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

BORRADOR

AUTORES:
Mikel L. Forcada
Mireia Ginestí Rosell
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

29 de julio de 2005

Índice general

Introducción

La presente documentación describe 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". En concreto, describe los datos correspondientes a la arquitectura de transferencia sintáctica superficial que se usará para los pares español–gallego, gallego–español, español–catalán[1] y catalán–español, aunque algunos aspectos de la especificación son también relevantes para la arquitectura de transferencia sintáctica profunda que se desarrolla en el mismo proyecto para el par español–euskera.

La documentación se estructura en diferentes capítulos, organizados como sigue:

- capítulo 1: **descripción general del sistema** de traducción automática de transferencia superficial y de los módulos de los que se compone.
- capítulo 2: especificación del **formato del flujo de datos entre módulos**, es decir, el formato 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)

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

  - 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 y el compilador correspondiente (apartado 3.2.4)
  - sección 3.3: *Módulo de transferencia estructural*, en la que se describe el funcionamiento del módulo (apartado 3.3.1), el formato de las reglas de transferencia estructural (apartado 3.3.2) y el compilador correspondiente (apartado 3.3.3)
  - sección 3.4: *Desformateador y reformateador*, en la que se describe el funcionamiento de estos módulos (apartado 3.4.1), las reglas de especificación de formato (apartado 3.4.2) y el proceso de generación de los módulos (apartado 3.4.3)
  - sección 3.5: *Módulos auxiliares*, en la que se describe el módulo de preproceso del transfer, que realiza ciertas operaciones en las unidades multipalabra

- capítulo 4: descripción de los **datos lingüísticos existentes** en este proyecto, es decir, de los diccionarios, los datos de desambiguación léxica categorial y las reglas de transferencia estructural que se han creado para el español, el catalán y el gallego

- capítulo 5: descripción de la **interfaz web** del sistema de traducción automática

- capítulo 6: descripción de la **instalación del sistema** y de los requisitos para ello

- capítulo 7: ideas de **trabajo futuro**, trabajo que ha quedado pendiente y que se puede abordar una vez finalizado el presente proyecto

Los ficheros a los que hace referencia este informe se citarán por su URI, de manera que estarán accesibles, como este informe, en la ubicación web provisional `http://www.torsimany.ua.es/esgleuca/`.

El presente documento tiene algunas secciones que están incompletas o no han sido escritas todavía.

Finalmente, debe tenerse en cuenta que los formatos descritos en el presente documento no deben considerarse como definitivos, ya que pueden ser modificados o adaptados como consecuencia del desarrollo del resto del proyecto.

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, clave FIT-340101-2004-3.
- 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 Alegría, 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, 6, 5] y español–portugués Traductor Universia [7, 11], 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 presentado sigue estas directrices con resultados muy interesantes.

El ingenio de traducción consiste en una *cadena de montaje* de ocho módulos, seis de los cuales se compilan automáticamente a partir de ficheros con los correspondientes datos lingüísticos. Los módulos, cuya organización se muestra en la figura 1.1, son:

- El *desformateador*, que separa el texto a traducir de la información de formato; su especificación se describe en el apartado 3.4.1.

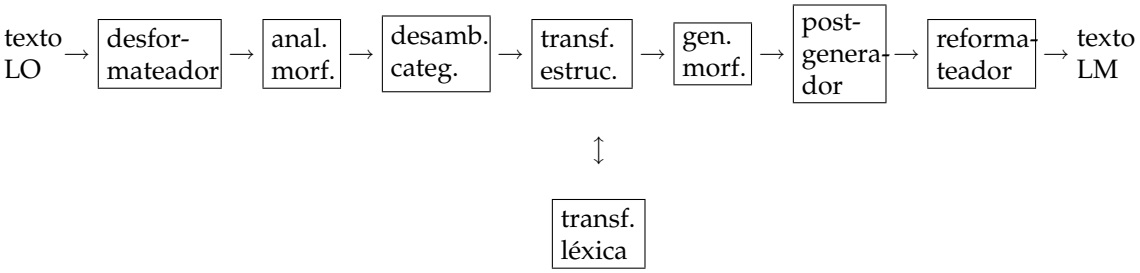

Figura 1.1: Los ocho módulos que forman la cadena de montaje del sistema de traducción automática por transferencia sintáctica superficial.

- 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. [1] Este módulo se genera a partir de un diccionario morfológico (cuyo formato, basado en el descrito en [5], se especifica en el apartado 3.1.2) para la lengua origen (LO) usando un compilador de analizadores morfológicos.

- El *desambiguador léxico categorial*, elige, usando un modelo estadístico (modelo oculto de Markov), una de las FL de una FS ambigua de acuerdo con su contexto; las FS 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 FS) y son una fuente muy importante de errores de traducción en caso de elegir el equivalente incorrecto (por haber elegido la FL incorrecta). El modelo estadístico se entrena sobre corpus suficientemente representativos de textos en LO. La especificación del desambiguador

[1] Para más detalles sobre la estructura de los datos entregados por el analizador morfológico, véase el final de la sección 3.2.2.1. Las unidades léxicas de más de una palabra se discuten en detalle en el apartado 3.1.2.5.

léxico categorial se describe en la sección 3.2.

