lema de una multipalabra con flexión intercalada justo detrás de la cabeza del lema). Para conocer su salida, escriba:

```
echo "sabates" | apertium-destxt | lt-proc ca-es.automorf.bin
|./ca-es.tagger --tagger ca-es | apertium-pretransfer
```

Como *sabates* no es una multipalabra este módulo no altera la entrada.

5.5.4. La salida del módulo de transferencia estructural

Para saber cómo se traduce a la lengua meta una palabra o un grupo de palabras y cómo lo procesan las reglas de transferencia estructural, escriba lo siguiente en el terminal:

```
echo "sabates" | apertium-destxt | lt-proc ca-es.automorf.bin
|./ca-es.tagger --tagger ca-es | apertium-pretransfer
| ./ca-es.transfer ca-es.autobil.bin
```

La salida para esta palabra sería:

```
^zapato<n><m><pl>$^.<sent>$[][]
```

Analizar cómo una palabra o frase sale de este módulo puede ayudar a detectar errores en el diccionario bilingüe o en las reglas de transferencia estructural. Los errores más comunes debidos al diccionario bilingüe son: la existencia de dos equivalentes para una misma forma léxica de la lengua origen, o una asignación incorrecta de símbolos morfológicos. Los errores debidos a las reglas de transferencia estructural pueden ser muy variados ya que dependen de las acciones realizadas por las reglas.

5.5.5. La salida del generador morfológico

Para saber cómo el sistema genera una palabra, escriba lo siguiente en el terminal:

```
echo "sabates" | apertium-destxt | lt-proc ca-es.automorf.bin
|./ca-es.tagger --tagger ca-es | apertium-pretransfer
| ./ca-es.transfer ca-es.autobil.bin | ltproc -g ca-es.autogen.bin
```

Con este comando se pueden detectar errores de generación causados por una entrada incorrecta en el diccionario monolingüe de la lengua meta o por divergencias entre la salida del diccionario bilingüe (la salida del módulo anterior) y la entrada en el diccionario monolingüe.

La salida correcta para la entrada *sabates* sería:

```
zapatos.[][]
```

Aquí ya no aparecen los símbolos morfológicos y la palabra aparece flexionada.

5.5.6. La salida del postgenerador

No es muy usual que se produzcan errores debido al postgenerador, ya que generalmente el diccionario de postgeneración es de pequeño tamaño y raras veces se modifica una vez añadidas las combinaciones más usuales, pero también puede comprobar cómo un texto en lengua origen sale de este módulo, escribiendo:

```
echo "sabates" | apertium-destxt | lt-proc ca-es.automorf.bin
|./ca-es.tagger --tagger ca-es | apertium-pretransfer
| ./ca-es.transfer ca-es.autobil.bin | ltproc -g ca-es.autogen.bin
| ltproc -p es-ca.autopgen.bin
```

5.5.7. La salida de Apertium

Se pueden poner todos los módulos del sistema en la estructura de tubería para ver cómo un texto en lengua origen pasa a través de los módulos y es traducido a la lengua meta. Sólo hace falta añadir el reformateador al comando anterior:

```
echo "sabates" | apertium-destxt | lt-proc ca-es.automorf.bin
|./ca-es.tagger --tagger ca-es | apertium-pretransfer
| ./ca-es.transfer ca-es.autobil.bin | ltproc -g ca-es.autogen.bin
| ltproc -p es-ca.autopgen.bin | apertium-retxt
```

Esto es lo mismo que utilizar el *script* (guión del intérprete de comandos de Linux) `apertium-translator` proporcionado por el paquete Apertium:

```
echo "sabates" | apertium-translator . ca-es
```

(El punto indica el directorio en el que se guardan los datos lingüísticos, en este caso el directorio actual).

Naturalmente, en lugar de escribir todos los comandos presentados cada vez que necesite probar una traducción, puede crear *scripts* para cada acción y utilizarlos para probar la salida de cada módulo.

5.5.8. Ejemplos de errores

Cuando se produce algún tipo de error lingüístico en la traducción, el sistema da como salida un símbolo seguido de la forma léxica causante del problema. El significado de los símbolos es el siguiente:

- `#` = El problema está en el generador, que no puede generar la forma léxica (el diccionario morfológico no la contiene en la dirección de generación)
- `@` = El problema está en el módulo de transferencia léxica, que no puede traducir la forma léxica (el diccionario bilingüe no la contiene)

1) Por ejemplo, al escribir el comando siguiente se puede obtener el resultado mostrado:

```
$ echo "nord" | apertium-translator . ca-es
$ #norte<n><m><sg>
```

Esto significa que la palabra ha sido traducida correctamente por el diccionario bilingüe pero que el sistema no la ha encontrado en el diccionario morfológico de español para generarla. El problema puede encontrarse en el diccionario morfológico, pero puede deberse también a una entrada bilingüe incorrecta, en la que los símbolos morfológicos que recibe la palabra traducida no se corresponden con los símbolos morfológicos que esta palabra tiene en el diccionario morfológico.

