Universitat d'Alacant
Universidad de Alicante

Esta tesis doctoral contiene un índice que enlaza a cada uno de los capítulos de la misma.
Existen asimismo botones de retorno al índice al principio y final de cada uno de los capítulos.

Ir directamente al índice

Para una correcta visualización del texto es necesaria la versión de Adobe Acrobat Reader 7.0 o posteriores

Aquesta tesi doctoral conté un índex que enllaça a cadascun dels capítols. Existeixen així mateix botons de retorn a l'index al principi i final de cadascun deis capítols .

Anar directament a l'index

Per a una correcta visualització del text és necessària la versió d' Adobe Acrobat Reader 7.0 o posteriors.

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

UNIVERSITAT D'ALACANT
GEDIP
30 MAIG 2002
ENTRADA | SORTIDA
Núm. 557 | Núm.

Universitat d'Alacant
Universidad de Alicante

Compiladores de datos lingüísticos para la generación de módulos de estados finitos para la traducción automática

Tesis doctoral presentada por:
Alicia Garrido Alenda
para acceder al grado de doctora

Departamento de Lenguajes y Sistemas Informáticos,
Universidad de Alicante,
E-03071 Alacant, Spain.

Directores:
Mikel L. Forcada Zubizarreta y Rafael C. Carrasco Jiménez

Mayo de 2002

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Universitat d'Alacant
Universidad de Alicante

Índice General

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

Prefacio

El objetivo principal de esta tesis ha sido desarrollar métodos y herramientas para generar sistemas de traducción automática, de manera que, sobre herramientas informáticas ya realizadas, únicamente sea necesario generar los datos lingüísticos referentes al par de lenguas entre las que se desea traducir. Con esta finalidad se han desarrollado compiladores que transforman los datos lingüísticos establecidos, en estructuras de datos eficientes para su consulta. En concreto, se han usado técnicas basadas en estados finitos. Estas técnicas se han mostrado muy eficientes en el campo de la traducción automática, tanto desde el punto de vista del espacio como de la velocidad. Como parte del sistema de traducción automática generado también se consideran los programas que consultan los datos lingüísticos transformados, aunque son programas cerrados en los que la única diferencia de un sistema de traducción a otro consiste en los datos que leen. Igualmente se han desarrollado algoritmos que permiten el mantenimiento eficiente de la versión informatizada de las fuentes lingüísticas necesarias.

Esta tesis está organizada de forma que inicialmente se realiza una introducción sobre la traducción automática y la formalización de las tareas de la traducción. A continuación se dedica el capítulo 2 a la descripción de los diccionarios utilizados para generar los sistemas de traducción de esta tesis. En el capítulo 3, se hace una descripción de las herramientas que se pueden utilizar para transformar estos diccionarios en estructuras de datos eficientes, así como una comparación de las ventajas e inconvenientes que presentan, para continuar en el capítulo 4 con la exposición de cómo se realiza esta transformación en los diferentes casos. En el capítulo 5 se exponen los diferentes algoritmos utilizados para hacer que estas estructuras de datos sean lo más eficientes posible espacialmente, y se realiza un estudio comparativo entre las diferentes implementaciones realizadas. A continuación, en el capítulo 6, se describen los algoritmos diseñados para mantener los diccionarios, ya transformados en una estructura de datos, de forma eficiente. En el capítulo 7 se realiza una descripción de los algoritmos que utilizan estas estructuras de datos en un sistema de traducción automática: el analizador

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

y generador morfológicos, el módulo de transferencia léxica y el postgenerador. El capítulo 8 se dedica a explicar cómo se genera un módulo no basado estrictamente en estas estructuras: el módulo de transferencia estructural. Por último, en el capítulo 9 se describe la aplicación de todas estas herramientas a la generación de un sistema de traducción automática que está en funcionamiento en internet, interNOSTRUM.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

Agradecimientos

A continuación quiero expresar mi agradecimiento a todas aquellas personas e instituciones que han colaborado de una u otra manera a la realización de esta tesis.

En primer lugar quiero agradecer a Mikel L. Forcada la ayuda y el tiempo que me ha dedicado desde el comienzo de mi actividad investigadora y que me alentó a desarrollar, ya que sin sus consejos, disposición y apoyo no hubiera podido desarrollar mi trabajo en este área. También quiero agradecerle, junto a Rafael C. Carrasco, su labor como directores de esta tesis; su asesoramiento, el esfuerzo dedicado y su apoyo han sido esenciales para mí a la hora de realizar este trabajo.

Quiero agradecer a Amaia Iturraspe Bellver, Anna Esteve Guillén, Sandra Monserrat Buendia y Maribel Guardiola Savall, el grupo de lingüistas con el que he trabajado a lo largo del desarrollo de este trabajo, las sugerencias aportadas para mejorar las prestaciones del producto, así como las bases de datos lingüísticos que han confeccionado, sin las cuales esta tesis no habría sido posible.

Al Departamento de Lenguajes y Sistemas Informáticos por su apoyo en mi actividad docente e investigadora. A la Caja de Ahorros del Mediterráneo y a la Universidad de Alicante por subvencionar el desarrollo del traductor automático **interNOSTRUM** con el que inicié esta tesis ya que fui becaria de este proyecto desde su comienzo en enero de 1.999, y al proyecto SISHI-TRA[1], dentro del cual he llegado a su conclusión.

Por último quiero agradecer a mi familia y mis amigos el apoyo que me han dado en todo momento y que me han animado a seguir, en especial a mi pareja, Paco, que ha sufrido estoicamente mis nervios y cambios de humor.

[1] Código TIC 2000-1599, subproyecto 2

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

Capítulo 1

Introducción

La denominación *traducción automática* (TA) se refiere tanto al resultado como al proceso mismo de traducir, mediante un sistema informático (ordenadores y programas), textos informatizados[1] escritos en una *lengua origen* (LO) para producir textos informatizados escritos en *lengua meta* (LM). Por extensión, es también el nombre de una rama de la lingüística computacional que se ocupa del diseño, la evaluación y el uso de los sistemas usados para efectuar este proceso.

1.1 Lingüística de la traducción automática

En esta sección[2] se describen brevemente algunos conceptos lingüísticos relevantes para el diseño de los sistemas de traducción automática, y se presentan brevemente las aproximaciones necesarias para poder aplicar estos conceptos lingüísticos en los sistemas reales de TA.

1.1.1 La aproximación oracional

La primera aproximación común a prácticamente todos los sistemas de traducción automática actuales es la siguiente: traducir un texto es traducir, una a una, las oraciones de las que se compone. Se trata de una aproximación bastante radical, ya que, por ejemplo, no tiene en cuenta la estructura

[1] La denominación *texto informatizado* se refiere a un fichero o archivo de ordenador que contien un texto codificado en un formato conocido.

[2] Esta sección se basa parcialmente en las ideas expuestas por Mikel L. Forcada en su ponencia "Tensions entre l'adequació lingüística i la factibilitat informàtica durant el desenvolupament d'un sistema de traducció automàtica", presentada en el XVII Encuentro de la Asociación de Jóvenes Lingüistas celebrado en Alicante en abril del 2002.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

del discurso o las relaciones de correferencia que pueden existir entre los pronombres de la oración actual y sus antecedentes, que pueden haber aparecido en oraciones anteriores. Los (escasos) sistemas que tratan alguno de estos fenómenos interoracionales lo hacen de manera particular y adicional, y para evitar algunos errores frecuentes, sin considerar la traducción del texto como un todo.[3]

En esta visión, traducir una oración escrita en LO a LM consiste en producir una oración en LM que tenga la misma *interpretación*, es decir, el mismo *significado*. Si seguimos este esquema, la traducción de una oración se realiza en dos etapas: en la primera, se obtiene una representación de la interpretación de la oración original, y en la segunda, se construye, a partir de esta interpretación, la oración correspondiente en LM. Tanto en lingüística general como en lingüística computacional, esta representación se denomina a veces *forma lógica* (Radford et al. 1999, p. 362; Allen 1995, cap. 8).

1.1.2 Principio de composicionalidad y traducción de oraciones

La obtención de la interpretación de una oración por parte de las personas se suele explicar normalmente en lingüística en términos del *principio de composicionalidad semántica* (PCS), también denominado simplemente *principio de composicionalidad*. Este principio enuncia que "la interpretación de una oración viene determinada por la interpretación de las palabras que aparecen en la oración y por la estructura sintáctica de la misma" (Radford et al. 1999, p. 357); este principio se puede formular de manera constructiva como sigue: la interpretación de una oración se construye

- a partir de las interpretaciones de cada una de las palabras que la forman
- componiendo estas interpretaciones siguiendo las agrupaciones indicadas por la sintaxis de la oración para construir interpretaciones más complejas

hasta que se obtiene la interpretación completa de la oración.

Esta formulación sugiere que las interpretaciones son de naturaleza estructurada, ya que se construyen por composición de interpretaciones más sencillas; además, si se representan, como es usual, la sintaxis de las oraciones

[3]Por ejemplo, para traducir correctamente del japonés al inglés es necesario asignar antecedentes a los *pronombres cero*, ya que el japonés elimina sistemáticamente el sujeto, objeto, etc. de las oraciones cuando el contexto permite determinarlos; véanse por ejemplo Mori et al. (1999); Nakaiwa (1999).

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

en forma de árboles, la formulación descrita es básicamente *ascendente* (la construcción se realiza recorriendo el árbol desde las hojas hacia la raíz). De hecho, visto así, el PCS aparece emparentado con el concepto central de la informática de compiladores, el de *traducción dirigida por la sintaxis* (Aho et al. 1986, p. 25). El PCS es un principio crucial en lingüística ya que sirve para explicar, por ejemplo, por qué las personas pueden asignar una interpretación a (es decir, comprender) oraciones que no han oído o leído nunca.

Esta formulación del PCS es adecuada para describir la primera etapa de las dos citadas más arriba para la traducción de una oración, la construcción de una intepretación para la oración original, la cual se realiza, a su vez, en etapas: se realiza el análisis morfológico y sintáctico, se asigna la interpretación de cada palabra y finalmente, se construye composicionalmente una representación estructurada de la oración. Sin embargo, no parece adecuada para la segunda etapa (la construcción de una oración en LM a partir de la citada interpretación). Sin embargo, recientemente (Tellier 2000) se ha formulado el principio de composicionalidad de manera completamente simétrica. En esta formulación, existen dos correspondencias bidireccionales básicas:

1. entre *palabras* e *interpretaciones elementales* y

2. entre *reglas de composición sintáctica* (que indican como se construyen constituyentes de las oraciones a partir de otros) y *reglas de composición semántica* (que indican cómo se construyen interpretaciones parciales o totales a partir de interpretaciones parciales).

El PCS, en su formulación simétrica, es importante en lingüística ya que permite explicar por qué la sintaxis de las lenguas naturales es estructurada, es decir, composicional, y, a veces, recursiva, ya que las interpretaciones que las personas quieren comunicar también son recursivas. Pero además, sirve para describir la segunda etapa en la traducción de una oración: analizando la interpretación estructurada, se construye un árbol de análisis sintáctico de la LM y para cada una las interpretaciones básicas que forman la interpretación estructurada se elige una palabra en LM adecuada y se coloca en la hoja correspondiente del árbol. Finalmente, se genera la oración a partir de este árbol.

El proceso de traducción de una oración, se puede, por tanto, describir esquemáticamente como se muestra en la figura 1.1.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

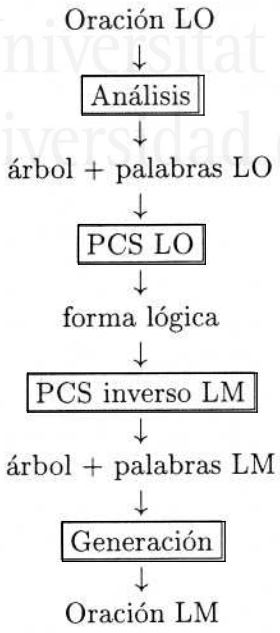


Figura 1.1: Proceso de traducción de una oración en términos del principio de composicionalidad semántica (PCS).

1.1.3 Ambigüedad

En toda la discusión anterior, se ha mencionado "la interpretación" de una oración como si fuese única. Sin embargo, una de las características más importantes del lenguaje humano es que es ambiguo; esta característica se refleja claramente en las oraciones. Las oraciones, desde el punto de vista del PCS, pueden ser ambiguas:

- porque una o más de sus palabras tengan más de una interpretación (*ambigüedad léxica*),
- porque la oración tenga más de un árbol de análisis sintáctico (*ambigüedad estructural* o *sintactica*)
- o, en algunos casos, por las dos causas a la vez.

La elección de la interpretación correcta por parte de un sistema de traducción automática, necesaria en muchos casos para producir una traducción correcta, es una tarea que queda lejos de ser trivial, ya que, si bien las personas pueden usar el contexto en el acto de la comunicación y sus expectativas y creencias sobre el mundo real para descartar con bastante seguridad muchas (idealmente todas menos una) de las posibles interpretaciones de una

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

oración, un sistema de traducción automática podrá, en el mejor de los casos, usar parte del *cotexto*, es decir, el texto que circunda a la palabra o a la oración ambigua.

1.1.4 La aproximación de transferencia

Según el relato anterior, el trabajo de una persona que se dedica profesionalmente a la traducción consiste en leer cada oración del texto en LO, determinar la interpretación adecuada (descartando las interpretaciones que no lo son en el caso de las oraciones ambiguas), y construir en la LM una oración que represente adecuadamente la interpretación; por tanto, el traductor o la traductora debe *comprender* completamente cada oración antes de traducirla. Esto es claramente implausible, ya que no se puede esperar que un traductor sea un experto en mecánica o en diseño de circuitos integrados para traducir documentos sobre estas disciplinas. En estos casos, los traductores operan *transformando estructuras* o *patrones* y *sustituyendo el léxico* (y, con particular cuidado, la terminología propia de la disciplina en cuestión), usando reglas y estrategias que les permiten obtener traducciones razonablemente correctas sin tener que comprender los textos en profundidad. El traductor, por tanto, produce el texto meta mediante una modificación o transformación del texto original.[4] Este modo de traducción es especialmente interesante a la hora de construir sistemas de traducción automática.

El nuevo esquema, en el que no se construye una interpretación explícita, se representa en la figura 1.2, se puede denominar traducción por *transferencia*. La transferencia es de dos clases: *transferencia léxica* o substitución de palabras de la LO por equivalentes adecuadas en LM y *transferencia sintáctica* o transformación de las estructuras sintácticas de la LO para convertirlas en estructuras adecuadas de la LM. En la mayoría de los sistemas, los dos procesos de transferencia se diseñan de manera que sean bastante independientes uno del otro (lo que constituye una aproximación adicional).