- 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. La especificación del formato de este archivo, que se inspira en la descrita en [6], se encuentra en la sección 3.3.

- 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). Este módulo se genera a partir de un diccionario bilingüe (descrito en el apartado 3.1.2) usando un compilador de módulos de transferencia léxica (descrito en el capítulo 3.1.3).

- El *generador morfológico*, que genera a partir de la FL en LM una forma superficial adecuada en LM, flexionándola adecuadamente. Este módulo se genera a partir de un diccionario morfológico para la LM mediante un compilador de generadores morfológicos. En el apartado 3.1.2 se encuentra la especificación del formato de los diccionarios morfológicos, y en el capítulo 3.1.3 se describen los compiladores correspondientes.

- El *postgenerador*, que realiza algunas operaciones ortográficas en LM tales como contracciones y apostrofaciones, y que es generado por un compilador a partir de reglas de transformación en un formato similar al de los diccionarios anteriores. El formato de los diccionarios de postgeneración se especifica en la sección 3.1.2 y el del compilador, en 3.1.3.

- El *reformateador*, que reintegra la información de formato original al texto traducido; su especificación se describe en la sección 3.4.1.

Cuatro de estos módulos, a saber, el analizador morfológico, el módulo de transferencia léxica (diccionario bilingüe), el generador morfológico y el postgenerador, están basados en *transductores de estados finitos* [5].

Capítulo 2

Especificación del formato del flujo de datos entre módulos

2.1. Introducción

Para entender 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 modificarlo para comprobar el resultado. Es posible comprobar de manera independiente la salida de cada módulo del traductor, para detectar así los puntos en que se producen los errores, ya sean lingüísticos o de sistema.

Asimismo, un flujo basado en texto facilita la construcción de prototipos de un traductor con módulos no totalmente integrados entre sí, ya que es posible modificar el flujo de datos que circula de un módulo al otro para adecuarlo al formato específico de cada módulo.

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*, mar-

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

cas que encapsulan el formato (véase el apartado 3.4.1) 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 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 .

El objetivo es que los módulos sean capaces de tratar dos tipos de flujos de datos: uno que consiste en texto llano con marcas de carácter con significado especial —orientado al procesamiento en servidores de internet que traduzcan grandes volumenes de texto— y otro en un formato XML diseñado para mejorar la interoperabilidad entre módulos con vistas a que puedan ser usados para otros fines o para el diagnóstico de errores. A continuación describiremos las características de ambos tipos de flujo.

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 3.4.1 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 las dos secciones que se presentan en este apartado 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 que se considere traducible.

- *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 no entrar en conflicto con los usados en el propio formato del flujo de datos.

2.2.1. Formato XML

En este tipo de flujo se usa el elemento **<b>** para definir los superblancos y los superblancos extensos. Para el caso de los **superblancos** la sintaxis es la siguiente:

```
<b>contenido del bloque de formato</b>
```

Hay que resaltar que para los formatos basados en SGML, es necesario incluir el formato en bloques `<![CDATA[...]]>` dentro de las marcas indicadas. Por su parte, los *superblancos extensos* se deben expresar, a modo de atributos, de la siguiente manera:

```
<b filename="nombre de fichero"/>
```

El *texto* estará incluido entre los elementos **b** que se acaban de explicar sin ninguna marca de estructura particular.

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

```
.<b/>
```

Resumiendo, el flujo de datos de un documento en cualquier formato de los que trata el traductor se reduce a otro documento XML que debe cumplir la siguiente DTD:

```
<!ELEMENT document (b|#PCDATA)*>
<!ELEMENT b (#PCDATA?)>
<!ATTLIST b filename CDATA #IMPLIED>
```

El resultado de encapsular el formato del fichero de la figura 2.2 en el flujo con formato XML se ve en la figura 2.3. Si hubiese algún superblanco que por su longitud se convirtiese en un superblanco extenso, la forma de especificarlo sería como sigue:

```
<b filename="/tmp/ficherotemporal"/>
```

donde `/tmp/ficherotemporal"` es un fichero que contiene el superblanco extenso para que pueda ser recuperado por el reformateador.

```
<?xml version="1.0" encoding="iso-8859-15"?>
<document>
<b><![CDATA[<html>
  <head>
    <title>]]></b>Título.<b/><b><![CDATA[</title>
  </head>
  <body>
    <p>]]></b>Frase<b><![CDATA[
       ]]></b>dividida.<b/><b><![CDATA[
  </body>
</html>]]></b>
</document>
```

Figura 2.3: El documento de la figura 2.2 con el formato encapsulado usando marcas en XML y segmentos `<![CDATA[...]]>`

2.2.2. Formato no XML

Este formato se basa en el formato utilizado en los sistemas de traducción interNOSTRUM [2, 6, 5] y Traductor Universia [7, 11].