2) La siguiente entrada bilingüe `es-ca` no tiene en cuenta el cambio de género entre *adhesiu* (masculino) y *pegatina* (femenino), con lo qual provoca un error del traductor:

```
<e>
  <p>
    <l>pegatina<s n="n"/></l>
    <r>adhesiu<s n="n"/></r>
  </p>
</e>

$ echo "adhesiu" | apertium-translator . ca-es
$ #pegatina<n><m><sg>
```

La entrada correcta es:

```
<e>
  <p>
    <l>pegatina<s n="n"/><s n="f"/></l>
    <r>adhesiu<s n="n"/><s n="m"/></r>
  </p>
</e>
```

3) El siguiente error se produce cuando la forma léxica de la lengua origen no se ha podido encontrar en el diccionario bilingüe, ya sea porque no hay ninguna entrada para este lema o porque la entrada no se corresponde con los símbolos morfológicos recibidos del analizador:

```
$ echo "illot" | apertium-translator . ca-es
$ @illot<n><m><sg>
```

4) Cuando una forma léxica de la lengua origen tiene dos equivalentes en el diccionario bilingüe, la salida del traductor será como la que sigue:

```
$ echo "llavor" | apertium-translator . ca-es
$ #pepita<n>/semilla<n><m><sg>
```

La solución es poner una restricción de dirección en una de las entradas bilingües.

Algunos errores pueden ser debidos a reglas de transferencia estructural. La manera de solucionar un error cuya causa no conocemos es comprobar la salida de cada uno de los módulos para detectar en qué punto surge el problema.

5.6. Generar un nuevo sistema con los nuevos datos

Si se realiza algún cambio a cualquiera de los ficheros de datos de Apertium (diccionarios, reglas de transferencia o fichero de definición del desambiguador), es necesario volver a compilar los módulos para que el sistema incorpore estos cambios. Para ello, hay que escribir `make` en el directorio en el que se guardan los datos lingüísticos, y el sistema genererará los nuevos ficheros binarios.

Si se ha hecho algún cambio en el fichero de definición del desambiguador o en el corpus utilizado para entrenar el mismo, será necesario además reentrenar el desambiguador: en el mismo directorio de datos lingüísticos, debe escribirse `make tagger_supervised` o `make tagger_unsupervised` según sea el caso.

Tras las operaciones de compilación, `apertium-translator` ya utilizará los nuevos datos.

Apéndice A

Definiciones de tipos de documento (DTD) en XML

A.1. DTD para el formato de los diccionarios

Definición de tipo de documento para el formato de los diccionarios morfológicos, bilingües y de postgeneración en XML; esta definición viene con el paquete `apertium` (última versión) que se puede descargar de `http://www.sourceforge.net`.

La explicación de los distintos elementos se encuentra en la página 24 y siguientes.

```
<!ELEMENT dictionary (alphabet?, sdefs,
                      pardefs, section+)>

<!ELEMENT alphabet (#PCDATA)>

<!ELEMENT sdefs (sdef+)>

<!ELEMENT sdef EMPTY>
<!ATTLIST sdef
        n ID #REQUIRED>

<!ELEMENT pardefs (pardef*)>

<!ELEMENT pardef (e+)>
<!ATTLIST pardef
        n CDATA #REQUIRED>

<!ELEMENT section (e+)>
```

```
<!ATTLIST section
        id ID #REQUIRED
        type (standard|inconditional) #REQUIRED>

<!ELEMENT e (i | p | par | re)+>
<!ATTLIST e
        r (LR|RL) #IMPLIED
        lm CDATA #IMPLIED
        a CDATA #IMPLIED
        c CDATA #IMPLIED>

<!ELEMENT par EMPTY>
<!ATTLIST par
        n CDATA #REQUIRED>

<!ELEMENT i (#PCDATA | b | s | g | j | a)*>

<!ELEMENT re (#PCDATA)>

<!ELEMENT p (l, r)>

<!ELEMENT l (#PCDATA | a | b | g | j | s)*>

<!ELEMENT r (#PCDATA | a | b | g | j | s)*>

<!ELEMENT a EMPTY>

<!ELEMENT b EMPTY>

<!ELEMENT g (#PCDATA | a | b | j | s)*>
<!ATTLIST g
        i CDATA #IMPLIED>

<!ELEMENT j EMPTY>

<!ELEMENT s EMPTY>
<!ATTLIST s
        n IDREF #REQUIRED>
```

A.2. DTD para el formato de los ficheros del desambiguador

DTD que define el formato del fichero de especificación del desambiguador. Esta definición viene con el paquete `apertium` (última versión) que se puede descargar de `http://www.sourceforge.net`.