Queda claro que para programar un sistema de TA se deben formular los procesos de traducción —muchos de los cuales son ejecutados de manera intuitiva por parte de los profesionales— de manera explícita y programable, es decir, en forma de reglas claras. Esto exige una reflexión lingüística (traductológica) sobre los citados procesos. Además, la formulación en términos

[4]Sager (1993) define la traducción como "un rango de actividades humanas deliberadas, que se realizan como resultado de instrucciones recibidas de un tercero, y que consisten en la producción de textos en una lengua meta (LM), basada, entre otras cosas, en la modificación de un texto en lengua origen (LO) para hacerlo adecuado a un nuevo propósito". Esta definición de traducción es más general, ya que no exige la comprensión del texto ni que éste sea estrictamente correcto.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

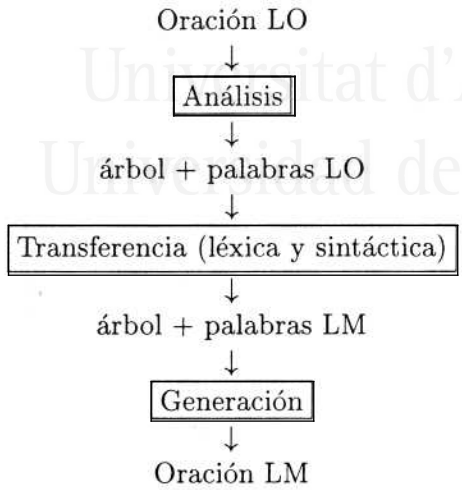


Figura 1.2: Proceso de traducción de una oración sin construir explícitamente una interpretación de la misma.

de reglas puede comportar aún más aproximaciones y simplificaciones.

1.1.5 Traducción entre lenguas similares

Cuando la lengua origen y la lengua meta son similares (por ejemplo, porque están estrechamente emparentadas, como en el caso del castellano y el catalán), puede no ser necesario realizar grandes transformaciones sintácticas, y por tanto, puede no ser necesario realizar completamente el análisis sintáctico. Sólo es necesario este análisis cuando se detecta una estructura divergente[5]. Estos modelos se pueden ver también como traductores palabra por palabra (transferencia fundamentalmente léxica) que realizan pequeños ajustes sintácticos en los casos en los que los resultados serían inaceptables. Algunos de los problemas que, por similares que sean las lenguas, no se pueden dejar de abordar son los siguientes:

- La concordancia: En la traducción se puede producir un cambio de género o número en alguna de las palabras de la frase, de forma que se hace necesario aplicar alguna regla para que las palabras que concordaban en la lengua origen también lo hagan en la lengua meta.

[5]Arnold et al. (1994, p. 185) discuten este problema en detalle y describe lo que denominan "traducción automática flexible" o "multinivel": "lo que se requiere es una aproximación a la traducción basada en reglas que sea suficientemente flexible para realizar análisis profundo sólo cuando sea necesario [...]".

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

- La homografía: Una misma palabra puede pertenecer a más de una categoría léxica, por tanto es conveniente disponer de algún sistema para elegir una categoría u otra, ya que en otro caso sólo pueden asignar una categoría léxica por palabra.

- La polisemia: Una palabra puede tener más de un significado en la lengua origen, de forma que cada uno de estos significados se traduzca de forma diferente en la lengua meta. Para solucionar este problema es necesario disponer de conocimiento del contexto de la palabra para saber qué significado asignar.

- El orden de las palabras: En el proceso de traducción se encuentran estructuras en la lengua origen que requieren un reordenamiento de su traducción a la lengua meta para obtener la estructura equivalente, por tanto es necesario aplicar algún tipo de regla que reconozca la estructura de la lengua origen y genere la correspondiente estructura de la lengua meta.

1.2 Modelos de traducción automática

Los sistemas de TA se pueden clasificar de manera general en tres grandes grupos (Hutchins y Somers 1992): sistemas basados en la traducción directa, sistemas basados en la traducción indirecta por interlingua y sistemas basados en traducción indirecta por transferencia.

1. Traducción directa. Es un sistema rudimentario de la traducción automática ya que no se genera ninguna representación intermedia de la lengua origen y traduce en dos fases:

   - Segmenta el texto de entrada en unidades (palabras o fragmentos) y
   - las sustituye por equivalentes adecuados en la lengua meta.

   Estos sistemas tienen dos formas de funcionamiento básico: pueden detectar fragmentos enteros de un texto en la lengua origen (LO) y sustituirlo por su correspondiente fragmento en la lengua meta (LM), en cuyo caso se trata de un sistema basado en memorias de traducción, o bien sustituir cada palabra del texto en la LO por su correspondiente palabra en la LM, en cuyo caso se dice que es un sistema de traducción palabra por palabra.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

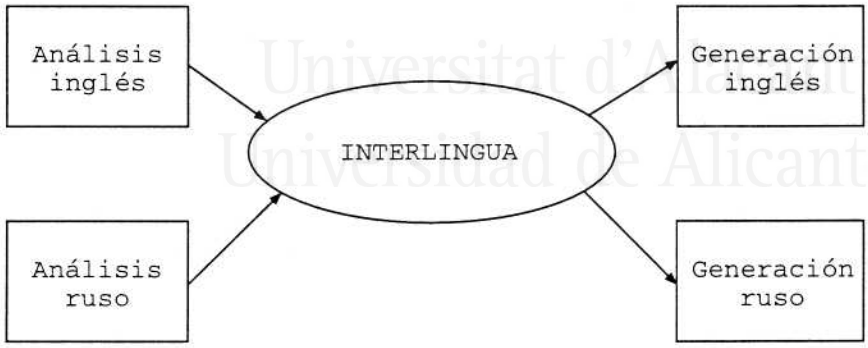


Figura 1.3: Sistema de TA basado en el modelo de traducción indirecta por interlingua para dos lenguas.

2. Traducción indirecta por interlingua. Los sistemas basados en esta estrategia analizan el texto en la lengua origen de forma que generan una representación intermedia, a partir de la cual generan directamente el texto en la lengua meta; es decir la representación intermedia generada es una representación abstracta tanto de la lengua origen como de la lengua meta, por tanto cada módulo de análisis es independiente del resto de módulos de análisis y generación (figura 1.3). Los sistemas de interlingua se corresponden bastante fielmente con la estrategia de traducción basada en la aplicación del principio de composicionalidad semántica descrita en el apartado 1.1.2, ilustrada en la figura 1.1: la *interlingua* es, por tanto, el lenguaje en el que se representan las interpretaciones de las oraciones. Algunos sistemas de TA que se basan en esta estrategia son el sistema KANT (Mitamura et al. 1991), el sistema KANTOO[6] (Nyberg y Mitamura 2000), el sistema DLT (Schubert 1988) o el sistema Rosetta (Rosetta 1994). Otro proyecto basado en esta estrategia es el sistema de TA de diálogos verbmobil[7].

3. Traducción indirecta por transferencia. Los sistemas basados en esta estrategia realizan las traducciones en tres fases (Arnold 1993):

   - La fase de análisis produce a partir del texto en la lengua origen, una representación abstracta (RALO). En la RALO se eliminan todos los detalles del texto en la lengua origen que son irrelevantes para la traducción y se destacan aquellas características o relaciones que sí lo son.

[6] Versión más eficiente del sistema KANT
[7] `http://verbmobil.dfki.de`

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

- La fase de transferencia convierte la RALO en otra representación abstracta similar correspondiente a la lengua meta (RALM).
- La fase de generación produce el texto en la lengua meta a partir de la RALM.

La arquitectura de traducción indirecta por transferencia es el modelo estándar para la TA contemporánea, y lo ha sido durante mucho tiempo (Arnold 1993). En general, los sistemas de TA basados en esta estrategia se clasifican de acuerdo con la profundidad del análisis que realizan, de forma que se puede hablar de sistemas de transferencia morfológica, de transferencia sintáctica y de transferencia semántica. La elección de la profundidad del análisis a la hora de realizar un sistema de TA se fundamenta generalmente en la naturaleza y divergencia de traducción entre las lenguas implicadas (Vandooren 1993).

(a) Traducción indirecta por transferencia morfológica avanzada. Estos sistemas utilizan como representación abstracta el análisis morfológico de las palabras para detectar secuencias de categorías léxicas de forma que aplican una forma de análisis sintáctico muy rudimentario (Forcada 2000). Esta estrategia es la que utilizan algunos sistemas de TA comerciales como Reverso de Softissimo[8] o Transcend de SDLX,[9] y sobre la que se basa el sistema que se estudia en esta tesis. Esta aproximación se corresponde bastante bien con la descrita en el apartado 1.1.5.

(b) Traducción indirecta por transferencia sintáctica. Los sistemas basados en esta estrategia utilizan como representación abstracta el árbol de análisis sintáctico (o una estructura equivalente) de frases del texto en la lengua origen. Este árbol describe las relaciones sintácticas que existen entre los componentes de una frase y además incluye la información morfológica necesaria para realizar la traducción. Sistemas de TA basados en esta estrategia son METAL (Slocum 1987; Bennett y Slocum 1985) en el que se basa el traductor automático T1 de Langenscheidt y el traductor INCYTA, ambos de Sail Labs, y el sistema de TA multilingüe en desarrollo por el grupo IXA[10] (Díaz-Ilarraza et al. 2000, 2001). La traducción automática por transferencia sintáctica se corresponde bastante bien con la aproximación de transferencia descrita en el

[8] http://www.reverso.net
[9] http://www.freetranslation.com
[10] http://ixa.si.ehu.es/

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

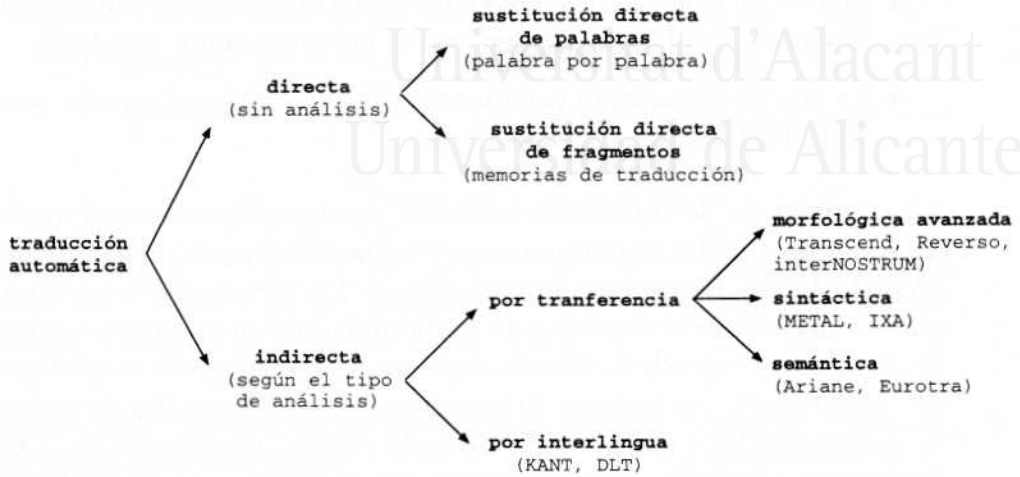


Figura 1.4: Esquema de los métodos utilizados para la traducción automática

apartado 1.1.4, ilustrada en la fig. 1.2, ya que en aquella aproximación la RALO y la RALM se basan precisamente en árboles de análisis sintáctico.

(c) Traducción indirecta por transferencia semántica. Los sistemas basados en esta estrategia utilizan una representación abstracta que considera las relaciones semánticas existentes entre los diferentes componentes de una frase del texto en la lengua origen. Esta representación puede ser un árbol sintáctico *decorado* con rasgos semánticos, una red semántica o estructuras del estilo de las formas lógicas. Alguno de los sistemas de TA que se basan en esta estrategia son el sistema Ariane (GETA) (Vauquois y Boitet 1985), el sistema Eurotra (Allegranza et al. 1991) o el sistema SUSY (Maas 1987).

La figura 1.4 muestra un esquema de esta clasificación.

1.3 Usos de la traducción automática

Actualmente los principales usos de la traducción automática se pueden clasificar en cuatro grandes grupos (Hutchins 1999):

- La *difusión* o *diseminación* de información consiste en utilizar la traducción automática para producir documentos que tienen que ser distribuidos a más de un usuario; habitualmente se utiliza para traducir manuales técnicos o documentos internos de empresa.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

- La *asimilación* de información consiste en utilizar la traducción automática para obtener información a partir de un documento escrito (por ejemplo los documentos en Internet) en otra lengua, por ejemplo, para decidir si es relevante o no antes de traducirlo completamente.

- El *intercambio de información* consiste en la traducción de mensajes electrónicos (correo electrónico) o conversaciones electrónicas a través de la red (*chat*).

- La *extracción de información* en grandes bases de datos de documentos escritos en otra lengua.

Para la asimilación, el intercambio y la extracción de información es esencial la rapidez en la traducción, ya que la tendencia actual se orienta hacia la inclusión de la traducción automática como un componente más de los sistemas de comunicación y gestión de información. Por este motivo, es necesario transformar los datos lingüísticos a un formato que sea eficaz temporalmente para su utilización por programas informáticos y al mismo tiempo sea eficiente espacialmente. Es decir, es necesaria una estructura que permita una consulta rápida de los datos que contiene y que ocupe el menor espacio posible. En un sistema de TA indirecta hay como mínimo tres fuentes de datos de tamaño considerable:

- El diccionario morfológico de la lengua origen que se usa en la fase de análisis y relaciona la forma superficial (FS) de una palabra con su forma léxica (FL), que se compone de lema, categoría léxica (nombre, verbo, etc.) e indicadores morfológicos de flexión (género, número, persona, etc.).

- El diccionario bilingüe para la fase de transferencia, que asocia formas léxicas de la lengua origen (FLLO) con formas léxicas de la lengua meta (FLLM).

- El diccionario morfológico de la lengua meta para la fase de generación, que relaciona una FL con su correspondiente FS. Es decir, realiza la operación inversa que un diccionario morfológico de la lengua origen.

Además de estas fuentes de datos léxicas y morfológicas, y dependiendo de la estrategia concreta de traducción automática, son necesarias fuentes de información lingüística, como por ejemplo, reglas para realizar transformaciones sintácticas.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

1.4 Definiciones y conceptos previos

Un *símbolo* es una entidad abstracta no definida formalmente, tal que las letras y los dígitos son ejemplos de símbolos utilizados frecuentemente. Una *cadena* es una secuencia finita de símbolos yuxtapuestos. La longitud de una cadena w, que se denota como $|w|$, es el número de símbolos que componen la cadena. Se denota por ϵ la cadena vacía, una cadena compuesta por cero símbolos significativos, es decir, $|\epsilon| = 0$.

Un *alfabeto* es un conjunto finito de símbolos. Un *lenguaje universal* es el conjunto de todas las cadenas (de longitud finita) que se pueden formar a partir de un alfabeto Σ, que se denota como Σ^*. Un *lenguaje* es un subconjunto de cadenas de Σ^*.

La concatenación de dos cadenas es la cadena formada por la escritura de la primera cadena seguida de la segunda, sin espacios en blanco intermedios. Para indicar la concatenación de dos cadenas normalmente no se utiliza ningún operador, de forma que se denota mediante la yuxtaposición de las cadenas a concatenar.

Informalmente, un *prefijo* de una cadena es cualquier secuencia de símbolos contiguos que aparecen al principio de esa cadena, y un *sufijo* es cualquier secuencia de símbolos contiguos que aparecen al final de esa cadena. Formalmente, $x \in \Sigma^*$ es un prefijo (respectivamente sufijo) de $w \in \Sigma^*$ si existe una cadena $y \in \Sigma^*$ tal que $xy = w$ (respectivamente $yx = w$). Por ejemplo, la cadena *sofá* tiene como prefijos las cadenas ϵ, *s*, *so*, *sof* y *sofá* y como sufijos las cadenas ϵ, *á*, *fá*, *ofá* y *sofá*. Cualquier prefijo o sufijo de una cadena, que no sea ella misma, se llama prefijo o sufijo *propio*.