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*, como en el caso del flujo XML, 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
Barra invertida	\	\\	Carácter de protección
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 2.4 muestra cómo quedaría encapsulado el documento de la figura 2.2 según el tipo de flujo en formato no XML.

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

Figura 2.4: El documento de la figura 2.2 con el formato encapsulado usando marcas no XML

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:

- *Flujo segmentado ambiguo*. Como características presenta 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 3.1 de la página 46 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 3.2 de la página 50 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.

2.3.1. Formato XML

Las *palabras* se etiquetan de la forma que se muestra a continuación:

```
<w>información de la palabra</w>
```

Para el caso del *flujo de datos segmentado ambiguo*, la *forma superficial* se indica en el interior de un elemento **<w>** mediante el contenido de un único elemento **<sf>**. A continuación, se sitúan la forma o *formas léxicas* que sean necesarias:

```
<w>
  <sf>forma superficial</sf>
  <lf>forma léxica 1</lf>
  <lf>forma léxica 2 (opcional)</lf>
  ...
</w>
```

Para el caso del flujo no ambiguo, sólo se especifica una única forma léxica.

```
<w>
  <lf>forma léxica</lf>
</w>
```

La DTD de este flujo de datos para textos *sin desambiguar* es la que se muestra en la figura 2.5 a continuación.

```
<!ELEMENT document (b|w|#PCDATA)*>
<!-- atención, el #PCDATA anterior sigue siendo necesario para los
     carácteres no etiquetados y que no forman parte del formato -->
<!ELEMENT b (#PCDATA?)>
<!ATTLIST b filename CDATA #IMPLIED>
<!ELEMENT w (sf,lf+)>
<!ELEMENT sf (#PCDATA)>
<!ELEMENT lf (#PCDATA|s)+>
<!ELEMENT s EMPTY>
<!ATTLIST s n IDREF #REQUIRED>
```

Figura 2.5: DTD para textos no desambiguados con formato XML

Para los ya *desambiguados,* los textos deben cumplir la DTD de la figura 2.6.

```
<!ELEMENT document (b|w|#PCDATA)*>
<!-- atención, el #PCDATA anterior sigue siendo necesario para los
     carácteres no etiquetados y que no forman parte del formato -->
<!ELEMENT b (#PCDATA?)>
<!ATTLIST b filename CDATA #IMPLIED>
<!ELEMENT w (lf)>
<!ELEMENT lf (#PCDATA|s)+>
<!ELEMENT s EMPTY>
<!ATTLIST s n IDREF #REQUIRED>
```

Figura 2.6: DTD para textos desambiguados con formato XML

La figura 2.7 muestra un ejemplo de segmentación del flujo que incluye la forma de encapsular el formato y la información léxica. Este ejemplo es para el caso de flujo segmentado ambiguo y corresponde al texto HTML original de la figura 2.2.

```
<?xml version="1.0" encoding="iso-8859-15"?>
<document>
<b><![CDATA[<html>
  <head>
    <title>]]></b>
<w>
  <sf>Título<sf>
  <lf>Título<s n="n"/><s n="m"/><s n="sg"/></lf>
</w>
<w>
  <sf>.</sf>
  <lf>.<s n="sent"/></lf>
</w><b/>
<b><![CDATA[</title>
  </head>
  <body>
    <p>]]></b>
<w>
  <sf>Frase</sf>
  <lf>Frase<s n="n"/><s n="f"/><s n="sg"/></lf>
</w>
<b><![CDATA[
       ]]></b>
<w>
  <sf>dividida</sf>
  <lf>dividir<s n="vblex"/><s n="pp"/><s n="f"/><s n="sg"/></lf>
</w>
<w>
  <sf>.</sf>
  <lf>.<s n="sent"/></lf>
</w><b/>
<b><![CDATA[
  </body>
<html>]]></b>
</document>
```

Figura 2.7: Ejemplo de flujo segmentado con el formato encapsulado en XML, correspondiente al documento HTML de la figura 2.2.

```
[<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 2.8: Ejemplo de flujo segmentado con el formato encapsulado en formato no XML, correspondiente al documento HTML de la figura 2.2.

2.3.2. Formato no XML

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 2.8.

Capítulo 3

Especificación de los módulos

3.1. Módulos de procesamiento léxico

3.1.1. Descripción del funcionamiento

[PENDIENTE]

3.1.1.1. Tratamiento de mayúsculas y minúsculas en los diccionarios

[PENDIENTE]

3.1.2. Formato de los datos: los diccionarios

3.1.2.1. 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. De este modo, el mismo formato sirve para especificar diccionarios morfológicos (monolingües) y diccionarios bilingües.

Otro criterio importante que se ha seguido en el diseño es que los diccionarios tienen que compilarse para elaborar *transductores de letras* con ellos, por cuestiones de eficiencia.

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, sino que puede referirse a un paradigma de flexión existente que le ahorrará gran parte del trabajo. Además, el uso de paradigmas reduce los requisitos de memoria, favorece la construcción de transductores de letras eficientes y aumenta la velocidad de compilación. 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 el apartado 3.1.2.4.2.

Se ha elegido que el formato de los diccionarios 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.

3.1.2.2. 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 transformacio-

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

nes 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 —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 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 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 3.1.2.3.5). 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 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 **diccionario 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 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:

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

3.1.2.3. 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 los apartados 3.1.2.4, 3.1.2.4.2 y 3.1.2.4.3 se describen las características particulares de las entradas de los tres tipos de diccionarios (morfológicos, bilingües y de postgeneración respectivamente).

3.1.2.3.1. 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.

3.1.2.3.2. 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.

3.1.2.3.3. 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.