La descripción de los distintos elementos se encuentra en la página 52.

```
<!ELEMENT tagger (tagset,forbid?,enforce-rules?,preferences?)>
<!ATTLIST tagger name CDATA #REQUIRED>

<!ELEMENT tagset (def-label+,def-mult*)>

<!ELEMENT def-label (tags-item+)>
<!ATTLIST def-label name CDATA #REQUIRED
                    closed CDATA #IMPLIED>

<!ELEMENT tags-item #EMPTY>
<!ATTLIST tags-item tags CDATA #REQUIRED
                    lemma CDATA #IMPLIED>

<!ELEMENT def-mult (sequence+)>
<!ATTLIST def-mult name CDATA #REQUIRED
                   closed CDATA #IMPLIED>

<!ELEMENT sequence ((tags-item|label-item)+)>

<!ELEMENT label-item #EMPTY>
<!ATTLIST label-item label CDATA #REQUIRED>

<!ELEMENT forbid (label-sequence+)>

<!ELEMENT label-sequence (label-item+)>

<!ELEMENT enforce-rules (enforce-after+)>

<!ELEMENT enforce-after (label-set)>
<!ATTLIST enforce-after label CDATA #REQUIRED>

<!ELEMENT label-set (label-item+)>

<!ELEMENT preferences (prefer+)>
```

```
<!ELEMENT prefer EMPTY>
<!ATTLIST prefer tags CDATA #REQUIRED>
```

A.3. DTD del módulo de transferencia estructural

DTD para el formato de las reglas de transferencia estructural. Esta Esta definición viene con el paquete `apertium` (última versión) que se puede descargar de `http://www.sourceforge.net`.

La descripción de los distintos elementos se encuentra en el apartado 3.4.2.

```
<!ENTITY % condition "(and|or|not|equal|in)">

<!ENTITY % container "(var|clip)">

<!ENTITY % sentence "(let|out|choose|modify-case|call-macro)">

<!ENTITY % value "(b|clip|lit|lit-tag|var|get-case-from|case-of)">

<!ENTITY % stringvalue "(clip|lit|var|get-case-from|case-of)">

<!ELEMENT transfer (section-def-cats, section-def-attrs,
                    section-def-vars, section-def-lists?,
                    section-def-macros?, section-rules)>

<!ELEMENT section-def-cats (def-cat+)>

<!ELEMENT def-cat (cat-item+)>
<!ATTLIST def-cat  n ID #REQUIRED>

<!ELEMENT cat-item EMPTY>
<!ATTLIST cat-item lemma CDATA #IMPLIED
                   tags CDATA #REQUIRED >

<!ELEMENT section-def-attrs (def-attr+)>

<!ELEMENT def-attr (attr-item+)>
<!ATTLIST def-attr n ID #REQUIRED>

<!ELEMENT attr-item EMPTY>
```

```
<!ATTLIST attr-item tags CDATA #IMPLIED>

<!ELEMENT section-def-vars (def-var+)>

<!ELEMENT def-var EMPTY>
<!ATTLIST def-var n ID #REQUIRED>

<!ELEMENT section-def-lists (def-list)+>

<!ELEMENT def-list (list-item+)>
<!ATTLIST def-list n ID #REQUIRED>

<!ELEMENT list-item EMPTY>
<!ATTLIST list-item v CDATA #REQUIRED>

<!ELEMENT section-def-macros (def-macro)+>

<!ELEMENT def-macro (%sentence;)+>
<!ATTLIST def-macro n ID #REQUIRED
                    npar CDATA #REQUIRED>

<!ELEMENT section-rules (rule+)>

<!ELEMENT rule (pattern, action)>

<!ELEMENT pattern (pattern-item+)>

<!ELEMENT pattern-item EMPTY>
<!ATTLIST pattern-item n IDREF #REQUIRED>

<!ELEMENT action (%sentence;)*>

<!ELEMENT choose (when+,otherwise?)>

<!ELEMENT when (test,(%sentence;)*)>

<!ELEMENT otherwise (%sentence;)+>

<!ELEMENT test (%condition;)+>

<!ELEMENT and ((%condition;),(%condition;)+)>

<!ELEMENT or ((%condition;),(%condition;)+)>
```

```
<!ELEMENT not (%condition;)>

<!ELEMENT equal (%value;,%value;)>
<!ATTLIST equal caseless (no|yes) #IMPLIED>

<!ELEMENT in (%value;, list)>
<!ATTLIST in caseless (no|yes) #IMPLIED>

<!ELEMENT list EMPTY>
<!ATTLIST list n IDREF #REQUIRED>

<!ELEMENT let (%container;,%value;)>

<!ELEMENT out (mlu|lu|b)+>

<!ELEMENT modify-case (%container;,%stringvalue;)>

<!ELEMENT call-macro (with-param)*>
<!ATTLIST call-macro n IDREF #REQUIRED>

<!ELEMENT with-param EMPTY>
<!ATTLIST with-param pos CDATA #REQUIRED>

<!ELEMENT clip EMPTY>
<!ATTLIST clip pos CDATA #REQUIRED
               side (sl|tl) #REQUIRED
               part CDATA #REQUIRED>

<!ELEMENT lit EMPTY>
<!ATTLIST lit v CDATA #REQUIRED>

<!ELEMENT lit-tag EMPTY>
<!ATTLIST lit-tag v CDATA #REQUIRED>

<!ELEMENT var EMPTY>
<!ATTLIST var n IDREF #REQUIRED>

<!ELEMENT get-case-from (clip|lit|var)>
<!ATTLIST get-case-from pos CDATA #REQUIRED>

<!ELEMENT case-of EMPTY>
<!ATTLIST case-of pos CDATA #REQUIRED
```

```
                side (sl|tl) #REQUIRED
                part CDATA #REQUIRED>

<!ELEMENT mlu (lu+)>

<!ELEMENT lu (%value;)+>

<!ELEMENT b EMPTY>
<!ATTLIST b pos CDATA #IMPLIED>
```

A.4. DTD para las reglas de especificación de formato

DTD para las reglas de especificación de formato, que puede descargarse de la página web `http://cvs.sourceforge.net/viewcvs.py/apertium/apertium/apertium/format.dtd`.