Definición 1.1 (Conjunto de prefijos de una cadena) *El conjunto de prefijos de una cadena $w \in \Sigma^*$ se define como*

$$\Pr(w) = \{x \in \Sigma^* : (\exists y \in \Sigma^* : xy = w)\}. \tag{1.1}$$

Definición 1.2 (Conjunto de sufijos de una cadena) *El conjunto de sufijos de una cadena $w \in \Sigma^*$ se define como*

$$\mathrm{Su}(w) = \{x \in \Sigma^* : (\exists y \in \Sigma^* : yx = w)\}. \tag{1.2}$$

El *prefijo común más largo* de un conjunto de cadenas $B \subseteq \Sigma^*$ es la cadena más larga del conjunto PC de prefijos comunes de cualquier longitud del conjunto de cadenas.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Definición 1.3 (Prefijo común más largo) *El prefijo común más largo* $\mathrm{PCML}(B)$ *de un conjunto de cadenas* $B \subseteq \Sigma^*$ *es la cadena más larga del conjunto*

$$\mathrm{PC}(B) = \bigcap_{w \in B} \mathrm{Pr}(w). \tag{1.3}$$

de sus prefijos comunes de cualquier longitud, es decir,

$$\mathrm{PCML}(B) = \mathrm{argmax}_{p \in \mathrm{PC}(B)} |p| \tag{1.4}$$

Por ejemplo, para calcular el prefijo común más largo del conjunto de cadenas formado por las palabras {*para,parca*}, en primer lugar se calculan los conjuntos:

$$\mathrm{Pr}(\mathrm{para}) = \{\epsilon, \mathrm{p}, \mathrm{pa}, \mathrm{par}, \mathrm{para}\}$$
$$\mathrm{Pr}(\mathrm{parca}) = \{\epsilon, \mathrm{p}, \mathrm{pa}, \mathrm{par}, \mathrm{parc}, \mathrm{parca}\}$$

A continuación se calcula el conjunto de los prefijos comunes de cualquier longitud:

$$\mathrm{PC}(\mathrm{para}, \mathrm{parca}) = \{\epsilon, \mathrm{p}, \mathrm{pa}, \mathrm{par}\}$$

Y por último se toma aquella cadena de este conjunto que tenga la mayor longitud:

$$\mathrm{PCML}(\mathrm{para}, \mathrm{parca}) = \mathrm{par}$$

El *cociente por la izquierda* de dos cadenas $x, w \in \Sigma^*$, que se denota como $x^{-1}w$, es el sufijo de w que resulta de eliminar su prefijo x; en caso de que x no sea prefijo de w, queda indefinido.

Definición 1.4 (Cociente izquierdo cadena-lenguaje) *El cociente por la izquierda de una cadena* $w \in \Sigma^*$ *con un conjunto de cadenas* $E \subseteq \Sigma^*$ *se define como*

$$w^{-1}E = \{x \in \Sigma^* \ : \ wx \in E\} \tag{1.5}$$

1.5 Procesamiento secuencial

Muchas operaciones de transformación del estilo del análisis o la generación morfológica, e incluso la consulta del diccionario bilingüe, se pueden formular de manera *secuencial* o *subsecuencial*. Intuitivamente, esto quiere decir que la entrada se procesa leyéndola de izquierda a derecha y que se pueden afianzar prefijos de la transformación que crecen según se lee de la entrada, ya que las formas flexionadas y sus correspondientes lemas *comparten prefijos* en el caso del análisis o generación morfológica, y los lemas en la lengua origen (LO) y

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

en la lengua meta (LM) pueden *compartir prefijos* en el caso de la consulta del diccionario bilingüe si las lenguas están relacionadas. Esta intuición se puede formalizar como se verá más adelante.

El procesamiento secuencial es apropiado cuando se tratan lenguas indoeuropeas, como el catalán y el castellano, ya que el análisis morfológico cambia normalmente las terminaciones (sufijos): el lema y sus formas flexionadas comparten un *prefijo*. Por ejemplo, un posible desarrollo del proceso de análisis morfológico de la palabra *casaban* es el siguiente:

- Cuando únicamente se ha leído *cas-* todavía hay muchas posibilidades, ya que puede ser, entre otras, cualquier forma de las palabras *caserío*, *castillo*, *casona*, *casa*, *casar*, *caspa* y *casaca*. Como mucho, se puede decir que el lema empieza por `cas`.

- Cuando se ha leído *casa-* el abanico de posibilidades se reduce a que puede ser una forma de *casa*, de *casar* o de *casaca*. Ya se puede asegurar que el lema empieza por `casa`.

- Cuando se ha leído *casab-* ya se puede decir que el lema es *casar*, que se trata de un verbo, y que más concretamente y con total seguridad, se trata de una forma del imperfecto de indicativo. El analizador puede completar el lema que tenía hasta el momento y proporcionar ya `casar<verb><impind>`[11].

- Después de leer *casaba-* todavía no se dispone de información para asegurar nada sobre la persona o el número del verbo.

- Finalmente, cuando se ha leído *casaban* ya se sabe que se trata de la tercera persona del plural. El analizador produce:

```
casar<verb><impind><3><pl>
```

El proceso se puede resumir en el alineamiento siguiente:

```
c a s a b  _      _        a   n
c a s a r  <verb> <impind> _  <3><pl>
```

Para la generación morfológica, es decir, para la generación de la FS a partir de la FL, el proceso que se sigue es similar, pero en lugar de leerse una palabra se lee un lema seguido de su categoría léxica y sus características morfológicas.

[11] Los símbolos "<" y ">" se utilizan para encapsular los indicadores morfológicos que describen la flexión de las formas superficiales, de manera que cada indicador se trate como un único símbolo.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

El proceso que sigue la traducción de lemas entre lenguas similares (emparentadas) es análogo, ya que muchas veces, las palabras comparten prefijos, quizá con pequeñas transformaciones ortográficas. Por ejemplo la traducción de la palabra analizada morfológicamente del castellano `recuerdo<n><m><sg>` al catalán, `record<n><m><sg>`, seguiría el siguiente proceso:

- Cuando únicamente se ha leído `rec-` todavía hay muchas posibilidades, y puede ser, entre otras, cualquier forma de las palabras *recorrer*, *recuperación*, *recurso*, *recuerdo*, *rectángulo* o *recodo* cuya traducción al catalán es *recórrer*, *recuperació*, *recurs*, *record*, *rectangle* y *recolze* respectivamente. Como mucho, se puede decir que el lema catalán empieza por `rec`.

- Cuando se ha leído `recu-` el abanico de posibilidades se reduce. Puede ser una forma de *recuperación*, *recurso* o *recuerdo* (*recuperació*, *recurs* y *record* en catalán). Como entre las traducciones ya no hay más prefijo común no se puede avanzar más.

- Cuando se ha leído `recue-` ya se sabe que el lema en castellano es *recuerdo* en este ejemplo y por tanto el lema en catalán es `record`, un nombre masculino. El módulo de transferencia léxica puede proporcionar ya `record<n><m>`.

- Después de leer `recuerdo<n><m>` todavía no se dispone de suficiente información para asegurar si es singular o plural.

- Finalmente, cuando se lee `recuerdo<n><m><sg>` ya se sabe que se trata del singular. El módulo de transferencia léxica produce:

  `record<n><m><sg>`

En el siguiente alineamiento se puede ver el proceso resumido:

```
r e c u e _ _ _ _ r d o <n><m><sg>
r e c _ o r d <n><m> _ _ _ _ _ <sg>
```

A continuación se formaliza el concepto de *transducción secuencial.*

Definición 1.5 (Transducción secuencial) *Una transformación o transducción de un lenguaje de entrada* $E \subseteq \Sigma^*$ *a otro de salida* Γ^*

$$\tau : E \rightarrow \Gamma^*$$

donde E *es el subconjunto de cadenas de* Σ^* *donde* τ *está definida, se dice que es* secuencial *si*

$$\tau(\epsilon) = \mu_0$$
$$\tau(w\sigma) = \tau(w)\zeta(w,\sigma) \quad \forall w, w\sigma \in \mathrm{Pr}(E), \sigma \in \Sigma$$

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

donde

- $\mu_0 \in \Gamma^*$ *es la cadena inicial de la transducción, que normalmente es la cadena vacía;*
- Σ *es el alfabeto de entrada;*
- $\zeta(w, \sigma) \in \Gamma^*$ *es la extensión añadida al leer* σ.

De esta forma cada vez que se lee un símbolo σ de la entrada, la función $\zeta(w, \sigma)$ alarga de $\tau(w)$ a $\tau(w\sigma)$ el prefijo de la transducción; es decir, va construyendo secuencialmente el prefijo de las transducciones de todas las cadenas de E que tienen el prefijo en común leído de la cadena de entrada. El alargamiento, en algunos casos puede ser nulo, cuando $\zeta(w, \sigma) = \epsilon$.

1.6 Procesamiento subsecuencial

Como se ha descrito en el apartado 1.1.3, una de las características del lenguaje humano es la ambigüedad; en particular, la ambigüedad puede afectar a palabras aisladas (ambigüedad léxica). Más concretamente, existen palabras que son ambiguas porque pueden tener más de un análisis morfológico; por ejemplo, la palabra del castellano *recuerdo* puede ser un nombre masculino singular o la primera persona singular del presente de indicativo del verbo *recordar*; por tanto al analizarla morfológicamente se obtendrían dos resultados. Los pasos, a grandes rasgos, que sigue este proceso son:

- Cuando únicamente se ha leído *rec-* todavía hay muchas posibilidades, ya que puede ser, entre otras, cualquier forma de las palabras *recorrer*, *recuperación*, *recurso*, *recordar* o *recuerdo*. Como mucho, se puede decir que el lema empieza por `rec`.
- Cuando se ha leído *recu-* el número de posibilidades se hace más pequeño y puede ser sólo una forma de *recuperación*, *recurso*, *recordar* o *recuerdo*, pero entre los análisis ya no hay más prefijo común y sólo se sabe que el lema empieza por `rec`.
- Cuando se ha leído *recue-* el abanico de posibilidades reduce todavía más y únicamente puede ser una forma de *recordar* o de *recuerdo*, pero entre los análsis continua sin haber más prefijo común, por tanto no se puede avanzar más aunque se lea *recuer-* o *recuerd-*.
- Cuando se ha leído *recuerdo-* todavía no se puede avanzar una salida, ya que puede tratarse del plural del nombre, por tanto es necesario leer

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

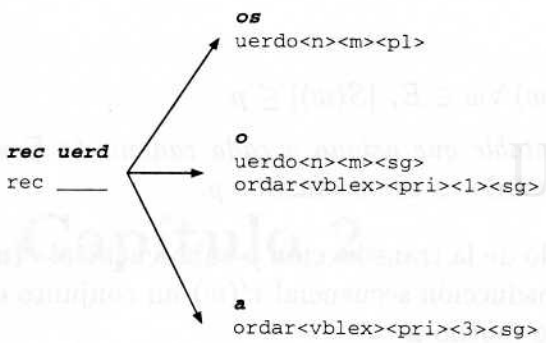


Figura 1.5: Ejemplo de retraso en la generación de salida de una transducción p-subsecuencial determinista

un carácter más, un carácter no alfabético, con lo que se descubre que la palabra se ha acabado. Es entonces cuando se deben generar las *colas* o *sufijos* correspondientes a los dos análisis válidos posibles de la palabra `recuerdo<n><m><sg>` y `recordar<vblex><pri><1><sg>`; estos sufijos son respectivamente `uerdo<n><m><sg>` y `ordar<vblex><pri><1><sg>`.

En la figura 1.5 se puede ver este proceso resumido, y donde se puede observar más claramente como la generación de la salida se retrasa hasta que se ha leído completamente la entrada, es decir, el final de la palabra de entrada (`os`, `o` o `a`). En este momento se obtienen las salidas parciales (`uerdo<n><m><pl>`, `uerdo<n><m><sg>|ordar<vblex><pri><1><sg>` y `ordar<vblex><pri><3><sg>`, respectivamente) que se tienen que concatenar al prefijo de la transducción obtenido hasta el momento (`rec`). Debido a este comportamiento es necesario definir un nuevo tipo de transducción que lo refleje: la *transducción subsecuencial.*

Definición 1.6 (Transducción p-subsecuencial) *Sea*

$$\tau : E \rightarrow 2^{\Gamma^*}$$

una función que asigna a cada cadena w en $E \subseteq \Sigma^*$ un conjunto de cadenas en Γ^* de manera que

$$\forall w \in \Pr(E), \quad |\tau(w)| \leq p$$

Es decir, asigna a cada cadena del lenguaje de entrada un conjunto que tiene como máximo p cadenas del lenguaje de salida.

La función τ es una transducción p-subsecuencial si existe una transducción secuencial computable τ' tal que:

$$\tau'(\epsilon) = \mu_0$$
$$\tau'(w\sigma) = \tau'(w)\zeta(w,\sigma) \quad \forall w \in \Pr(E), \sigma \in \Sigma$$

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

de forma que

$$\tau(w) = \tau'(w)S(w)\ \forall w \in E,\ |S(w)| \leq p$$

donde $S(w)$ es una función computable que asigna a cada cadena de E un conjunto de colas (sufijos) cuyo tamaño es como máximo p.

Por tanto, cada cadena resultado de la transducción p-subsecuencial $\tau(w)$ se calcula concatenando a una transducción secuencial $\tau'(w)$ un conjunto de sufijos $S(w)$, cuyo cardinal es como mucho p.

En particular, las transducciones secuenciales son un caso particular en el que $p = 1$ y $S(w) = \{\epsilon\}$ para todo $w \in E$.

Nótese que cualquier transducción $\tau : E \rightarrow 2^{\Gamma^*}$ se puede escribir siempre como la siguiente transducción p-subsecuencial trivial:

$$\begin{aligned}&\mu_0 = \epsilon,\\ &\zeta(w,\sigma) = \epsilon\ \ \forall w \in \Pr(E),\\ &S(w) = \tau(w)\ \forall w \in E\end{aligned}$$

sin embargo, en general interesará un modelado p-subsecuencial adelantado que afiance siempre el prefijo más largo posible, ya que así es posible organizar un sistema de traducción automática como una cadena de montaje en la que el prefijo afianzado por una etapa puede ser procesado cuanto antes por la etapa siguiente. Este tipo de transducciones p-subsecuenciales se denominan transducciones p-subsecuenciales adelantadas.

Definición 1.7 (Transducción p-subsecuencial adelantada) *Una transducción p-subsecuencial*

$$\tau(w) = \tau'(w)S(w)$$

es una transducción p-subsecuencial adelantada si

$$\begin{aligned}&\mu_0 = \mathrm{PCML}(\tau(E))\\ &\zeta(w,\sigma) = [\mathrm{PCML}(\tau(ww^{-1}E))]^{-1}[\mathrm{PCML}(\tau(w\sigma(w\sigma)^{-1}E))]\\ &S(w) = [\mathrm{PCML}(\tau(ww^{-1}E))]^{-1}\tau(w)\end{aligned}$$

donde $\mathrm{PCML}(X)$ es el prefijo común más largo (def. 1.3) del conjunto de transducciones X, y $w^{-1}E$ y $(w\sigma)^{-1}E$ son cocientes por la izquierda (def. 1.4).