```
<?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**>

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

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

3.1.2.3.4. 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.

3.1.2.3.5. 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

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

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

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.

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.

3.1.2.3.6. 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 (secciones 3.1.2.3.7,

3.1.2.3.9, 3.1.2.3.12 y 3.1.2.3.13 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 el apartado 3.1.2.3.15).

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 el apartado 3.1.2.4.2).

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

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

Figura 3.4: Uso del elemento <p>

a paradigmas,[2] no suele ser necesario usarlo.

3.1.2.3.7. 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 caracteres, correspondencia que especifica una transformación léxica que será realizada por un camino de estados en el transductor de estados finitos 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 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.

3.1.2.3.8. 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 habitualmente 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*,

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

```
[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

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 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 suele indicarse únicamente el primer símbolo 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>**).

3.1.2.3.9. 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.

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

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

3.1.2.3.10. 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 en la figura 3.6.

3.1.2.3.11. 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**>, es decir, tiene la misma estructura que una sección del diccionario <**section**>.

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 sección 3.1.2.3.14); 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 (apartado 3.1.2.4.2).

3.1.2.3.12. 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.

3.1.2.3.13. 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.

3.1.2.3.14. 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 el apartado 3.1.2.5). 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 3.4.1).

3.1.2.3.15. 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 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/> 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 el apartado 3.1.2.4.3 para más detalles). La restricción RL indica que esta entrada es sólo de generación, ya que no tiene ningún sentido para el análisis.

3.1.2.3.16. 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 palabra por palabra. Se utiliza en las multipalabras con flexión intercalada para señalar el principio y el final del lema o lemas sin flexión que van unidos a la palabra flexionada y que, junto a ella, forman una unidad inseparable. En el apartado 3.1.2.5 se explican con más detalle los distintos tipos de multipalabra, y en la figura 3.21 del mismo puede verse un ejemplo de utilización de este elemento.

3.1.2.3.17. 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 3.5.1), 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 esta acción (véase el apartado 3.3.2). Si, por el contrario, estas formas deben ser sólo de análisi, la entrada debe tener la restricción `LR`.

En la sección 3.1.2.5 se ilustra más detalladamente el uso de este elemento. Puede verse un ejemplo de uso en la figura 3.20 de la mencionada sección.

3.1.2.4. 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.

3.1.2.4.1. Diccionarios morfológicos

En estos diccionarios, que se usan para construir los analizadores y los generadores morfológicos, se deben marcar mediante `<a/>` las palabras 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 del apartado 3.1.2.3.15 puede verse un ejemplo de entrada con este elemento.

3.1.2.4.2. Diccionarios bilingües

Como ya hemos explicado, en nuestro sistema no 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 3.3 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*).

3.1.2.4.3. 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 el apartado 3.1.2.3.15) 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*. 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 entrada `"<a/>de<b/>el<b/>"`

```
<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

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.

3.1.2.5. 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 el apartado 3.1.2.3.17. 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 3.5.1) 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

```
<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.

quien debe decidir 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 3.3), 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/>`.

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 3.21. Para entender este ejemplo, 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/>`.

```
<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 3.21: Ejemplo de entrada en el diccionario morfológico que contiene un grupo **<g>**.

El uso del elemento **<g>** se debe a la necesidad de reconstituir estos lemas multipalabra que están *partidos* porque se flexionan en medio; por ejemplo, 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 3.5.1) que se ejecuta antes del módulo de transferencia estructural

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 3.21, el elemento **<g>** elemento sirve para diferenciar el grupo '`<b/>de<b/>menos`' en el diccionario morfológico. En el diccionario bilingüe (véase la figura 3.22), el elemento **<g>** sirve para establecer una correspondencia entre los grupos "`<b/>de<b/>menos`" y "`<b/>a<b/>faltar`".