La descripción de los distintos elementos se encuentra en el apartado 3.5.2.

```
<!ELEMENT format (options,rules)>
<!ATTLIST format name CDATA #REQUIRED>

<!ELEMENT options (largeblocks, input, output,
                   escape-chars, space-chars, case-sensitive)>

<!ELEMENT largeblocks EMPTY>
<!ATTLIST largeblocks size CDATA #REQUIRED>

<!ELEMENT input EMPTY>
<!ATTLIST input zip-path CDATA #IMPLIED
               encoding CDATA #REQUIRED>

<!ELEMENT output EMPTY>
<!ATTLIST output zip-path CDATA #IMPLIED
                encoding CDATA #REQUIRED>

<!ELEMENT escape-chars EMPTY>
<!ATTLIST escape-chars regexp CDATA #REQUIRED>

<!ELEMENT space-chars EMPTY>
<!ATTLIST space-chars regexp CDATA #REQUIRED>
```

```
<!ELEMENT case-sensitive EMPTY>
<!ATTLIST case-sensitive value (yes|no) #REQUIRED>

<!ELEMENT rules (format-rule|replacement-rule)+>

<!ELEMENT format-rule (begin-end|(begin,end))>
<!ATTLIST format-rule eos (yes|no) #IMPLIED
                      priority CDATA #REQUIRED>

<!ELEMENT begin-end EMPTY>
<!ATTLIST begin-end regexp CDATA #REQUIRED>

<!ELEMENT begin EMPTY>
<!ATTLIST begin regexp CDATA #REQUIRED>

<!ELEMENT end EMPTY>
<!ATTLIST end regexp CDATA #REQUIRED>

<!ELEMENT replacement-rule (replace+)>
<!ATTLIST replacement-rule regexp CDATA #REQUIRED>

<!ELEMENT replace EMPTY>
<!ATTLIST replace source CDATA #REQUIRED
                  target CDATA #REQUIRED
                  prefer (yes|no) #IMPLIED>
```

Apéndice B

Símbolos gramaticales utilizados en los módulos

B.1. Símbolos gramaticales utilizados en los diccionarios

B.1.1. Lista de símbolos

aa	adjetivo-adjetivo (función antecedente relativo)
acr	acrónimo
al	otros
an	adjetivo-nombre (función antecedente relativo)
ant	antropónimo
cni	condicional
cnjadv	conjunción adverbial
cnjcoo	conjunción coordinativa
cnjsub	conjunción subordinativa
def	definido
dem	demostrativo
det	determinante
detnt	determinante neutro
enc	enclítico
f	femenino
fti	futuro de indicativo
fts	futuro de subjuntivo
ger	gerundio
ifi	pretérito perfecto o indefinido
ij	interjección
imp	imperativo
ind	indeterminado
inf	infinitivo

itg	interrogativo
loc	locativo
lpar	([
lquest	¿
m	masculino
mf	masculino-femenino
n	nombre
nn	nombre-nombre (función antecedente relativo)
np	nombre propio
nt	neutro
num	numeral
p1	primera persona
p2	segunda persona
p3	tercera persona
pii	pretérito imperfecto de indicativo
pis	pretérito imperfecto de subjuntivo
pl	plural
pos	posesivo
pp	participio
pr	preposición
preadv	preadverbio
predet	predeterminante
pri	presente de indicativo
prn	pronombre
pro	proclítico
prs	presente de subjuntivo
ref	reflexivo
rel	relativo
rpar	)]
sent	. ? ; : !
sg	singular
sp	singular-plural
sup	superlativo
tn	tónico
vaux	verbo auxiliar
vbhaver	verbo *haver*
vblex	verbo léxico
vbmod	verbo modal
vbser	verbo *ser*

B.1.2. Especificación de formas léxicas

Orden de colocación de las etiquetas en los diccionarios morfológicos de este sistema (orden de izquierda a derecha en la tabla):