Es decir, en la definición, los conjuntos

$$\begin{aligned}ww^{-1}E &= \{x \in E\ :\ w \in \Pr(x)\}\\ w\sigma(w\sigma)^{-1}E &= \{x \in E\ :\ w\sigma \in \Pr(x)\}\end{aligned} \tag{1.6}$$

son respectivamente los conjuntos de todas las cadenas de E que tienen como prefijos w y $w\sigma$.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

Capítulo 2

Uso de diccionarios en sistemas de traducción automática

2.1 Información presente en los diccionarios

Los diccionarios de un sistema de traducción automática pueden contener la información léxica necesaria sobre las lenguas a traducir en *unidades léxicas sencillas*, que son la representación del análisis morfológico de una palabra en lenguaje natural (forma superficial) individual, y en *unidades multipalabra*, que son la representación del análisis morfológico de un conjunto de formas superficiales que se consideran como una unidad o como la unión indivisible de un conjunto de formas superficiales.

2.1.1 Unidades léxicas sencillas

En TA se utilizan los diccionarios para realizar el tratamiento léxico de las lenguas involucradas; por tanto, los diccionarios básicamente consisten en listas que contienen información sobre cada palabra. En función de la naturaleza del diccionario, estas listas estarán compuestas por pares de naturaleza diferente.

- En el caso de los diccionarios morfológicos de la lengua origen y la lengua meta, los pares contienen la FS de la palabra y su FL correspondiente. Por ejemplo, en un diccionario morfológico del castellano asociaciones de este tipo son:

```
(rojo,rojo<adj><m><sg>)
(camiones,camión<n><m><pl>)
(actrices,actriz<n><f><pl>)
```


Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

- En un diccionario bilingüe los pares contienen la FL de la lengua origen y su traducción a la lengua meta correspondiente. Por ejemplo, en un diccionario bilingüe castellano–catalán se pueden encontrar entradas como las siguientes:

```
(rojo<adj><m><sg>,vermell<adj><m><sg>)
(camión<n><m><pl>,camió<n><m><pl>)
(actriz<n><f><pl>,actriu<n><f><pl>)
```

2.1.2 Unidades multipalabra

La mayor parte de un diccionario estará formado por entradas sencillas del tipo de las unidades léxicas sencillas, pero el lenguaje natural presenta ciertos fenómenos que se pueden intentar resolver a través de la inclusión de entradas un poco más complejas, de más de una palabra, en los diccionarios: las denominadas *unidades multipalabra*. Esto permite resolver algunos casos de *homografía* y *polisemia*, además de posibilitar el tratamiento correcto de las locuciones o expresiones idiomáticas.

Como ya se ha comentado, uno de los fenómenos que se puede resolver en cierta medida con la inclusión en el diccionario de unidades multipalabra es la homografía. Una palabra es *homógrafa* cuando puede pertenecer a más de una categoría léxica, y en función de esta categoría léxica tener más de una traducción, como es el caso de la palabra *así* en castellano, que se puede traducir al catalán como *així* si es un adverbio, o como *vaig agafar* si es un verbo (en concreto, la primera persona del pretérito indefinido de *asir*). Si en ciertos contextos solo puede aparecer una de las categorías léxicas, se pueden incluir en el diccionario entradas de más de una palabra que reflejen dichos contextos, unidades multipalabra (UMP) que se tomen como una única entrada, de forma que la traducción en estos casos siempre sea correcta. Por ejemplo, en el contexto *aun así*, la palabra *así* casi con total seguridad será un adverbio, y debe traducirse por *fins i tot així*. Por tanto si se incluye esta entrada en el diccionario se evitan algunos errores de traducción debidos a la homografía de *así*.

Otro problema que tiene solución en cierta medida a través de los diccionarios es el que se produce cuando el análisis de una FS sólo genera una FL pero el lema tiene más de una interpretación (polisemia) y cada interpretación tiene una traducción diferente. Por ejemplo, la palabra *dirección* se puede interpretar en el sentido de domicilio, localización, etc. y se traduce al catalán como *adreça* o como institución dirigente, orientación, rumbo, etc. que se traduce como *direcció*. Para solucionar este problema, algunas veces

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

se puede optar por la solución anterior: incluir UMP en las que la interpretación de la palabra sea única. En este ejemplo se pueden incluir las entradas *dirección postal* que se traduciría por *adreça postal*, y *Dirección General* que se traduciría por *Direcció General*.

Por último, fenómenos que se presentan con cierta frecuencia en el lenguaje natural son las locuciones, las construcciones modales y las frases hechas cuya traducción no se corresponde con la traducción individual de sus elementos. Para resolver este tipo de problemas también se puede recurrir a su inclusión como UMP. Por ejemplo, la expresión idiomática del castellano *hacer la vista gorda*, no se traduce al catalán como *fer la vista grossa*, sino como *fer els ulls grossos*; por tanto, se puede incluir en el diccionario una entrada que considere *hacer la vista gorda* como una unidad, que contendrá toda la flexión del verbo *hacer*.

Como se ha visto a través de los ejemplos, los diccionarios pueden incluir UMP que tendrán una FL simple, como son los ejemplos expuestos en los casos de homografía y polisemia, y UMP con una FL compuesta, como es el caso del ejemplo de la expresión idiomática. En la tabla 2.1 se pueden ver los análisis morfológicos completos de estos ejemplos. En la notación utilizada en dichos ejemplos, el carácter _ representa un espacio en blanco entre formas superficiales con un único análisis morfológico común, y el carácter + representa un espacio en blanco entre formas superficiales con análisis morfológicos individuales, pero que forman una unidad.

	Forma superficial	Análisis morfológico
FL simple	aún así	aún_así<adv>
	dirección postal	dirección_postal<n><f><sg>
	Dirección General	Dirección_General<n><f><sg>
FL compuesta	hacer la vista gorda	hacer<vblex><inf>+el<det><def><f><sg>+vista<n><f><sg>+gorda<adj><f><sg>

Tabla 2.1: Ejemplos de las diferentes formas léxicas correspondientes a UMP que puede contener un diccionario.

2.2 Descripción de los diccionarios

En el desarrollo de un sistema de traducción automática (TA) se debe tener una especial consideración con el tratamiento que se realizará de los diccionarios que contendrá el sistema ya que son uno de sus pilares básicos. Los

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

diccionarios expresan la parte léxica de la información lingüística, asociando *salidas* a *entradas*. Para expresar la información morfológica de las formas superficiales en los diccionarios se definen etiquetas especiales que se usan en las FL para indicar las características morfológicas. Ejemplos de etiquetas que indican características morfológicas en los diccionarios utilizados en esta tesis son:

```
!simbol <m>; # masculino
!simbol <pii>; # pretérito imperfecto de indicativo
!simbol <pri>; # presente de indicativo
!simbol <pp>; # participio
```

En principio, los diccionarios podrían ser meras listas que enumerasen por extensión correspondencias del tipo (FS, FL) o (FLLO, FLLM), pero los lingüistas suelen encontrar *regularidades* en la lengua. Para expresar dichas regularidades los diccionarios deben permitir:

1. Definir correspondencias entre fragmentos de cadenas de entrada y fragmentos de cadenas de salida, que se denominarán *parejas*. De esta manera se pueden expresar en el diccionario conceptos como "raíz", "desinencia" ("morfema" en general), es decir, divisiones con sentido lingüístico de las correspondencias. Por ejemplo, en un diccionario morfológico se pueden establecer las siguientes correspondencias:

   ```
   (gat:gato<n>) # Esta pareja establece la
   # correspondencia entre la raíz "gat" y el nombre
   # "gato" junto a la etiqueta que indica su categoría
   #  léxica
   (ábamos:ar<vblex><pii><1><pl>) # Esta correspondencia
   # puede significar que la desinencia "ábamos" es común
   # a muchos verbos acabados en "ar" en la primera persona
   # del plural del pretérito imperfecto de indicativo.
   ```

   En un diccionario bilingüe también se pueden encontrar regularidades entre la lengua origen y la lengua meta que se pueden expresar a través de este tipo de correspondencias. Por ejemplo, las siguientes correspondencias son susceptibles de aparecer en un diccionario bilingüe castellano–catalán:

   ```
   (pro:pro) # Con esta pareja se puede indicar que muchas
   # palabras con el prefijo "pro" en la lengua origen
   # (castellano en este caso) como proponer, producir, etc.
   # tienen el mismo prefijo en la lengua meta (catalán en
   ```


Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

```
# este caso), proposar, produir, etc.
(ción<n>:ció<n>) # Esta correspondencia puede significar
# que muchos nombres que terminan en "ción" en castellano,
# acaban en "ció" en catalán.
```

2. Definir grupos de correspondencias entre fragmentos que pueden aparecer en el mismo contexto. Estos grupos se denominarán *paradigmas* y se utilizan, entre otras cosas, para expresar la flexión de las palabras. Por ejemplo, el grupo de correspondencias que pueden aparecer al final de una palabra con género y número en un diccionario morfológico del castellano es:

```
[pl_fem] > (o:<m><sg>)|
           (a:<f><sg>)|
           (os:<m><pl>)|
           (as:<f><pl>);
```

3. Expresar de manera implícita la correspondencia entre dos FL de lenguas diferentes expresando únicamente la correspondencia *entre prefijos*. Es decir, no se utiliza una correspondencia directa del tipo (FLLO,FLLM), sino (pFLLO,pFLLM), donde pFLLO es un prefijo de la forma léxica de la lengua origen y pFLLM es un prefijo de la forma léxica de la lengua meta. De manera que si aparece la pareja (x, y) se entiende que se establece la correspondencia entre xw y yw para cualquier cadena w, siempre que xw y yw sean formas léxicas válidas. Esta operación permite construir diccionarios bilingües que no contienen toda la información léxica, sino únicamente aquella que es estrictamente necesaria para la traducción, que normalmente consiste en el lema y la categoría léxica. Esta opción sólo se usa en los diccionarios bilingües, ya que únicamente en la fase de transferencia de la traducción se establecen correspondencias entre formas léxicas.

4. Definir el alfabeto del lenguaje que servirá para reconocer las palabras desconocidas y para recortar el texto en palabras. Conocer el alfabeto del lenguaje es esencial para segmentar correctamente el texto en palabras, ya que no se puede dividir un texto en palabras considerando como palabras el vocabulario que contiene el diccionario, ya que las palabras que no constan en el diccionario quedarían segmentadas en caracteres individuales. Por ejemplo, la segmentación lógica de `zarkot(98)` es "`zarkot`" (palabra desconocida), "`(`", "`98`", "`)`", para ello es necesario saber que el carácter "`(`" no forma parte del alfabeto del lenguaje, y por

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

tanto representa el final de la palabra desconocida. También es importante debido a que hay palabras cuyo prefijo es a su vez una palabra del lenguaje. Por ejemplo, la palabra *casamiento* tiené como prefijo la palabra *casa*, de forma que si *casamiento* no consta en el diccionario, pero sí constan *casa* y *miento*, la palabra original quedaría segmentada en dos palabras. Esta opción sólo es útil en los diccionarios morfológicos de análisis, ya que en el resto del proceso de traducción las palabras desconocidas no son tenidas en cuenta ni es necesario identificar las fronteras entre palabras, que ya quedan claramente determinadas por el módulo de análisis morfológico.

2.3 Partes de los diccionarios

Los diccionarios están contenidos en ficheros de texto en los que los espacios en blanco, los tabuladores y los retornos de carro (líneas en blanco) pueden usarse libremente para hacer más legible el texto. Cualquier texto entre "#" y el final de la línea no se considera como parte del diccionario propiamente dicho y puede ser utilizado como comentario. Como puede observarse en los ejemplos del punto 1 de la sección anterior, se pueden usar para explicar la función de las distintas partes del diccionario.

En la clase de diccionarios que se detallan a continuación, las correspondencias no se proporcionan por extensión, sino que hay un trabajo lingüístico subyacente, de forma que se expresan regularidades tal como se ha visto en la sección anterior.

Antes de detallar como están construidos los diccionarios, conviene definir algunas nociones que son utilizadas de forma genérica en su construcción. A saber:

- Una transducción simple es una *pareja* tal y como se ha definido en el punto 1 de la sección anterior.

- La concatenación de una o más transducciones es también una transducción válida. La concatenación de transducciones es equivalente a la concatenación de las cadenas de entrada de las diferentes transducciones, dando lugar a una nueva cadena de entrada. De igual manera se forma la correspondiente nueva cadena de salida. Es decir, si se concatenan dos parejas (x, y) y (w, z), el resultado que se obtiene es el equivalente a una pareja formada por (xw, yz). Por ejemplo, la concatenación de las transducciones "`(gat:gato<n>)(as:<f><pl>)`", es equivalente a la transducción "`(gatas:gato<n><f><pl>)`".

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

2.3.1 Diccionarios morfológicos y bilingües

Un diccionario morfológico consta de cinco secciones:

1. **La sección del alfabeto del lenguaje.** En esta cadena se indican de forma explícita todos los caracteres en minúsculas que son susceptibles de aparecer en una forma superficial (FS), ya que sus correspondientes mayúsculas se obtienen de forma ímplicita. Esta cadena puede incluir rangos, de forma que no es necesario definir el alfabeto por extensión. La especificación correcta de esta sección es muy importante ya que la segmentación del texto (página 23) se realizará en función de los caracteres que se incluyan en esta sección del diccionario.

2. **La sección de declaración de símbolos,** donde los símbolos que representan las características morfológicas (tales como `<f>` o `<sg>`) se declaran explícitamente y son considerados en las transducciones como un único símbolo del alfabeto.

3. **La sección de paradigmas,** donde se declaran los paradigmas de flexión. Cuando un conjunto de lemas en el diccionario comparte un patrón de flexión común, se puede asignar un nombre a este patrón declarándolo entre corchetes (como `[verbs_en_ducir]` para los verbos en castellano como *traducir*, *producir*, *inducir*, etc.) y declararlo anticipadamente para poder ser utilizados en el diccionario. Un paradigma establece una lista de relaciones entre entradas y salidas que pueden aparecer alternativamente en un determinado contexto. Los paradigmas se pueden anidar indefinidamente; es decir, los paradigmas definidos pueden utilizarse para definir nuevos paradigmas. Por ejemplo, en la figura 2.1 se puede ver como para construir el paradigma de flexión de los verbos de la primera conjugación del castellano, se utilizan los paradigmas de presente e imperfecto de indicativo, definidos anteriormente. Un paradigma da un nombre a un conjunto de transducciones alternativas; un nombre de paradigma, por ejemplo "`[pi1]`", es siempre una transducción válida. En cualquier caso se deben respetar dos restricciones: todos los símbolos utilizados han de estar declarados y los paradigmas tienen que estar definidos (por esta razón los paradigmas se pueden anidar pero no son recursivos).
   Nótese que los paradigmas no necesariamente se utilizan para describir finales (sufijos), ya que pueden situarse en cualquier lugar de una transducción.