3.1.3. Generación automática de los módulos de procesamiento léxico

[PENDIENTE]

```
<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 3.22: Ejemplo de entrada en el diccionario bilingüe que contiene dos grupos **<g>** en correspondencia.

3.2. Desambiguador léxico categorial

3.2.1. Descripción del funcionamiento

El desambiguador léxico categorial está basado en modelos ocultos de Markov [10] 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 [4], esto es, conjuntos de categorías gramaticales.

A pesar de trabajar con naturaleza 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 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. Hay dos tipos de lexicalizaciones, las que añaden nuevos observables y las que además añaden nuevos estados al modelo de Markov [9]; el desambiguador léxico categorial que se presenta 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>`, (en formato no XML, véase el capítulo 2), se agrupan bajo la categoría `VLEXSUBJ` (*verbo subjuntivo*). En este caso, una de las dos etiquetas finas es descartada; el fichero permite elegir 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

3.2.2.1. Introducción

A continuación se describe el formato de los ficheros en que se especifica cómo agrupar las etiquetas *finas* proporcionadas por el analizador morfológico en etiquetas *gruesas* más generales. En dicho fichero, además, se pueden especificar *restricciones* que ayuden a la estimación del modelo estadístico encargado del proceso de desambiguación léxica.

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:

$$\begin{aligned} \text{formaanalizada} &\rightarrow \text{multiformaléxica} \; [\; \text{multiformaléxica} \;]^* \\ \text{multiformaléxica} &\rightarrow \text{formaléxica} \; [\; \text{formaléxica} \;]^* \; \text{cola-lema?} \\ \text{formaléxica} &\rightarrow \text{lema etiquetafina} \\ \text{cola-lema} &\rightarrow \text{lema} \\ \text{etiquetafina} &\rightarrow \text{símbologram} \; [\; \text{símbologram} \;]^* \end{aligned} \tag{3.1}$$

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 el apartado 3.1.2.5 para más información).
- *etiquetafina* corresponde a uno o más símbolos gramaticales (*símbolo-gram*).

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*.

Para poder tratar con cada posible flujo de datos, es decir, para separar multiformas léxicas, formas léxicas, lemas y símbolos gramaticales, el desambiguador recibe un parámetro que especifica el formato del flujo de datos que está siendo utilizado (en el capítulo 2 se especifican en detalle los dos posibles formatos para el flujo).

3.2.2.2. Especificación del formato

El formato del fichero (codificado en XML) se especifica mediante la DTD que se observa en el apéndice A.2 y que se puede descargar de la dirección `http://www.torsimany.ua.es/esgleuca/tagger.dtd`.

El significado de las etiquetas es como sigue:

[4] Separadas entre ellas por un delimitador que se corresponde con el elemento `<j/>` (ver sección 3.1.2.3.17).

[5] Separados entre ellos mediante el elemento `<b/>` (ver sección 3.1.2.3.14).

`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 secuencias 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. En la versión actual, al generar el desambiguador, hacerlo al revés produce un error y la pérdida de la categoría más específica.

`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. Por simplicidad se asume que tras la secuencia definida puede venir un número indeterminado de símbolos gramaticales. 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, elipsis (en el caso del euskara), 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`. Se estudiará la posibilidad de incluir restricciones que abarquen más de dos categorías sin necesidad de utilizar modelos de Markov de orden mayor.

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`.

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 3.23 y 3.24 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 3.2 (puede leerse su descripción en la página 46). 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*).

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

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 3.3 (una *formadesambiguada* consistirá siempre en una única *multiformaléxica*).

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

```
<?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 3.23: Ejemplo de fichero de definición del desambiguador (continúa en la figura 3.24).

```
<!-- ... -->
<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 3.24: Ejemplo de fichero de definición del desambiguador (viene de la figura 3.23).

Por último, para el entrenamiento también se precisa del diccionario de la lengua implicada. Este diccionario se emplea para determinar conjuntamente con la especificación del etiquetario a emplear las distintas clases de ambigüedad con las que trabaja el desambiguador.

3.2.4. Generación automática del desambiguador

[PENDIENTE]

3.3. Módulo de transferencia estructural

3.3.1. 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 [6] y usado por los sistemas de TA español–catalán interNOSTRUM [2, 6, 5] y español–portugués Traductor Universia [7, 11], 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 6) 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 palabras 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 (flujo no XML, véase el capítulo 2)[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

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): *enviou-me* (gl) → *me envió* (es); *para nos dicir* (gl) → *para decirnos* (es);

Las *multipalabras* (sus diferentes tipos están explicados en 3.1.2.5) 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 no

XML), el módulo auxiliar `pretransfer` (véase 3.5.1), 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 3.3.2.38).

- 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 el apartado 3.3.2.37 para obtener una explicación más detallada del uso de este elemento.

3.3.2. 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 38).

- 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.

3.3.2.1. Elemento `<transfer>`

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

3.3.2.2. 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`**>.

3.3.2.3. 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.

3.3.2.4. Elemento de categoría `<cat-item>`

A lo largo de este documento hemos visto hasta ahora dos maneras de definir categorías gramaticales según las distintas fases del proceso de traducción:

- etiquetas finas asociadas a los lemas de las formas superficiales de la LO entregadas por el analizador mofológico
- agrupación de etiquetas finas en etiquetas más gruesas para el desambiguador categorial

Pues bien, en el módulo de transferencia se vuelven a agrupar las etiquetas finas del analizador morfológico en categorías que se usan para definir los patrones a detectar y procesar. Cada elemento usado para definir las categorías que usa el módulo de transferencia tiene un atributo obligatorio `tags` cuyo valor es una secuencia de nombres de etiqueta separados por puntos; esta secuencia es una subsecuencia de la etiqueta fina (ver definición en 3.2.2.1), es decir, de la secuencia de símbolos gramaticales que define cada posible forma léxica entregada por el desambiguador léxico categorial. Así, una categoría representa a un determinado conjunto de formas léxicas. 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,

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

<def-cat n="molt"/>
  <cat-item lemma="molt" tags="preadv"/>
  <cat-item lemma="molt" tags="adv"/>
  <cat-item lemma="molt" tags="adj.m.sg"/>
</def-cat>
```