Adjetivos generales (difícil, rojo)	**Categoría**	**Género**	**Número**	
	adj	m	sg	
		f	pl	
		mf	sp	
Adjetivos interrogativos, posesivos, indeterminados y superlativos (qué, tus, otra, buenísimo)	**Categoría**	**Tipo**	**Género**	**Número**
	adj	itg	m	sg
		pos	f	pl
		ind	mf	sp
		sup		
Adverbios (siempre, mañana)	**Categoría**			
	adv			
Preadverbios (muy, tan)	**Categoría**			
	preadv			
Adverbios interrogativos (dónde)	**Categoría**	**Tipo**		
	adv	itg		
Conjunciones adverbiales (que, así como)	**Categoría**			
	cnjadv			
	cnjcoo			
	cnjsub			
Determinantes (el, uno, este, mi)	**Categoría**	**Tipo**	**Género**	**Número**
	det	def	m	sg
		ind	f	pl
		dem	mf	sp
		pos		
Determinante neutros (lo)	**Categoría**			
	detnt			
Predeterminantes (todos)	**Categoría**	**Género**	**Número**	
	predet	m	sg	
		f	pl	
		nt	sp	
Interjecciones (hola)	**Categoría**			
	ij			
Nombres comunes (casa, perro)	**Categoría**	**Género**	**Número**	
	n	m	sg	
	n	f	pl	
	n	mf	sp	
Nombres propios (Pedro, Londres)	**Categoría**	**Tipo**		
	np	ant		
		loc		
		al		

Acrónimos (IRPF, INEM)	**Categoría**	**Tipo**	**Género**	**Número**	
	n	acr	m f mf	sg pl sp	
Numerales (tres)	**Categoría**	**Género**	**Número**		
	num	m f mf	sg pl sp		
Preposiciones (de, por)	**Categoría**				
	pr				
Pronombres interrogativos (quién, qué)	**Categoría**	**Tipo**	**Género**	**Número**	
	prn	itg	m f	sg pl	
Pronombres enclíticos, proclíticos y tónicos de persona (yo, vosotros, ayudar**te**, **te** ayudo)	**Categoría**	**Tipo**	**Persona**	**Género**	**Número**
	prn	enc pro tn	p1 p2 p3	m f mf nt	sg pl sp
Pron. procl. reflexivo (se):	prn	pro	ref	p3	mf sp
Pron. tón. reflexivo (si):	prn	tn	ref	p3	mf sp
Pron. tónicos posesivos (mío, suyo)	**Categoría**	**Tipo**	**Pos.**	**Género**	**Número**
	prn	tn	pos	m f	sg pl
Otros pron. tónicos (aquella, nadie, otro)	**Categoría**	**Tipo**	**Género**	**Número**	
	prn	tn	m f mf nt	sg pl sp	
Relativos pronominales y adjetivales (que, cuyo)	**Categoría**	**Tipo**	**Género**	**Número**	
	rel	nn an aa	m f f	sg pl pl	
Relativos adverbiales (como, donde)	**Categoría**	**Tipo**			
	rel	adv			
Verbos (formas personales) (subo, vamos)	**Tipo**	**Tiempo y modo**	**Persona**	**Número**	
	vblex vbser vbhaver vbmod	cni fti fts ifi imp pii pis pri prs	p1 p2 p3	sg pl	
Verbos (infinitivo y gerundio) (cantar, buscando)	**Tipo**	**Forma**			
	vblex vbser vbhaver vbmod	inf ger			
Verbos (participio) (dormido, cansadas)	**Tipo**	**Forma**	**Género**	**Número**	
	vblex vbser vbhaver vbmod	pp	m f	sg pl	

B.2. Categorías del desambiguador

B.2.1. Desambiguador de español

Estas son las categorías o etiquetas gruesas con las que trabaja el desambiguador léxico categorial del español.

Etiqueta	Descripción	Cerrada	Ejemplos
	Etiquetas simples		
PARAPR	Lexicalización *para* como preposición	Sí	
PARAVBPRI	Lexicalización *para* como verbo léxico en presente de indicativo	Sí	
PARAVBIMP	Lexicalización *para* como verbo léxico en imperativo	Sí	
QUECNJ	Lexicalización *que* como conjunción	Sí	
QUEREL	Lexicalización *que* como relativo	Sí	
COMOPR[1]	Lexicalización *como* como preposición	Sí	
COMOREL	Lexicalización *como* como relativo	Sí	
COMOVB	Lexicalización *como* como verbo léxico en presente de indicativo	Sí	
MASADV	Lexicalización *más/menos* como advervio	Sí	
MASADJ	Lexicalización *más/menos* como adjetivo	Sí	
MASNP	Lexicalización *Más* como nombre propio	Sí	
ALGO	Lexicalización *algo* como nombre propio	Sí	
ACRONIMOM	Acrónimo	No	BCH
ACRONIMOF	Acrónimo	No	ONUH
ACRONIMOMF	Acrónimo	No	ATS
INTNOM	Pronombre interrogativo	Sí	quién, cuál
ADJINT	Adjetivo interrogativo	Sí	cuánto, qué
INTADV	Adverbio interrogativo	Sí	cuándo, dónde
PREADV	Advervio que puede preceder a otro adverbio o adjetivo	Sí	muy, bien, mal
ADV	Adverbio	No	nunca, ahí
CNJSUBS	Conjunción subordinante	Sí	que
CNJCOORD	Conjunción coordinante	Sí	y, pero
CNJADV	Conjunción subordinante adverbial	No	si
DETNT	Determinante neutro	Sí	lo
DETM	Determinante	Sí	el, un
DETF	Determinante	Sí	la, una
DETMF	Determinante	Sí	cada
INTERJ	Interjección	No	ojalá, hola
NOM	Nombre	No	casa, coche
ANTROPONIM	Nombre propio de persona	No	Fernando