4. **La sección de unidades léxicas** empieza con el literal "`%dic`" y consiste en un gran paradigma que contiene todas las unidades léxicas del

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

```
!alfabet a-zçáéíóúàèìòùäëïöüâêîôûñ
#etiquetas morfológicas
!simbol <n>;                    !simbol <pri>;
!simbol <f>;                    !simbol <1>;
!simbol <m>;                    !simbol <2>;
!simbol <sg>;                   !simbol <3>;
!simbol <pl>;                   !simbol <pii>;
!simbol <vblex>;                !simbol <pp>;
!simbol <inf>;                  !simbol <ger>;
!simbol <num>;

#paradigma de género y número
[p4term] > (o:<m><sg>) | (os:<m><pl>)
         | (a:<f><sg>) | (as:<f><pl>);

#paradigma del presente de indicativo
[pi1] > (s:<2><sg>)   | (\0:<3><sg>)
      | (mos:<1><pl>) | (n:<3><pl>);

#paradigma del imperfecto de indicativo
[ii1] > (a:<1><sg>)   | (as:<2><sg>) | (a:<3><sg>)
      | (ais:<2><pl>) | (an:<3><pl>);

#paradigma de los verbos de la 1ª conjugación
[V1C] > (ar:ar<vblex><inf>) | (ando:ar<vblex><ger>)
      | (ad:ar<vblex><pp>)[p4term] | (o:ar<vblex><pri><1><sg>)
      | (a:ar<vblex><pri>)[pi1] | (áis:ar<vblex><pri><2><pl>)
      |(ab:ar<vblex><pii>)[ii1] | (ábamos:ar<vblex><pii><1><pl>);

[numeros]=[0-9]+([.,][0-9]+)? <num>; #etiqueta los números

#diccionario
%dic
(cant:cant)[V1C];                   #cantar
(gat:gato<n>)[p4term];              #gato
(dej:dej)[V1C](_caer:_caer);        #unidad multipalabra "dejar caer"
%final
[numeros]
```

Figura 2.1: Ejemplo de diccionario morfológico de castellano

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

diccionario. Cualquier transducción válida puede ser una entrada del diccionario; por ejemplo, el lingüista puede haber escogido formar un paradigma "`[vestir]`" en el diccionario morfológico con todas las formas del verbo irregular *vestir*; la entrada del diccionario para *vestir* puede ser simplemente el nombre del paradigma, y podría ser reutilizado para definir otras entradas que comparten el mismo patrón de flexión como puede ser *revestir* de la siguiente forma "`(re:re)[vestir]`" o *investir* como "`(in:in)[vestir]`". En la figura 2.1 se puede ver como se realizan las entradas del verbo *cantar* y del nombre *gato*.

5. **La sección de expresiones regulares** empieza con el literal "`%final`" y es la sección del diccionario donde se especifican las expresiones regulares que pertenezcan al lenguaje, como pueden ser los números. Las expresiones regulares se definen en la sección de paradigmas, y se utilizan posteriormente en esta sección del diccionario.

Para validar un diccionario morfológico de estas características se utiliza la gramática de la figura 2.2, donde los símbolos se definen como:

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

cadena: es una secuencia de caracteres que puede incluir rangos como "a-z".

etiqueta: es una secuencia de caracteres alfabéticos y números encerrada entre "<" y ">".
("<" [a-zá-úA-ZÁ-Ú0-9] [a-zá-úA-ZÁ-Ú0-9_]*">").

nompara: (nombre de paradigma) es un identificador (secuencia de caracteres y números que debe comenzar por un carácter) encerrado entre corchetes.
("[" [a-zá-úA-ZÁ-Ú] [a-zá-úA-ZÁ-Ú0-9_]*"]").

elem: es un elemento con significado especial, que puede ser un "\0" (símbolo vacío) o un "_" (un espacio en blanco).

carnorm: (carácter normal) es cualquier carácter alfanumérico o que se considere parte de una forma superficial.
([a-zá-úA-ZÁ-Ú0-9.·'ïÏüÜàèòÀÈÒ]).

caresp: (carácter especial) cualquier caracter que no sea una alfanumérico, o bien una secuencia de escape como por ejemplo "\(". Los caracteres que es necesario representar con una secuencia de escape son aquellos que se utilizan para la construcción del diccionario ("(", "[", ":", etc).

y las acciones que se realizan son:

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Alfabeto: describe el alfabeto de las palabras de la lengua origen mediante una **cadena**.

Simbolo: permite declarar cero o más símbolos mediante **etiquetas**, separados por un punto y coma. Estos símbolos se declaran al principio del diccionario para poder ser utilizados posteriormente.

UnPara: (un paradigma) declara paradigmas que empiezan por un nombre (**nompara**) y continuan de dos formas distintas: como una secuencia de entradas separadas por "|" y terminada en punto y coma, o bien como una expresión regular.

Dic: una vez terminada la parte de declaraciones, comienza el diccionario propiamente dicho, que puede tener una o dos secciones. La primera sección está formada por el vocabulario del diccionario y es obligatoria; consta de una o más entradas, separadas por puntos y coma. La segunda sección es opcional y se utiliza para reconocer entradas especiales como por ejemplo los números o los símbolos de puntuación.

Entrada: es una concatenación de una o más transducciones. Una transducción puede ser una referencia a un paradigma declarado previamente o bien una pareja, que está formada por un par de secuencias, una correspondiente a la entrada y otra correspondiente a la salida. Cada secuencia está compuesta por la concatenación de uno o más elementos simples, que a su vez pueden ser: una **etiqueta** correspondiente a un símbolo, un **carnorm**, un "+" o un elemento especial (**elem**).

Expr: genera expresiones regulares del estilo de las utilizadas por los programas de UNIX, seguidas de una etiqueta correspondiente a un símbolo que debe haberse declarado previamente. Los elementos básicos de estas expresiones regulares son **carnorm** o **caresp**.

La gramática utilizada para construir un diccionario bilingüe es idéntica a la descrita anteriormente, salvo en que no tienen la sección para declarar el alfabeto del lenguaje debido a que el módulo que utiliza este diccionario no lee formas superficiales, sino únicamente formas léxicas que ya están delimitadas.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Ejemplos de entradas en un diccionario bilingüe castellano–catalán son los siguientes:

```
(comer<vblex>:menjar<vblex>);
(calle<n><m>:carrer<n><f>)
```

Estas gramáticas las utiliza un compilador para generar un transductor a partir de los diccionarios (véase capítulo 4).

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Volver al Indice/Tornar a l'index

Diccio	→	Alfabeto Simbolo Parad Dic
Alfabeto	→	`!alfabet` **cadena**
Simbolo	→	`!simbol` **etiqueta** ; Simbolo
Simbolo	→	
Parad	→	UnPara Parad
Parad	→	
UnPara	→	**nompara** > ComPara Entrada ;
UnPara	→	**nompara** = Expr
ComPara	→	ComPara Entrada |
ComPara	→	
Dic	→	`%dic` ComDic ComEx
ComEx	→	`%final` ComDic
ComEx	→	
ComDic	→	ComDic ; Entrada
ComDic	→	Entrada
Entrada	→	Transduc Entrada
Entrada	→	Transduc
Transduc	→	Pareja
Transduc	→	**nompara**
Pareja	→	(Secuencia : Secuencia)
Secuencia	→	Secuencia ElemSim
Secuencia	→	ElemSim
ElemSim	→	**elem**
ElemSim	→	**etiqueta**
ElemSim	→	**carnorm**
ElemSim	→	+
Expr	→	ExpReg **etiqueta** ;
ExpReg	→	ExpReg ExpSim
ExpReg	→	ExpSim
ExpSim	→	(ExpReg) Repe
ExpSim	→	[Serie] Repe
ExpSim	→	Uncarac
Serie	→	Uncarac Serie
Serie	→	Uncarac
Repe	→	*
Repe	→	+
Repe	→	?
Repe	→	
Uncarac	→	**caresp**
Uncarac	→	**carnorm**

Figura 2.2: Gramática utilizada para validar y procesar los diccionarios construidos en esta tesis.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

Capítulo 3

Transductores de estados finitos

3.1 Transductores secuenciales deterministas

Para implementar las transducciones secuenciales de la definición 1.5 es necesario redefinir de manera realizable[1] la función $\zeta(w, \sigma)$ que, tal como está definida, tiene el argumento $w \in \Sigma^*$, que puede tomar un número infinito de valores y requeriría una tabla infinita y, por tanto, irrealizable.

La solución clásica es que $\zeta(w, \sigma)$ se calcule usando un conjunto subyacente de estados:

$$\zeta(w, \sigma) = \lambda(q(w), \sigma) \tag{3.1}$$

donde $q : \Sigma^* \to Q$ es una función que asigna a cada cadena w de Σ^* un estado $q(w)$ de un conjunto finito Q de estados. De esta forma la tabla λ es finita.

El siguiente paso es calcular la función $q : \Sigma^* \to Q$ de manera realizable; para ello, se define recursivamente como sigue:

$$\begin{aligned} q(\epsilon) &= q_I \\ q(w\sigma) &= \delta(q(w), \sigma) \quad \forall w \in \Sigma^*, \sigma \in \Sigma \end{aligned} \tag{3.2}$$

donde

- q_I es el estado inicial
- $\delta : Q \times \Sigma \to Q$ es la función que calcula el siguiente estado.

Además es conveniente definir un conjunto $F \subseteq Q$ de estados de aceptación para delimitar el rango de la transducción τ a un subconjunto propio de Σ^*, que define el lenguaje que se acepta a la entrada. De esta forma, solo se valida la transducción $\tau(w)$ si $q(w) \in F$.

[1] Es decir, con recursos finitos.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

La siguiente definición describe los transductores secuenciales (Mohri 1997; Oncina et al. 1993) deterministas:

Definición 3.1 (Transductor secuencial de estados finitos) *Un transductor secuencial de estados finitos se define como*

$$T = (Q, \Sigma, \Gamma, \delta, \lambda, q_I, F) \tag{3.3}$$

donde

- Q *es un conjunto finito de estados;*
- Σ *es un alfabeto de entrada;*
- Γ *es un alfabeto de salida;*
- $\delta : Q \times \Sigma \rightarrow Q$ *es la función de transición que proporciona el estado al que se llega desde cada estado con cada símbolo de entrada;*
- $\lambda : Q \times \Sigma \rightarrow \Gamma^*$ *es la función de salida que proporciona la cadena de salida que se asocia a las transiciones que salen de cada estado con cada símbolo de entrada;*
- $q_I \in Q$ *es el estado inicial;*
- $F \subseteq Q$ *es el conjunto de estados de aceptación.*

De esta forma las transducciones de la definición 1.5 se obtienen a través de las funciones δ y λ del transductor secuencial, ya que δ cambia de estado con cada símbolo del alfabeto de entrada y λ genera un cadena de salida parcial asociada a un estado y un símbolo de entrada ($\zeta(w, \sigma)$).

3.2 Transductores p-subsecuenciales deterministas

Como se ha visto en la sección 1.6, los transductores secuenciales tienen limitaciones, ya que para cada cadena del lenguaje de entrada sólo pueden generar una salida, mientras que en el lenguaje puede haber más de una salida válida. Por este motivo, conviene formalizar también en términos de estados finitos las transducciones subsecuenciales de la definición 1.6 y definir una clase un poco más general de transductores: los transductores p-subsecuenciales.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Definición 3.2 (Transductor p-subsecuencial de estados finitos) *Un transductor de estados finitos p-subsecuencial se define como*

$$T = (Q, \Sigma, \Gamma, \delta, \lambda, q_I, F, \psi) \tag{3.4}$$

donde

- Q *es un conjunto finito de estados;*
- Σ *es un alfabeto de entrada;*
- Γ *es un alfabeto de salida;*
- $\delta : Q \times \Sigma \to Q$ *es la función de transición que devuelve el estado al que se llega desde cada estado dado un símbolo de entrada;*
- $\lambda : Q \times \Sigma \to \Gamma^*$ *es la función de salida que devuelve la cadena de salida que se asocia a cada transición que parte de un estado dado un símbolo de entrada;*
- $q_I \in Q$ *es el estado inicial;*
- $F \subseteq Q$ *es el conjunto de estados de aceptación;*
- $\psi : F \to 2^{\Gamma^*}$ *es la función que asigna a cada estado de aceptación un conjunto de colas que se concatenará a la cadena de salida cuando se llegue a dicho estado de aceptación y se haya procesado totalmente la entrada, tal que* $\max_{q \in F} |\psi(q)| = p$*, es decir, p es el número máximo de transducciones posibles de cualquier entrada.*

De esta forma las transducciones de la definición 1.6 se obtienen a través de las funciones δ, λ y ψ del transductor p-secuencial, ya que δ cambia de estado con cada símbolo del alfabeto de entrada y λ genera un cadena de salida parcial asociada a un estado y un símbolo de entrada ($\zeta(w, \sigma)$) y ψ genera el conjunto de sufijos a concatenar ($S(w)$).

Este tipo de transductores también son deterministas respecto al alfabeto de entrada Σ. Los transductores de Oncina et al. (1993) son transductores 1-subsecuenciales; estos autores usan una definición basada en *aristas*, de forma que utilizan un conjunto de aristas ($Q \times \Sigma \times \Gamma^* \times Q$) en lugar de las funciones δ y λ. En esta tesis se ha optado por la notación con δ y λ para los transductores deterministas ya que es más sencillo presentar así su paralelismo con las transducciones de las definiciones 1.5 y 1.6.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

3.3 Construcción estándar de transductores deterministas

La construcción estándar de un transductor determinista respecto a la entrada consiste en la construcción de un transductor de prefijos, que genera para cada prefijo de la palabra el prefijo común más largo de las transducciones de todas las palabras que comparten dicho prefijo (Oncina et al. 1993).

Los datos D para construir este transductor vienen dados en forma de un conjunto finito de parejas del tipo $(w, w') \in \Sigma^* \times \Gamma^*$, donde $w' \in \tau(w)$, es decir, el dominio finito sobre el que se define el transductor será $D = \{(w, w') : w \in \Sigma^*, w' \in \Gamma^*, w' \in \tau(w)\}$. Por tanto, se dispone de un vocabulario de entrada o muestra $E = \{w : (w, w') \in D\}$ en función del cual se definen los estados que forman el transductor:

Definición 3.3 (Transductor de prefijos adelantados) *El transductor de prefijos adelantados para* $E = \{w : (w, w') \in D\}$ *se define como*

$$T = (Q, \Sigma, \Gamma, \delta, \lambda, q_I, F, \psi) \tag{3.5}$$

donde

- *Los estados del transductor de prefijos son:*

$$Q = \Pr(E) \tag{3.6}$$

  es decir, $Q = \{p \in \Sigma^* : p \in \Pr(w), \ \ w \in E\}$.

- *El estado inicial se elige* $q_I = \epsilon$.

- *La función de transición o siguiente estado viene dado por* $\delta(x, \sigma) = x\sigma$ *si* $x\sigma \in Q$*; en cualquier otro caso la función no está definida.*

- *El conjunto de estados finales viene dado por el conjunto de palabras completas que el transductor reconoce, es decir,* $F = E$.

- *La función salida del transductor se define como*

$$\lambda(x, \sigma) = [\mathrm{PCML}(\tau(E_x))]^{-1}[\mathrm{PCML}(\tau(E_x\sigma))] \tag{3.7}$$

  donde $\tau(E_x) = \cup_{w \in \Sigma^*, xw \in E} \tau(xw)$ *representa las transducciones de las cadenas de* E *que empiezan por el prefijo* x*. De esta forma se asigna a cada estado el prefijo común más largo que lleva hasta el momento menos el prefijo común más largo (*PCML*, definición 1.3) que llevaba hasta el estado anterior*[2].