Figura 3.25: 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 (*molt*, que equivale a *muy/mucho* en español y tiene más de un símbolo gramatical asociado, en este caso preadverbio, adverbio y adjetivo masculino singular).

`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. Además, puede utilizarse un atributo opcional, `lemma`, para definir formas léxicas a partir de su lema (ver figura 3.25).

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

En esta sección se definen los atributos morfológicos de las categorías definidas que se identificarán en las formas léxicas detectadas, cada uno con una etiqueta **<def-attr>**.

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

Cada **<def-attr>** define un atributo con información morfológica (género, nombre, persona, tiempo, etc.) mediante una lista de elementos de atributo de categoría (**<attr-item>**) y tiene un nombre único obligatorio `n`.

```
<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 3.26: 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*.

3.3.2.7. 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`. Puede ser una subsecuencia de las etiquetas de esta forma léxica (indicada en el atributo `tags` y separadas por punto) o una subcadena del lema de la forma léxica (indicada en el atributo `lemma`). Pueden especificarse ambos tipos de atributo simultáneamente (`lemma` y `tags`). En la figura 3.26 puede verse un ejemplo de utilización de este elemento para el atributo de *número* y el atributo de *nombre*.

3.3.2.8. 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).

3.3.2.9. 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.

3.3.2.10. 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.

3.3.2.11. Elemento de definición de listas de cadenas <def-list>

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

3.3.2.12. 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.

3.3.2.13. 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.

3.3.2.14. 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.

3.3.2.15. Elemento de sección de reglas <section-rules>

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

3.3.2.16. Elemento de regla <rule>

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

3.3.2.17. 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.

3.3.2.18. 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 3.32).

3.3.2.19. 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,

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

Figura 3.27: 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).

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).

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

3.3.2.21. 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 3.27.

3.3.2.22. 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.

3.3.2.23. Elemento de condición <when>

Con este elemento se describe una opción condicional (véase 3.3.2.22). 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.

3.3.2.24. 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>.

3.3.2.25. 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>).

3.3.2.26. 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 3.32
- 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 3.30 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 3.30.

- 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 3.28 y 3.29. 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 comparación) en una lista definida en la sección correspondiente (**<section-def-lists**), a la que se hace referencia mediante el 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.

3.3.2.27. 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 3.28):

- `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: `lemma` (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).

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

Figura 3.28: 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 3.29: Uso del elemento **<lit-tag>**: se comprueba si la etiqueta de número (`nbr`) de la segunda unidad léxica de la LM (`tl`) es `ND` (número por determinar)

3.3.2.28. 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*.

3.3.2.29. 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 3.29.

3.3.2.30. 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>**, representa la referencia de la variable y se puede cambiar su valor .

```
<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 3.30: 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.

3.3.2.31. 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>`.

3.3.2.32. 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 3.31, 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*.

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

Sirve para obtener el patrón de mayúsculas/minúsculas, es decir, 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 3.3.2.27. En la figura 3.31 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.

3.3.2.34. 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 la segunda letra en minúsculas, "aa" para poner todo en minúsculas y "AA" para poner todo en mayúsculas. En la figura 3.31 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 el apartado 3.1.1.1 para más información sobre la gestión de mayúsculas y minúsculas en los diccionarios).

```
<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 3.31: 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*)

3.3.2.35. 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 3.32.

3.3.2.36. 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 por 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 3.31 y 3.33 puede verse un ejemplo de uso.

3.3.2.37. 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**>, cadenas 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 morfológicos 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 morfológicos 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 3.31, la parte `lemq` del <**clip**> se coloca detrás de la cabeza del lema y de los símbolos morfológicos 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 morfológico que descri-

```
<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 3.32: Fragmento de una regla donde el patrón detectado es determinante–nombre (continúa en la fig. 3.33): 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 3.33: Fragmento de una regla (viene de la fig. 3.32). 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).

be 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.

3.3.2.38. 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 '+' en el flujo de datos no XML. 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.