[1]El analizador morfológico considera que *como* puede ser una preposición ya que puede sustituirse por *en calidad de* en algunos contextos (Ej.- *'Os hablo como director de la película'*

Etiqueta	**Descripción**	**Cerrada**	**Ejemplos**
TOPONIM	Nombre propio de lugar	No	Alicante
NPALTRES	Otros nombre propios	No	Linux, Seat
NUM	Numeral	Sí	tres, cuatro
PREDETNT	Predetermiante neutro	Sí	todo
PREDET	Predeterminante Ej. todo	Sí	toda
PREP	Preposición	Sí	ante, desde
PRNTNNT	Pronombre tónico neutro	Sí	algo, esto
PRNTN	Pronombre tónico	Sí	ambos, nadie
PRNENCREF	Pronombre enclítico reflexivo	Sí	se
PRNPROREF	Pronombre proclítico reflexivo	Sí	se
PRNENC	Pronombre enclítico	Sí	me, nos
PRNPRO	Pronombre proclítico	Sí	le, te
VLEXINF	Verbo léxico en infinitivo	No	cantar, reír
VLEXGER	Verbo léxico en gerundio	No	hablando
VLEXPARTPI	Verbo léxico en participio	No	dicho, cantado
VLEXPFCI	Verbo léxico en presente, futuro o condicional de indicativo	No	digo, diré, diría
VLEXIPI	Verbo léxico en pretérito imperfecto o perfecto simple de indicativo	No	cantaba, dijo
VLEXSUBJ	Verbo léxico subjuntivo	No	hablase, dijeramos
VLEXIMP	Verbo léxico imperativo	No	canta, comed
VSERINF	Verbo ser en infinitivo	Sí	ser
VSERGER	Verbo ser en gerundio	Sí	siendo
VSERPARTPI	Verbo ser en participio	Sí	sido
VSERPFCI	Verbo ser en presente, futuro o condicional de indicativo	Sí	soy, seré, sería
VSERIPI	Verbo ser en pretérito imperfecto o perfecto simple de indicativo	Sí	era, fui
VSERSUBJ	Verbo ser subjuntivo	Sí	fueras
VSERIMP	Verbo ser imperativo	Sí	sé
VHABERINF	Verbo haber en infinitivo	Sí	haber
VHABERGER	Verbo haber en gerundio	Sí	habiendo
VHABERPARTPI	Verbo haber en participio	Sí	habido
VHABERPFCI	Verbo haber en presente, futuro o condicional de indicativo	Sí	hay, habrán, habría
VHABERIPI	Verbo haber en pretérito imperfecto o perfecto simple de indicativo	Sí	había, hubo
VHABERSUBJ	Verbo haber subjuntivo	Sí	hubieran
VMODALINF	Verbo modal en infinitivo	Sí	deber, poder
VMODALGER	Verbo modal en gerundio	Sí	debiendo
VMODALPARTPI	Verbo modal en participio	Sí	podido
VMODALPFCI	Verbo modal en presente, futuro o condicional de indicativo	Sí	puede, deberá, podría
VMODALIPI	Verbo modal en pretérito imperfecto o perfecto simple de indicativo	Sí	podía, debió
VMODALSUBJ	Verbo modal subjuntivo	Sí	pudiese, debiéramos

Etiqueta	Descripción	Cerrada	Ejemplos
VMODALIMP	Verbo modal imperativo	Sí	poded, debed
ADJM	Adjetivo	No	gracioso
ADJF	Adjetivo	No	graciosa
ADJMF	Adjetivo	No	inteligente
ADJPOS	Adjetivo posesivo	Sí	mío
REL	Relativo	Sí	quien, cuya
RELADV	Relativo adverbial	Sí	cuando, donde
Etiquetas compuestas			
PREPDET	Contracción de preposición y determinante	Sí	del, al
PRCNJ	Multipalabra formado por preposición y conjunción	Sí	a que
PRREL	Multipalabra formado por preposición y relativo	Sí	en que
INFLEXPRNENC	Verbo léxico en infinitivo con enclíticos	No	dármelo, cantarlo
GERLEXPRNENC	Verbo léxico en gerundio con enclíticos	No	cantándosela
IMPLEXPRNENC	Verbo léxico en imperativo con enclíticos	No	dímelo
INFSERPRNENC	Verbo ser en infinitivo con enclíticos	Sí	serlo
GERSERPRNENC	Verbo ser en gerundio con enclíticos	Sí	siéndolo
IMPSERPRNENC	Verbo ser en imperativo con enclíticos	Sí	sedlo
INFHABPRNENC	Verbo haber en infinitivo con enclíticos	Sí	habérsela
GERHABPRNENC	Verbo haber en gerundio con enclíticos	Sí	habiéndole
INFMODPRNENC	Verbo modal en infinitivo con enclíticos	Sí	poderla, deberlo
GERMODPRNENC	Verbo modal en gerundio con enclíticos	Sí	debiéndosela
IMPMODPRNENC	Verbo modal en imperativo con enclíticos	Sí	debédmela
Otras etiquetas			
LQUEST	Apertura de frase interrogativa		?
LPAR	Apertura paréntesis o corchete		(, [
RPAR	Cierre paréntesis o corchete		),]
CM	Coma		,
SENT	Finalizador de oraciones		., :, ;, ?, !