[2]Esta definición sólo es válida si $\mathrm{PCML}[\tau(E)] = \epsilon$, en otro caso en las transiciones del estado inicial se emite $\mathrm{PCML}[\tau(E)]$.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

- *Por último, la función ψ se define como*

$$\psi(w) = [\mathrm{PCML}(\tau(E_w))]^{-1}\tau(w) \tag{3.8}$$

de forma que sólo existe si $w \in E$ y $(w, w') \in D$ con $w' = \tau(w)$.

De forma que

- si $|\psi(w)| = 0 \ \ \forall w \in E$, se trata de un transductor secuencial
- si $0 \leq |\psi(w)| \leq p \ \ \forall w \in E$ se trata de un transductor p-subsecuencial. Los transductores con $p = 1$ se denominan simplemente subsecuenciales (Oncina et al. 1993).

De esta forma se obtiene un transductor en el que las cadenas de salida asociadas a las transiciones que parten de un estado son las cadenas más largas que se pueden emitir en función de la entrada leída; es decir, se han *adelantado* los prefijos de las cadenas de salida todo lo posible.

La definición de transductor de prefijos adelantados expuesta también es utilizada por Oncina et al. (1993) para construir un transductor canónico subsecuencial pero utilizando una notación basada en un conjunto de aristas, como se ha comentado anteriormente.

El transductor que se obtiene con esta construcción no es mínimo; para minimizarlo se tienen que aplicar técnicas (Mohri 1994) basadas en la equivalencia de estados que se verán en la sección 5.3. De esa forma se obtendrá un transductor mínimo equivalente, es decir, que reconocen el mismo lenguaje y para cualquier cadena de ese lenguaje, ambos transductores generan la misma salida.

3.4 Problemática de los transductores deterministas

Los transductores p-subsecuenciales de estados finitos presentan algunos inconvenientes:

- Como se vio en la sección 1.6, si una transducción únicamente se valida cuando aparece un determinado sufijo, es necesario retardar la salida. Por ejemplo, en la figura 1.5 se puede ver como la generación de la salida únicamente se puede realizar al acabar de leer la palabra de entrada, ya que, hasta ese momento, el transductor no dispone de suficiente información para generar ninguna salida con seguridad, únicamente el prefijo común más largo a todas ellas $\mathrm{PCML}(\{\tau(\text{recuerdo}, \text{recordar})\}) = \text{rec}$.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

En la sección 5.4, dedicada a experimentos, se podrán ver datos sobre el tamaño típico de las colas de sufijos de transductores p-subsecuenciales en diccionarios reales.

- La adición de una nueva transducción al conjunto de las transducciones que ya realiza el transductor obliga a realinear todas las entradas y salidas, ya que se pueden haber producido cambios en el prefijo común más largo de la salida asociado a cada prefijo de entrada.

Estos problemas se evitan si se opta por un transductor determinista respecto al par *entrada–salida*, pero indeterminista respecto a la entrada. En este tipo de transductores se pueden mantener vivas durante el procesamiento de la entrada varias hipótesis que se validan o rechazan según se avanza en la transducción. Por ejemplo, si un transductor de este tipo contiene las cadenas de entrada `voz`, `voy` y `veto`, cuyas salidas son `voz<n><f><sg>`, `ir<vblex><pri><1><sg>` y `veto<n><m><sg>` respectivamente, y se lee de la entrada la palabra *voz*, al leer de la entrada el carácter *v* mantendrá vivas dos posibles transducciones, `v` y `i`, ya que con un mismo carácter de entrada se generan dos carácteres de salida diferente (es decir, el transductor es determinista respecto al par formado por el carácter de entrada y por el de salida, pero indeterminista respecto al carácter de entrada considerado individualmente); cuando se lee el carácter *o* todavía se mantienen vivas dos posibles hipótesis de transducción, `vo` y `ir`; por último al leer el carácter `z` la única transducción posible es `voz<n><f><sg>`, descartando la transducción `ir` que estaba construyendo.

Por otra parte, añadir una nueva transducción a un transductor determinista respecto al par *entrada–salida* es más sencillo, ya que la adición de caminos nuevos es directa. Si se quiere un transductor mínimo existen algoritmos (sección 6.4) que permiten añadir una transducción manteniendo el transductor mínimo.

Sin embargo, los transductores deterministas respecto al par *entrada–salida* también presentan un inconveniente: se basan en un alineamiento preestablecido de la entrada y la salida. Esto significa que el transductor resultante no sólo dependerá de las correspondencias establecidas entre la entrada y la salida, sino también del alineamiento establecido, ya que este alineamiento influye directamente en el determinismo (respecto al par carácter de entrada y carácter de salida, como se ha visto en el ejemplo anterior) de estos transductores.

En la figura 3.1 se puede observar un transductor determinista respecto al par *entrada–salida* y en la figura 3.2 un transductor subsecuencial equivalente en el que se han simplificado las cadenas de estados para que su lectura resulte más sencilla; es decir, en lugar de poner toda la secuencia de estados con

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

sus transiciones respectivas carácter de entrada a carácter de entrada, se han agrupado en un estado con una transición, cuyas etiquetas son el resultado de concatenar las etiquetas de las transiciones de los estados agrupados. Estos transductores representan el diccionario formado por las siguientes parejas:

```
(retáis:retar<vblex><pri><2><pl>);
(retamos:retar<vblex><pri><1><pl>);
(retamos:retar<vblex><ifi><1><pl>);
(reto:retar<vblex><pri><1><sg>);
(reta:retar<vblex><pri><3><sg>);
(reto:reto<n><m><sg>);
(recordáis:recordar<vblex><pri><2><pl>);
(recordamos:recordar<vblex><pri><1><pl>);
(recordamos:recordar<vblex><ifi><1><pl>);
(recuerdo:recordar<vblex><pri><1><sg>);
(recuerda:recordar<vblex><pri><3><sg>);
(recuerdo:recuerdo<n><m><sg>);
(contáis:contar<vblex><pri><2><pl>);
(contamos:contar<vblex><pri><1><pl>);
(contamos:contar<vblex><ifi><1><pl>);
(cuento:contar<vblex><pri><1><sg>);
(cuenta:contar<vblex><pri><3><sg>);
(cuento:cuento<n><m><sg>);
(tentáis:tentar<vblex><pri><2><pl>);
(tentamos:tentar<vblex><pri><1><pl>);
(tentamos:tentar<vblex><ifi><1><pl>);
(tiento:tentar<vblex><pri><1><sg>);
(tienta:tentar<vblex><pri><3><sg>);
(tiento:tiento<n><m><sg>)
```

Como se puede observar, el transductor determinista respecto al par *entrada–salida* puede ir avanzando a través de más de una transducción, que se validará o no en función de la entrada que se vaya procesando. Por ejemplo, se quiere procesar la cadena de entrada *tienta*; inicialmente se lee el carácter *t* y el transductor mantiene una única hipótesis de tranducción viva, `t`; al leer el carácter *i* el transductor mantiene dos posibles transducciones vivas, `ti` y `te`; al leer la secuencia de caracteres *ent* el transductor construye las dos posibles transducciones `tient` y `tent`; finalmente cuando se lee el carácter *a*, se completa la transducción `tentar<vblex><pri><3><sg>` y se descarta la transducción `tient`.

El correspondiente transductor subsecuencial sólo puede mantener una transducción viva, y por tanto debe retrasar su salida (en ocasiones, hasta ver el final de la palabra como en el ejemplo anterior con la cadena de entrada

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

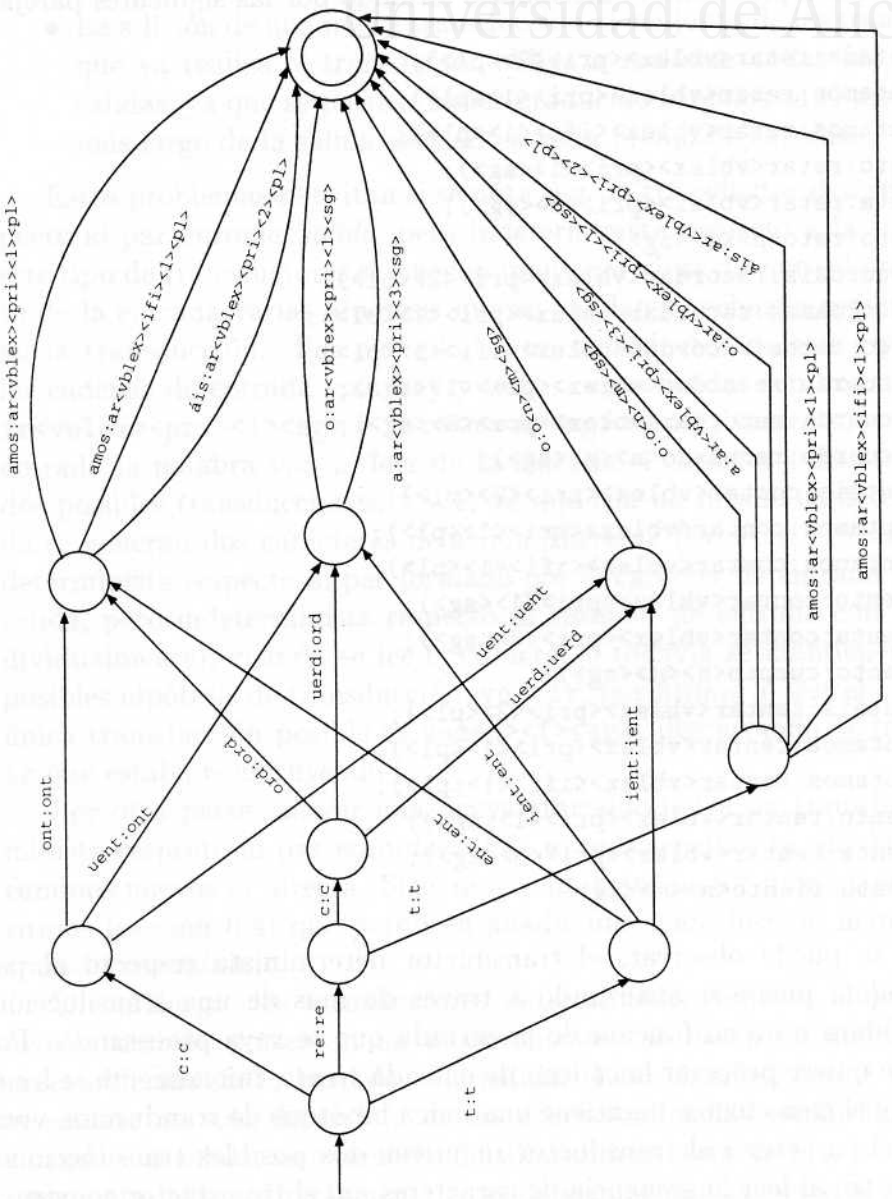


Figura 3.1: Transductor determinista respecto al par *entrada–salida* (con las cadenas de estados simplificadas para una mejor lectura)

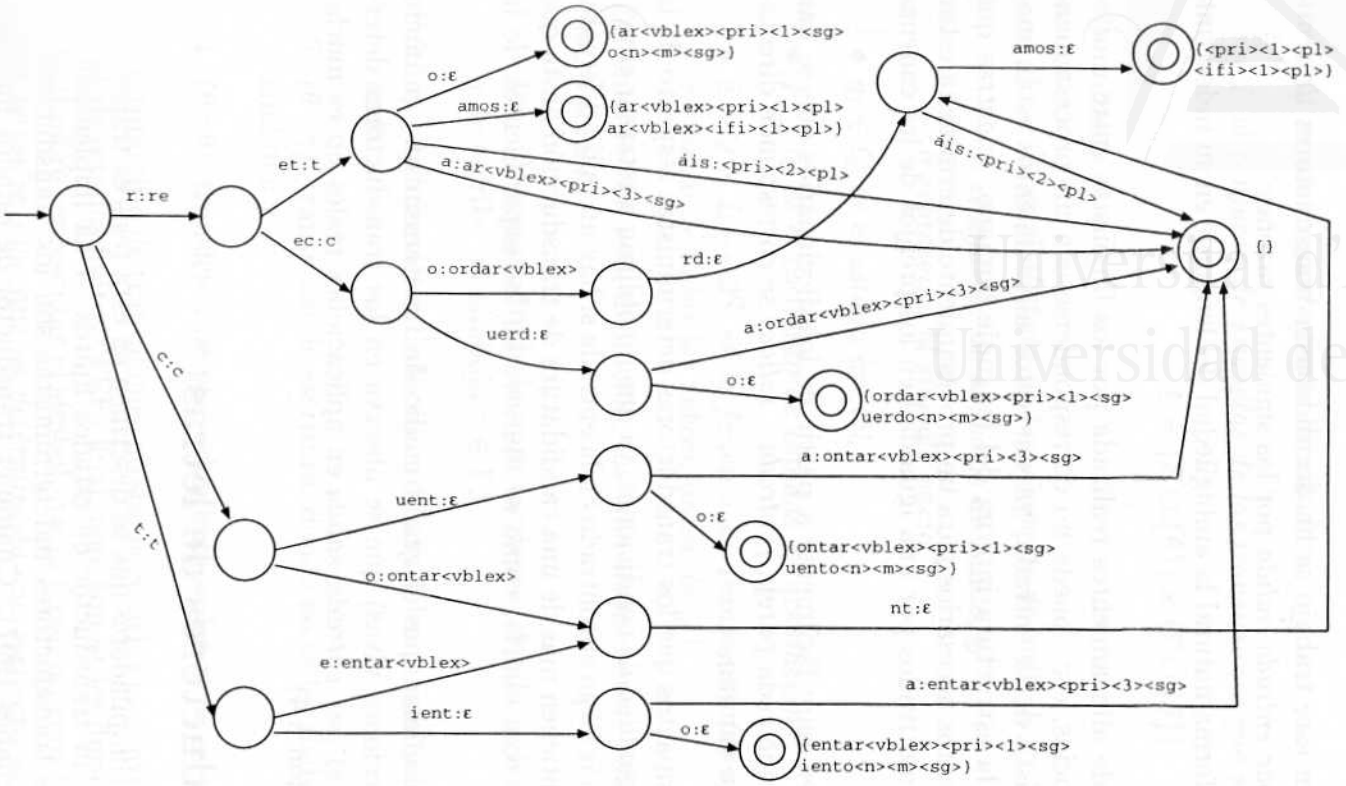


Figura 3.2: Transductor determinista subsecuencial (con las cadenas de estados simplificadas para una mejor lectura)

Universitat d'Alacant
Universidad de Alicante

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

tienta).

En resumen, en este trabajo se ha decidido usar transductores *deterministas repecto al par entrada–salida* por las siguientes razones:

1. Integran de forma natural la ambigüedad del lenguaje en su indeterminismo.

2. La elección de alineamientos realizada por los lingüistas entre morfemas, desinencias, etc. puede no corresponderse con un procesamiento determinista de la entrada, ya que estos alineamientos están motivados por la estructura interna del lenguaje humano, mientras que los alineamientos necesarios para un procesamiento determinista están únicamente motivados por una igualdad en los prefijos de las cadenas de entrada.

3. Se construyen muy fácilmente a partir de los diccionarios que se han descrito, ya que cada pareja *(entrada : salida)* se corresponde directamente con un camino.