3.3.2.39. 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.

3.3.3. Generación automática del módulo de transferencia estructural

[PENDIENTE]

3.4. Desformateador y reformateador

3.4.1. 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 3.4.2.

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 flujo de datos entre módulos (que se discute en detalle en el capítulo 2); por ejemplo, en el formato de flujo no basado en XML (secciones 2.2.2 y 2.3.2), los *superblancos* irán entre corchetes `'['` y `']'`. Cada una de estas cadenas encapsuladas será tratada como si de un blanco **<b/>** (sección 3.1.2.3.14) 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 11 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.

3.4.1.1. 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. Por ejemplo, en el flujo de datos no basado en XML 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. Por ejemplo, en el flujo de datos descrito en el capítulo 2 comienzan por la cadena '`[@`', siguen con el nombre del archivo donde 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 3.34 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 3.35 muestra el resultado de procesar el anterior documento.

```
<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 3.34: 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 3.35: Ejemplo de documento HTML con los bloques de formato encapsulados por el desformateador

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>`
- Los saltos de línea sencillos también se encapsulan.

3.4.2. 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 concreto no es relevante, sólo se utiliza el orden que se establece entre 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.

- 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:

```
<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`, véase el documento [8].

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 3.36.

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 menor prioridad 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.

```
<?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 3.36: Parte de la definición de reglas del formato HTML

3.4.3. 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` [8] 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, XML, RTF y texto llano.

3.5. Módulos auxiliares

3.5.1. Módulo de preproceso del transfer

3.5.1.1. Motivación

El módulo de preproceso del transfer, `pretransfer`, se ocupa de fragmentar las unidades multipalabra compuestas (ver apartado 3.1.2.5) 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 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.

3.5.1.2. 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.

Para las reglas del módulo de transferencia se deja la responsabilidad de generar una salida de este módulo que tenga el orden original que acepta el generador, así como 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.

Capítulo 4

Descripción de los datos lingüísticos existentes

[PENDIENTE]

4.1. Traductor español–catalán

4.2. Traductor español–gallego

4.3. Cómo introducir nuevos datos lingüísticos

4.3.1. Introducir palabras en los diccionarios

4.3.2. Introducir reglas de transferencia estructural

4.3.3. Añadir datos para el desambiguador léxico categorial

Capítulo 5

Descripción de la interfaz web

[PENDIENTE]

Capítulo 6

Instalación del sistema

[PENDIENTE]

Capítulo 7

Ideas de trabajo futuro

[PENDIENTE]

ALGUNAS IDEAS:

- En algún momento podría ser recomendable la construcción de herramientas para facilitar el manejo de diccionarios en el formato propuesto que faciliten la labor a los lingüistas, aunque no se considera prioritario dada la simplicidad del formato XML. Por ejemplo:
  - formularios para inserción de palabras en los diccionarios
  - vistas XSLT para hacer más visibles los diccionarios
  - una herramienta que pueda extraer los lemas de los diccionarios morfológicos automáticamente, utilizando el atributo `lm` de las entradas.
  - En XML hay un mecanismo (*xinclude*) que puede servir para incluir la sección de definición de símbolos gramaticales común a tres (o más) diccionarios desde un fichero externo. De este modo no sería necesario incluir esta sección en cada uno de los diccionarios.
- Puede ser útil construir adivinadores (*guessers*) de tipos de palabras desconocidas por el diccionario morfológico mediante la identificación de determinados prefijos o sufijos.
- Realizar mejoras en el módulo de transferencia para que pueda realizar algún tipo de análisis sintáctico
- Modificar el desambiguador para que puedan incluirse prohibiciones de longitud mayor a 2 sin cambiar el orden de Markov.

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 está disponible en `http://xixona.dlsi.ua.es/dix.dtd`.

La explicación de los distintos elementos se encuentra en el apartado 3.1.2.3.

```
<!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>

<!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. La descripción de los distintos elementos se encuentra en el apar-

tado 3.2.2.2.

```
<!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. La descripción de los distintos elementos se encuentra en el apartado 3.3.2.

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

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

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

<!ENTITY % value "(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 lemma CDATA #IMPLIED
                    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. La descripción de los distintos elementos se encuentra en el apartado 3.4.2.