B.2.2. Desambiguador de catalán

Dada la similitud con las categorías del desambiguador de español, sólo se describen en la siguiente aquellas etiquetas que son nuevas o diferentes para el desambiguador de catalán.

Etiqueta	Descripción	Cerrada	Ejemplos
Etiquetas simples			
MOLTADV	Lexicalización *molt/gaire* como adverbio	Sí	
MOTLPREADV	Lexicalización *molt/gaire* como adverbio	Sí	
VOLERMOD	Lexicalización *voler* como verbo modal	Sí	
VOLERLEX	Lexicalización *voler* como verbo léxico	Sí	

Etiqueta	Descripción	Cerrada	Ejemplos
VA	Lexicalización *va* como forma del verbo *anar*	Sí	

B.2.3. Desambiguador de gallego

Dada la similitud con las categorías del desambiguador de español, sólo se describen en la siguiente aquellas etiquetas que son nuevas o diferentes para el desambiguador de gallego.

Etiqueta	Descripción	Cerrada	Ejemplos
	Etiquetas simples		
VBIRNPS	Lexicalización *ir* como infinitivo y gerundio	Sí	
VBIRPARTPI	Lexicalización *ir* como participio	Sí	
VBIRPS	Lexicalización *ir* como formas personales de indicativo y subjuntivo	Sí	
VBIRIMP	Lexicalización *ir* como imperativo	Sí	
VHABERNPS	Lexicalización *haber* como infinitivo y gerundio	Sí	
VHABERPARTPI	Lexicalización *haber* como participio	Sí	
VHABERPS	Lexicalización *haber* como formas personales de indicativo y subjuntivo	Sí	
VHABERIMP	Lexicalización *haber* como imperativo	Sí	
APREP	Lexicalización *a* como preposición *anar*	Sí	
VLEXNPS	Verbo léxico: infinitivo y gerundio	No	achegar, achegándomos
VLEXPS	Verbo léxico: formas personales de indicativo	No	achegue, achegaré
VSERNPS	Verbo ser: infinitivo y gerundio	Sí	ser, seres
VSERPS	Verbo ser: formas personales de indicativo	Sí	fosen, es
VHABERNPS	Verbo haber: infinitivo y gerundio	Sí	haber
VHABERPS	Verbo haber: formas personales de indicativo	Sí	achegue, achegaré
	Etiquetas compuestas		
PREPDETM	Contracción de preposición y determinante masculino	Sí	do, ao
PREPDETF	Contracción de preposición y determinante femenino	Sí	da, ás
PREPDETN	Contracción de preposición y determinante neutro	Sí	do
PREPDETDET	Contracción de preposición y dos determinantes	Sí	destoutro
PREPPRTNNT	Contracción de preposición y		

Etiqueta	Descripción	Cerrada	Ejemplos
	pronombre tónico neutro	Sí	daquilo
PREPPRNTN	Contracción de preposición y pronombre tónico	Sí	daqueloutra
PREPTNTN	Contracción de preposición y dos pronombres tónicos	Sí	nestoutra
PREPNUM	Contracción de preposición y numeral	Sí	dunha
PREDETDET determinante	Contracción de predeterminante y Sí	tódalas	
INTADVDET determinante	Contracción de interrogativo adverbial y Sí	u-la	
DETDETM	Contracción de dos determinantes masculinos	Sí	ámbolos
DETDETF	Contracción de dos determinantes femeninos	Sí	ámbalas
PRNPRN	Contracción de dos pronombres tónicos	Sí	esoutra
PRNPRN	Contracción de dos pronombres proclíticos	Sí	chas
CNJCDET	Contracción de conjunción coordinada y determinante	Sí	maila
CNJSUB determinante	Contracción de conjunción subordinada y Sí	cás	
INFSERPRNENC	Verbo ser en infinitivo con enclíticos	Sí	serlo
GERSERPRNENC	Verbo ser en gerundio con enclíticos	Sí	siéndolo
IMPSERPRNENC	Verbo ser en imperativo con enclíticos	Sí	sedlo
INFHABPRNENC	Verbo haber en infinitivo con enclíticos	Sí	habérsela
GERHABPRNENC	Verbo haber en gerundio con enclíticos	Sí	habiéndole
INFMODPRNENC	Verbo modal en infinitivo con enclíticos	Sí	poderla, deberlo
GERMODPRNENC	Verbo modal en gerundio con enclíticos	Sí	debiéndosela
IMPMODPRNENC	Verbo modal en imperativo con enclíticos	Sí	debédmela

Apéndice C

Abreviaturas usadas en el texto

ANSI *American National Standards Institute*, instituto nacional norteamericano de estándares; cuando se usa informalmente en la expresión *texto ANSI*, se refiere a un texto codificado en alguna de las codificaciones de un byte por carácter definidas en el estándar ISO-8859 [1].

ca Código ISO 639 de dos letras[1] del catalán

DTD Definición del tipo de documento en XML

es Código ISO 639 de dos letras del español

eu Código ISO 639 de dos letras del euskera

FL Forma léxica (véase la página 7)

FLLM Forma léxica de la lengua meta

FLLO Forma léxica de la lengua origen

FS Forma superficial (véase la página 7)

gl Código ISO 639 de dos letras del gallego

HTML *Hypertext markup language*, lenguaje de marcas para hipertextos.