4. Son más compactos que los transductores deterministas respecto de la entrada en igualdad de condiciones, ya que no obligan a retrasar el prefijo común más largo encontrado en la entrada si hay un infijo diferente, sino que mantienen más de una candidatura de transducción hasta que es imposible continuarla, como se muestra en los experimentos de la sección 5.4.

También cabe señalar que el grado medio de indeterminismo (número medio de transducciones vivas) que se observa en los transductores deterministas respecto al par *entrada–salida* en aplicaciones reales no es mucho mayor que 1 (sección 5.4).

3.5 Transductores de letras

El objetivo de los compiladores que se describen en esta tesis es transformar un diccionario en un transductor de estados finitos. Una implementación conveniente de los transductores indeterministas son los transductores de letras (Roche y Schabes 1997). Cualquier transductor de estados finitos se puede convertir en un transductor de letras equivalente. Un transductor de letras (TL) se define como

Definición 3.4 (Transductor de letras) *Un transductor de letras es*

$$T = (Q, L, \delta, q_I, F) \tag{3.9}$$

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

donde

- *Q es un conjunto finito de estados;*
- *L el conjunto de etiquetas de las transiciones que se define como*

$$L = (\Sigma \cup \{\theta\}) \times (\Gamma \cup \{\theta\}) \tag{3.10}$$

  donde

  - *Σ es el alfabeto de los símbolos de entrada,*
  - *Γ el alfabeto de símbolos de salida, y*
  - *θ representa el símbolo vacío.*

- *$q_I \in Q$ es el estado inicial;*
- *$F \subseteq Q$ es el conjunto de estados de aceptación;*
- *$\delta : Q \times L \rightarrow 2^Q$ es la función de transición (donde 2^Q representa el conjunto de todos los subconjuntos de Q)*

De acuerdo con esta definición, las transiciones entre estados pueden ser de cuatro tipos:

1. $(\sigma : \gamma)$, transición de lectura-escritura, donde se lee un símbolo $\sigma \in \Sigma$ y se escribe un símbolo $\gamma \in \Gamma$;
2. $(\sigma : \theta)$, transición de lectura, donde se lee un símbolo pero no se escribe nada;
3. $(\theta : \gamma)$, transición de escritura, donde no se lee nada pero se escribe un símbolo; y
4. $(\theta : \theta)$, transición de paso, donde ni se lee ni se escribe nada.

Este último tipo de transiciones no son necesarias ni convenientes en el transductor de estados finitos (TEF) final, pero son útiles durante la construcción. Es costumbre representar el símbolo vacío θ con un cero ("0"). Un transductor de letras se dice que es *determinista* con respecto al alfabeto L cuando se puede escribir $\delta : Q \times L \rightarrow Q$. Nótese que un transductor de letras puede ser determinista con respecto al alfabeto L pero ser indeterminista respecto al alfabeto de entrada Σ.

Los tipos de transiciones comentados tienen una clara relación con las denominadas operaciones de edición (Wagner y Fischer 1974):

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

- la transición del tipo $(\sigma : \gamma)$ equivale a una operación de sustitución;
- la operación de borrado equivale a una transición $(\sigma : \theta)$;
- y finalmente la operación de inserción equivale en el TL a una transición $(\theta : \gamma)$.

Los transductores de letras que se tratan en este documento se pueden representar gráficamente como se puede observar en el ejemplo de la figura 3.3, donde los círculos etiquetados como q_I, q_s y q_F son estados y las flechas representan las transiciones entre ellos, etiquetadas con un par de símbolos, de forma que $q_s \in \delta(q_I, (\sigma, \gamma))$. El estado inicial, q_I, como es costumbre, está marcado por una flecha sin origen; el estado $q_F \in F$ es un estado de aceptación, marcado con un doble círculo; el estado q_s es un estado intermedio.

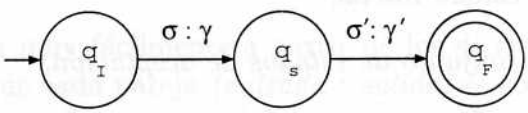


Figura 3.3: Ejemplo de representación gráfica de un transductor de letras

Una cadena $w' \in \Gamma^*$ se considera una transducción de una cadena de entrada $w \in \Sigma^*$ si hay al menos un camino[3] desde el estado inicial q_I a un estado de aceptación en F en el cual las etiquetas de las transiciones forman la pareja $(w : w')$ una vez concatenadas. En principio puede haber más de un camino para una transducción dada; esto debe evitarse y se evita parcialmente mediante la determinización (como se verá en el capítulo 5). Por otra parte, puede haber más de una salida válida para una cadena de entrada debido a la homografía, que se produce cuando una FS tiene más de una FL. Por ejemplo, la palabra *juego* puede ser tanto un verbo como un nombre cuando se analiza morfológicamente.

El algoritmo 3.1 describe el funcionamiento de un TL indeterminista respecto a la entrada como analizador morfológico. El proceso comienza situando el transductor en el estado inicial q_I, y construyendo una lista de pares vivos *estado–salida* $\mathcal{V}^{[1]}$. Cada par estado–salida contiene la salida construida hasta el momento, correspondiente a un camino recorrido dentro del transductor, y el último estado visitado en ese camino.

1. Inicialmente $\mathcal{V}^{[1]}$ contendrá el par (q_I, ϵ). A esta lista se añaden todos los estados a los que se puede llegar desde el estado de este par sin leer

[3] Un camino se define como una secuencia $s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \ldots \xrightarrow{a_{n-1}} s_n$, $s_i \in Q, a_i \in L \;/\; s_{j+1} \in \delta(s_j, a_j)$.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

ningún carácter (transiciones con θ en la lectura pero con otro símbolo en la escritura, de forma que este símbolo es añadido a la salida en curso). Esta operación se denomina clausura–ε de un par estado–salida y se describe en el algoritmo 3.2.

2. En cada nuevo paso se construye una nueva lista de pares estado–salida $\mathcal{V}^{[t+1]}$. Para ello se toma un carácter de la entrada, y para cada par (q, x) de $\mathcal{V}^{[t]}$, se buscan los estados a los que se llega desde cada q con dicho carácter, formando un conjunto $\mathcal{M}$ de pares vivos $(q', x\gamma)$, donde q' es cada estado al que se llega desde q al leer el carácter de la entrada, y $x\gamma$ es el resultado de concatenar a la salida parcial asociada a q el símbolo de salida γ correspondiente.

3. A continuación se construye $\mathcal{V}^{[t+1]}$ como la unión de los conjuntos obtenidos al aplicar la clausura–ε a cada elemento de $\mathcal{M}$; si no se ha llegado al final de la palabra se vuelve al paso 2.

4. Por último se comprueba si algún par estado–salida de la lista construida en el último paso, corresponde a alguno de los estados de aceptación del transductor, en cuyo caso se devuelve la salida contenida en ese par.

La definición general de transductor de letras es completamente paralela a la de autómata finito no determinista (AFN) y la de transductor de letras determinista, paralela a su vez a la de AFD; así, un transductor de letras puede ser determinizado y minimizado (respecto al alfabeto L) utilizando los algoritmos existentes para AFN y AFD (Hopcroft y Ullman 1979, p. 19–25, p. 68; van de Snepscheut 1993, p. 262–267).

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

Algoritmo 3.1: Análisis morfológico de una palabra

Entrada: $\sigma[1]\dots\sigma[t_n] \in \Sigma^*$
TLI $T = (Q, L, \delta, q_I, F)$
Salida: conjunto finito de transducciones válidas
Método:

1. Inicialización
   $\mathcal{V}^{[0]} \leftarrow \widehat{C}_\varepsilon((q_I, \epsilon))$,
   $t \leftarrow 0$
2. `Haz`
   $t \leftarrow t + 1$
   $\mathcal{M} \leftarrow \emptyset$
   $\mathcal{V}^{[t]} \leftarrow \emptyset$
   `Para todo` $(q, x) \in \mathcal{V}^{[t-1]}$
   `Para todo` $\gamma \in (\Gamma \cup \{\theta\})$
   $\mathcal{M} \leftarrow \mathcal{M} \cup \{(\delta(q, (\sigma[t], \gamma)), x\gamma)\}$
   `fin_para`
   `fin_para`
   `Para todo` $(q, x) \in \mathcal{M}$
   $\mathcal{V}^{[t]} \leftarrow \mathcal{V}^{[t]} \cup \widehat{C}_\varepsilon((q, x))$
   `fin_para`
   `mientras` $t < t_n$
3. Devuelve $\{x : (q, x) \in \mathcal{V}^{[t]} \wedge q \in F\}$

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Volver al Indice/Tornar a l'index

Universitat d'Alacant
Universidad de Alicante

Algoritmo 3.2: Realizar la clausura–ε a un par estado–salida

Entrada: un par estado–salida $(q, x) \in Q \times \Gamma^*$
Salida: conjunto de pares estado–salida $\widehat{C}_\varepsilon((q,x)) \subseteq 2^{Q \times \Gamma^*}$
Método:

1. Inicialización
   $\mathcal{N}^{[0]} \leftarrow \{(q,x)\}$,
   $\widehat{C}_\varepsilon((q,x)) \leftarrow \mathcal{N}^{[0]}$,
   $t \leftarrow 0$
2. `Haz`
   $t \leftarrow t+1$
   $\mathcal{N}^{[t]} \leftarrow \emptyset$
   `Para` todo $(q_i, x_i) \in \mathcal{N}^{[t-1]}$ haz
   `Para` todo $\gamma \in (\Gamma \cup \{\theta\})$ haz
   $\mathcal{N}^{[t]} \leftarrow \mathcal{N}^{[t]} \cup \{(\delta(q_i, (\theta, \gamma)), x_i\gamma)\}$
   `fin_para`
   `fin_para`
   $\widehat{C}_\varepsilon((q,x)) \leftarrow \widehat{C}_\varepsilon((q,x)) \cup \mathcal{N}^{[t]}$
   `mientras` $(\mathcal{N}^{[t]} - \widehat{C}_\varepsilon((q,x))) \neq \emptyset$
3. Devuelve $\widehat{C}_\varepsilon((q,x))$

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Universitat d'Alacant
Universidad de Alicante

Capítulo 4

Construcción de transductores a partir de diccionarios

4.1 Introducción

Siguiendo la separación clásica de algoritmos y datos, los módulos de análisis, transferencia y generación se basarán en programas genéricos que consultan los diccionarios correspondientes como fuentes de información. En esta última línea, la manera de realizar la consulta toma una relevancia especial, ya que los diccionarios llegan a alcanzar tamaños considerables. Si se considera un diccionario como una base de datos, la consulta de una palabra puede realizarse a través de una búsqueda (secuencial, binaria, etc.) de la palabra para analizarla o traducirla posteriormente, o bien seguir la palabra dentro del diccionario como si fuera un camino, es decir, letra a letra, de forma que se puede construir su análisis o traducción simultáneamente. Esta última opción requiere que el diccionario esté en un formato específico de grafo o árbol, normalmente un "árbol de recuperación" (*retrieval tree*) o "trie" cuya estructura habitual consiste en que cada nodo representa el prefijo de una palabra, para poder seguir los caminos que forman las palabras.

En esta tesis se ha optado por la segunda opción de consulta de un diccionario. Para ello se ha utilizado una solución basada en autómatas finitos debido a que son compactos (puede almacenarse un gran diccionario en relativamente poco espacio), rápidos (se puede obtener la información que almacenan con gran eficiencia) y están bien caracterizados teóricamente.

Este tratamiento de los diccionarios es aplicable a varias tareas que se realizan dentro del campo de la traducción automática:

- Análisis morfológico.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

- Consulta de un diccionario bilingüe para la traducción palabra por palabra.
- Generación morfológica.
- Aplicación de reglas derivadas de la ortografía de la lengua meta (básicamente contracciones, apostrofaciones y guionados), como por ejemplo:
  - En castellano, cuando aparece la secuencia de palabras *de el*, se debe aplicar una regla de contracción que genere *del*.
  - En catalán, cuando aparece una secuencia de palabras como *la oliva*, se debe aplicar una regla de apostrofación que genere *l'oliva*, o cuando aparece una secuencia de palabras como *digueu me*, se debe aplicar una regla de guionado que genere *digueu-me*.

El proceso de compilación como un transductor de estados finitos de un diccionario consta de varias etapas:

1. La **lectura del diccionario** (durante la cual, se construye un transductor).
2. La **minimización del transductor** obtenido en la fase anterior.
3. La **transcripción del transductor mínimo**, que consiste en almacenar el transductor obtenido en un fichero binario con un formato adecuado para su utilización eficiente.

Estas operaciones son aplicables a los dos tipos de transductores que se construyen a partir de un diccionario: transductores de letras y transductores p-subsecuenciales. Hay una cuarta operación que se puede realizar únicamente sobre los transductores de letras, el **mantenimiento incremental** del diccionario a través del transductor mínimo correspondiente (añadir o eliminar vocabulario), debido a que el transductor de letras se basa en un alineamiento explícito que no se modifica al añadir un nuevo par entrada–salida, mientras que el transductor p-subsecuencial debe volver a calcular el prefijo común más largo para cada estado (sección 3.3) ya que este puede haberse modificado con la nueva entrada.

Cabe destacar aquí que los transductores de letras tienen en esta tesis un tratamiento más perfeccionado que los transductores p-subsecuenciales debido a que estos últimos se han utilizado en esta tesis para establecer una comparación con los primeros, y no para ser utilizados realmente en una aplicación. Por ello, en la construcción de los transductores, que se expone

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

en la siguiente sección, se observará que el transductor de letras acepta cadenas que no pertenecen al vocabulario del diccionario, pero sí al lenguaje de entrada (son las palabras desconocidas), mientras que el transductor p-subsecuencial acepta exclusivamente las cadenas que están contenidas en el diccionario.

4.2 Construcción de un transductor de letras indeterminista a partir de un diccionario

En esta sección se describe como se transforma un diccionario con el formato descrito en la sección 2.3 en un transductor de letras indeterminista, que será minimizado y determinizado posteriormente.

En primer lugar se almacena el *alfabeto del lenguaje*, que será tratado cuando se haya construido el transductor correspondiente al resto del diccionario.

Las etiquetas declaradas en la primera sección del diccionario son leídas por el compilador, que las guarda en una tabla de símbolos y les asigna un identificador numérico, que será el utilizado en las transiciones del transductor. Si se declara una etiqueta dos veces, el compilador emite un error que indica qué etiqueta se ha duplicado y escribe la fila y la columna del fichero origen donde se encuentra por segunda vez dicha etiqueta.

A continuación empieza el proceso de la sección de paradigmas que son compilados en subtransductores que se integran para construir el transductor completo. Para procesar un *paradigma de flexión en primer lugar se* tratan las transducciones asociadas a dicho paradigma. Estas secuencias de caracteres se procesan de forma que se generan dos transiciones (una en el sentido de lectura y otra inversa[1]) entre dos estados por cada par de caracteres formado tomando un carácter de la parte izquierda y otro de la derecha, siguiendo el orden de lectura (de izquierda a derecha), hasta acabar con ambas cadenas. Las cadenas que forman una pareja se pueden alinear implícita o explícitamente. La opción de poder realizar alineamientos explícitos permite al lingüista expresar regularidades de la lengua, como se ha visto en la sección 2.2.