```
<!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	
ACRONIMO	Acrónimo	No	ONU, sida, BCH
INTPRN	Pronombre interrogativo	Sí	quién, cuál
INTADJ	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
CNJSUB	Conjunción subordinante	Sí	que
CNJCOORD	Conjunción coordinante	Sí	y, pero
CNJADV	Conjunción subordinante adverbial	No	si
DETNT	Determinante neutro	Sí	lo
DET	Determinante	Sí	el, un
INTERJ	Interjección	No	ojalá, hola
NOM	Nombre	No	casa, coche
ANTROPONIM	Nombre propio de persona	No	Fernando
TOPONIM	Nombre propio de lugar	No	Alicante
NPOTROS	Otros nombre propios	No	Linux, Seat
NUM	Numeral	Sí	tres, cuatro
PREDETNT	Predetermiante neutro	Sí	todo
PREDET	Predeterminante Ej. todo	Sí	toda

[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
PREP	Preposición	Sí	ante, desde
PRNTNNT	Pronombre tónico neutro	Sí	algo, esto
PRNTN	Pronombre tónico	Sí	ambos, nadie
PRNENCREF	Pronombre enclítico reflexivo	Sí	
PRNPROREF	Pronombre proclítico reflexivo	Sí	se
PRNENC	Pronombre enclítico	Sí	
PRNPRO	Pronombre proclítico	Sí	le, te
VLEXINF	Verbo léxico en infinitivo	No	cantar, reír
VLEXGER	Verbo léxico en gerundo	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	No	ser
VSERGER	Verbo ser en gerundo	No	siendo
VSERPARTPI	Verbo ser en participio	No	sido
VSERPFCI	Verbo ser en presente, futuro o condicional de indicativo	No	soy, seré, sería
VSERIPI	Verbo ser en pretérito imperfecto o perfecto simple de indicativo	No	era, fui
VSERSUBJ	Verbo ser subjuntivo	No	fueras
VSERIMP	Verbo ser imperativo	No	sé
VHABINF	Verbo haber en infinitivo	No	haber
VHABGER	Verbo haber en gerundo	No	habiendo
VHABPARTPI	Verbo haber en participio	No	habido
VHABPFCI	Verbo haber en presente, futuro o condicional de indicativo	No	hay, habrán, habría
VHABIPI	Verbo haber en pretérito imperfecto o perfecto simple de indicativo	No	había, hubo
VHABSUBJ	Verbo haber subjuntivo	No	hubieran
VMODINF	Verbo modal en infinitivo	No	deber, poder
VMODGER	Verbo modal en gerundo	No	debiendo
VMODPARTPI	Verbo modal en participio	No	podido
VMODPFCI	Verbo modal en presente, futuro o condicional de indicativo	No	puede, deberá, podría
VMODIPI	Verbo modal en pretérito imperfecto o perfecto simple de indicativo	No	podía, debió
VMODSUBJ	Verbo modal subjuntivo	No	pudiese, debiéramos
VMODIMP	Verbo modal imperativo	No	poded, debed
ADJ	Adjetivo	No	gracioso, verde
REL	Relativo	Sí	quien, cuya
RELADV	Relativo adverbial	Sí	cuando, donde
Etiquetas compuestas			

Etiqueta	Descripción	Cerrada	Ejemplos
PREPDET	Contracción de preposición y determinante	Sí	del, al
PRCNJ	Multipalabra formado por preposición y conjunción	Sí	
PRREL	Multipalabra formado por preposición y relativo	Sí	
INFLEXENC	Verbo léxico en infinitivo con enclíticos	No	dármelo, cantarlo
GERLEXENC	Verbo léxico en gerundio con enclíticos	No	cantándosela
IMPLEXENC	Verbo léxico en imperativo con enclíticos	No	dímelo
INFSERENC	Verbo ser en infinitivo con enclíticos	Sí	serlo
GERSERENC	Verbo ser en gerundio con enclíticos	Sí	siéndolo
IMPSERENC	Verbo ser en imperativo con enclíticos	Sí	sedlo
INFHABENC	Verbo haber en infinitivo con enclíticos	Sí	habérsela
GERHABENC	Verbo haber en gerundio con enclíticos	Sí	habiéndole
INFMODENC	Verbo modal en infinitivo con enclíticos	Sí	poderla, deberlo
GERMODENC	Verbo modal en gerundio con enclíticos	Sí	debiéndosela
IMPMODENC	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

[Sección incompleta: sólo se describirán las etiquetas que cambien respecto de las usadas para el español]

B.2.3. Desambiguador de gallego

[Sección incompleta: sólo se describirán las etiquetas que cambien respecto de las usadas para el español]

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 6)

FLLM Forma léxica de la lengua meta

FLLO Forma léxica de la lengua origen

FS Forma superficial (véase la página 6)

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] Antonio M. Corbí-Bellot, Mikel L. Forcada, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Gema Ramírez-Sánchez, Felipe Sánchez-Martínez, Iñaki Alegria, Aingeru Mayor, and Kepa Sarasola. An open-source shallow-transfer machine translation engine for the romance languages of spain. In *Proceedings of the European Association for Machine Translation 10th Annual Conference*, pages 79–86, May 2005.

[4] 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.

[5] 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.

[6] 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.

[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] M.E. Lesk. Lex — a lexical analyzer generator. Technical Report Technical Report 39, AT&T Bell Laboratories, Murray Hill, N.J., 1975.

[9] Ferran Pla and Antonio Molina. Improving part-of-speech tagging using lexicalized HMMs. *Journal of Natural Language Engineering*, 10(2):167–189, June 2004.

[10] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[11] 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).