LM Lengua meta (lengua de llegada)

LO Lengua origen (lengua de partida)

RTF *Rich text format*, formato de texto enriquecido.

[1] Véase `http://www.w3.org/WAI/ER/IG/ert/iso639.htm`

TA Traducción automática

XML *Extensible markup language*, lenguaje de marcas extensible.

Bibliografía

[1] Unicode. `http://www.unicode.org`.

[2] R. Canals-Marote, A. Esteve-Guillén, A. Garrido-Alenda, M. Guardiola-Savall, A. Iturraspe-Bellver, S. Monserrat-Buendia, S. Ortiz-Rojas, H. Pastor-Pina, P.M. Perez-Antón, and M.L. Forcada. El sistema de traducción automática castellano-catalán interNOSTRUM. *Procesamiento del Lenguaje Natural*, 27:151–156, 2001. XVII Congreso de la Sociedad Española de Procesamiento del Lenguaje Natural, Jaén, Spain, 12-14.09.2001.

[3] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. In *Third Conference on Applied Natural Language Processing. Association for Computational Linguistics. Proceedings of the Conference.*, pages 133–140, Trento, Italia, 31 marzo–3 abril 1992.

[4] A. Garrido, A. Iturraspe, S. Montserrat, H. Pastor, and M. L. Forcada. A compiler for morphological analysers and generators based on finite-state transducers. *Procesamiento del Lenguaje Natural*, (25):93–98, 1999.

[5] Alicia Garrido-Alenda and Mikel L. Forcada. Morphtrans: un lenguaje y un compilador para especificar y generar módulos de transferencia morfológica para sistemas de traducción automática. *Procesamiento del Lenguaje Natural*, 27:157–164, 2001.

[6] Alicia Garrido-Alenda, Mikel L. Forcada, and Rafael C. Carrasco. Incremental construction and maintenance of morphological analysers based on augmented letter transducers. In *Proceedings of TMI 2002 (Theoretical and Methodological Issues in Machine Translation, Keihanna/Kyoto, Japan, March 2002)*, pages 53–62, 2002.

[7] Alicia Garrido-Alenda, Patrícia Gilabert Zarco, Juan Antonio Pérez-Ortiz, Antonio Pertusa-Ibáñez, Gema Ramírez-Sánchez, Felipe

Sánchez-Martínez, Miriam A. Scalco, and Mikel L. Forcada. Shallow parsing for portuguese-spanish machine translation. In A. Branco, A. Mendes, and R. Ribeiro, editors, *TASHA 2003: Workshop on Tagging and Shallow Processing of Portuguese*, pages 21–24, October 2003.

[8] N. Ide. *The XML Framework and Its Implications for the Development of Natural Language Processing Tools*. Luxembourg, 2000.

[9] M.E. Lesk. Lex — a lexical analyzer generator. Technical Report Technical Report 39, AT&T Bell Laboratories, Murray Hill, N.J., 1975.

[10] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[11] Sergio Ortiz-Rojas, Mikel L. Forcada, and Gema Ramírez-Sánchez. Construcción y minimización eficiente de transductores de letras a partir de diccionarios con paradigmas. *Procesamiento del Lenguaje Natural*, (25):51–57, 2005.

[12] Ferran Pla and Antonio Molina. Improving part-of-speech tagging using lexicalized HMMs. *Journal of Natural Language Engineering*, 10(2):167–189, June 2004.

[13] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[14] E. Roche and Y. Schabes. *Introduction*. MIT Press, Cambridge, Massachusetts, 1997.

[15] E. Roche and Y. Schabes. Introduction. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing*, pages 1–65. MIT Press, Cambridge, Mass., 1997.

[16] J. L. A. van de Snepscheut. *What computing is all about*. Springer-Verlag, New York, 1993.

[17] Patrícia Gilabert Zarco, Javier Herrero-Vicente, Sergio Ortiz-Rojas, Antonio Pertusa-Ibáñez, Gema Ramírez-Sánchez, Felipe Sánchez-Martínez, Marcial Samper-Asensio, Míriam A. Scalco, and Mikel L. Forcada. Construcción rápida de un sistema de traducción automática español-portugués partiendo de un sistema español–catalán. *Procesamiento del Lenguaje Natural*, (32):279–285, 2003. (Actas del XIX congreso de la Sociedad Española de Procesamiento del Lenguaje Natural).