Si se ha utilizado el alineamiento explícito en una pareja, las cadenas de entrada y salida de la pareja tienen la misma longitud. Por ejemplo "`(am\0\0\0:amar<vblex>)`". El alineamiento debe conservarse en el transductor. Si ambas cadenas tienen longitud diferente y no contienen "\0",

[1]La transición inversa se crea y se guarda en otra estructura por motivos de rapidez en la ejecución de los algoritmos necesarios para minimizar el transductor.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

por ejemplo "(am:amar<vblex>)", entonces el compilador las alineará por la izquierda y se completa la más corta con símbolos vacíos "\0". Si, por alguna razón, una de las cadenas debe ser vacía, se instancia como un sólo "\0", por ejemplo "(\0:<3p><sg>)". Si se utilizan "\0" pero la longitud de las cadenas de entrada y salida no coinciden, como por ejemplo "(com\0\0:comer<vblex>)", el compilador asumirá que el diseñador del diccionario ha intentado realizar un alineamiento explícito pero se ha equivocado; de acuerdo con esto, el compilador emite un aviso a este efecto, ignora los "\0" y utiliza el alineamiento implícito anterior. De esta forma, las cadenas procesadas tienen que tener la misma longitud o no contener símbolos "\0".

Para representar gráficamente la secuencia de estados y transiciones construida con este proceso, se puede ver para el ejemplo de la pareja (ones:<m><pl>) el TL obtenido en la figura 4.1. Nótese que el símbolo vacío del diccionario "\0" se corresponde con el símbolo vacío θ del transductor.

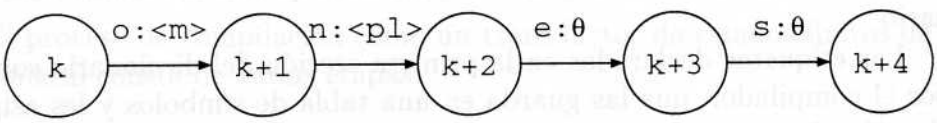


Figura 4.1: Secuencia de estados y transiciones correspondientes a la pareja (ones:<m><pl>).

Cuando un paradigma de flexión contiene más de una pareja (el caso más usual: se puede seguir más de un camino), todos los primeros estados de estas parejas se conectan con transiciones (θ:θ) a un recién creado estado inicial del paradigma, y todos los correpondientes estados finales de estas parejas se conectan a un también recién creado estado final con transiciones (θ:θ). Para ilustrar esta explicación, se muestra un ejemplo de paradigma de dos entradas y su transductor correspondiente en la figura 4.2, donde los estados k y $k+1$ son respectivamente el estado inicial y final del transductor correspondiente al paradigma.

A partir de los paradigmas correspondientes a las expresiones regulares se contruyen a su vez transductores que aceptan las cadenas correspondientes a dichas expresiones y las generan con su transducción correspondiente. La estructura de un transductor correspondiente a una expresión regular será más o menos compleja en función de la expresión regular en cuestión. La construcción básica de este tipo de transductores consiste en crear un estado inicial y un estado final de la expresión y a continuación generar la estructura de estados y transiciones necesaria para aceptar las cadenas correspondientes a esa expresión y generar las transducciones de dichas cadenas. Un ejemplo

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

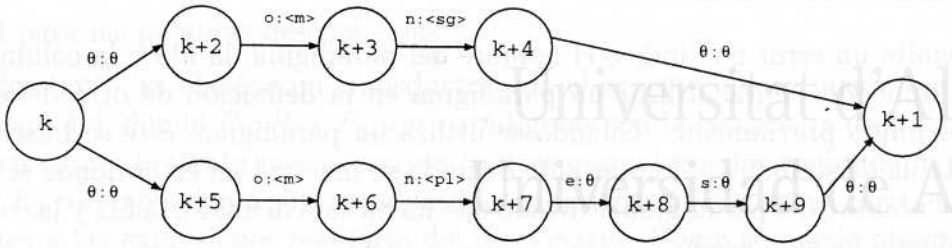


Figura 4.2: Subtransductor correspondiente al paradigma de género y número `[pl_on]>(ón:<m><sg>) |(ones:<m><pl>);` que se utiliza para nombre acabados en ón/ones.

sencillo que ilustra este tipo de construcción se puede ver en la figura 4.3, donde k y $k+1$ son respectivamente el estado inicial y final de la expresión, y la transición que llega al estado $k+5$ genera la etiqueta correspondiente al análisis morfológico de las cadenas correspondientes a la expresión.

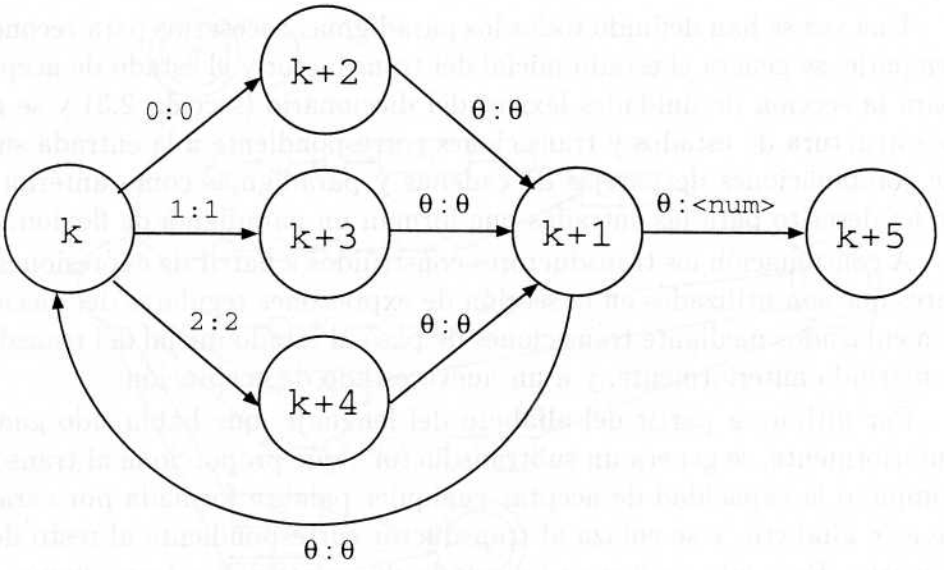


Figura 4.3: Transductor correspondiente al paradigma "`[nbase3]>[0-2]+ <num>;`" que acepta las cadenas correspondientes a la expresión regular "`[0-2]+`" y las genera analizadas morfológicamente como número (`<num>`).

El paradigma, una vez generada su estructura de estados y transiciones, se guarda en una tabla de símbolos en la que se consignan los siguientes datos: nombre, estado inicial y estado final de su estructura. Si en algún momento se intenta definir un paradigma con el mismo nombre, el compilador

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

emite un error que indica el nombre del paradigma, la fila y la columna del duplicado. Para utilizar un paradigma en la definición de otro debe estar definido previamente. Cuando se utiliza un paradigma, éste se busca en la tabla de símbolos. Si no ha sido definido se muestra un error donde se indica el nombre del paradigma utilizado que no ha sido definido, la fila y la columna donde ha sido usado.

Un paradigma se duplica en memoria cada vez que es utilizado. La duplicación en memoria de un paradigma consiste en realizar una copia exacta de toda la estructura de estados y transiciones correspondiente. Para ello, se comienza a duplicar desde el estado inicial, que se enlaza con transiciones vacías (θ:θ), hasta el estado final del paradigma que también se enlaza con transiciones vacías. Hay casos en los que la duplicación no es necesaria, por ejemplo cuando un paradigma se usa únicamente como sufijo de una o varias entradas; esta situación se detecta y la simplificación correspondiente reduce el uso de memoria y proporciona mayor rapidez al compilador al construir y minimizar el transductor.

Una vez se han definido todos los paradigmas necesarios para reconocer el lenguaje, se genera el estado inicial del transductor y el estado de aceptación para la sección de unidades léxicas del diccionario (sección 2.3) y se genera la estructura de estados y transiciones correspondiente a la entrada sucesiva de combinaciones de parejas de cadenas y paradigmas como anteriormente se ha descrito para las entradas que forman un paradigma de flexión.

A continuación los transductores construidos a partir de expresiones regulares que son utilizados en la sección de expresiones regulares del diccionario son enlazados mediante transiciones de paso al estado inicial del transductor, construido anteriormente, y a un nuevo estado de aceptación.

Por último, a partir del alfabeto del lenguaje, que había sido guardado anteriormente, se genera un subtransductor[2], que proporciona al transductor completo la capacidad de aceptar cualquier palabra formada por caracteres de este alfabeto, y se enlaza al transductor correspondiente al resto del diccionario. Para ello, se genera un estado de aceptación y transiciones, desde el estado inicial del transductor a éste y desde este estado de aceptación a sí mismo, que lean y escriban (sin cambiarlos) los caracteres definidos en el alfabeto del lenguaje. La transducción de las cadenas aceptadas por este estado únicamente se transcribe a la salida si el transductor no proporciona una transducción alternativa para esas cadenas, es decir, el transductor completo tiene dos niveles de validación diferentes para las cadenas de entrada: validación fuerte para las palabras contenidas en el diccionario y validación

[2]Recuérdese que esta opción sólo se utiliza en los diccionarios morfológicos de análisis (página 23).

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

débil para las palabras desconocidas[3].

Por tanto, se obtiene un transductor con el aspecto que se puede ver en la figura 4.4, donde P_1, P_2 y P_3 son paradigmas con la estructura de estados descrita anteriormente en esta sección (nótese que existen dos copias de P_1), y E_1 y E_2 corresponden a subtransductores que aceptan las cadenas correspondientes a las expresiones regulares del diccionario. Como se puede observar este transductor tiene estados de aceptación de diferentes tipos: W acepta las cadenas formadas por el alfabeto del lenguaje, S_1 acepta las cadenas que forman parte de la sección de unidades léxicas del diccionario y S_2 acepta las cadenas correspondientes a las expresiones regulares del diccionario. Como puede verse, la existencia de diferentes tipos de estados de aceptación supone una extensión de la definición 3.4, de forma que el conjunto de estados de aceptación F está formado por la unión de varios conjuntos de estados de aceptación F_i, donde i es el tipo de estado de aceptación[4].

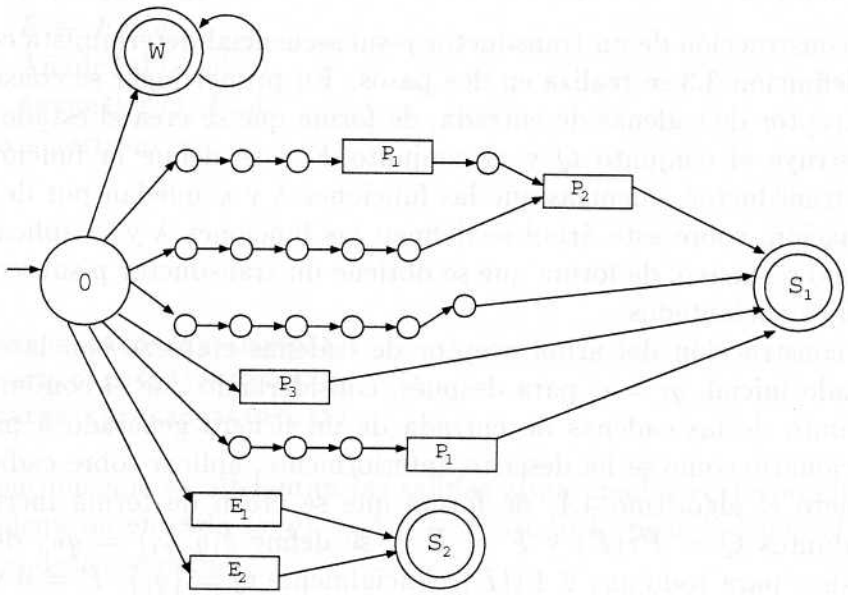


Figura 4.4: Transductor correspondiente a un posible diccionario.

[3]La utilidad de esta capacidad se verá en la sección 6.2.

[4]En este caso concreto cada F_i está formado por un único estado.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

4.3 Construcción de un transductor p-subsecuencial determinista a partir de un diccionario

Para construir un transductor p-subsecuencial determinista a partir de un diccionario como los descritos en la sección 2.3, es necesario transformar el diccionario en una enumeración de parejas de la forma `(entrada:salida)`. Para ello se despliegan los paradigmas de forma que se eliminan las regularidades existentes en los posibles alineamientos y proporciona la información entrada-salida por extensión. Por ejemplo, la entrada `(gat:gato<n>)[pl_fem]` del diccionario de la figura 2.1, se transforma en:

```
(gato:gato<n><m><sg>);
(gatos:gato<n><m><pl>);
(gata:gato<n><f><sg>);
(gatas:gato<n><f><pl>)
```

La construcción de un transductor p-subsecuencial determinista como los de la definición 3.3 se realiza en dos pasos. En primer lugar se construye el árbol aceptor de cadenas de entrada, de forma que se crea el estado inicial, se construye el conjunto Q y el conjunto F y se define la función δ del futuro transductor, mientras que las funciones λ y ψ quedan por definir. A continuación, sobre este árbol se definen las funciones λ y ψ aplicando un algoritmo recursivo, de forma que se obtiene un transductor p-subsecuencial de prefijos adelantados.

La construcción del árbol aceptor de cadenas empieza con la creación del estado inicial, $q_I = \epsilon$, para después, considerando que el conjunto E es el conjunto de las cadenas de entrada de un fichero generado a partir de un diccionario como se ha descrito anteriormente, aplicar sobre cada pareja del fichero el algoritmo 4.1, de forma que se crean de forma incremental los conjuntos $Q = Pr(E)$ y $F = E$, y se define $\delta(q, w_l) = qw_l$ de forma simultánea para todo $qw_l \in \mathrm{Pr}(E)$. Inicialmente $Q = \{q_I\}$, $F = \emptyset$ y δ está sin definir para ningún carácter.

Un ejemplo de esta construcción se puede ver en la figura 4.5, que corresponde a las siguientes entradas de un diccionario expandido:

```
(poco:poco<adv>);
(poco:poco<adj><m><sg>);
(pocos:poco<adj><m><pl>);
(poca:poco<adj><f><sg>);
(pocas:poco<adj><f><pl>);
(caro:caro<adj><m><sg>);
```


Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Algoritmo 4.1: Añade la cadena de entrada de una pareja al árbol aceptor de cadenas

Entrada: una pareja `entrada-salida` (w, w'), Q, F y δ
Salida: Q, F y δ modificadas

$q \leftarrow q_I$
$l \leftarrow 0$
`mientras` $(l < |w|)$
  `si` $\exists\delta(q, w_l)$
    $q \leftarrow \delta(q, w_l)$
  `sino`
    $\delta(q, w_l) \leftarrow$ CreaEstado()
    $q \leftarrow \delta(q, w_l)$
    $Q \leftarrow Q \cup q$
  `fin_si`
  $l \leftarrow l + 1$
`fin_mientras`
$F \leftarrow F \cup q$
AñadeTabla(w', q)
`devuelve` Q, F, δ
fin_algoritmo

```
(caros:caro<adj><m><pl>);
(cara:caro<adj><f><sg>);
(caras:caro<adj><f><pl>)
```

A continuación se adelantan las salidas almacenadas correspondientes a cada cadena de entrada ($\tau(q)$, con $q \in F$) todo lo posible hacia el estado inicial; es decir, se define

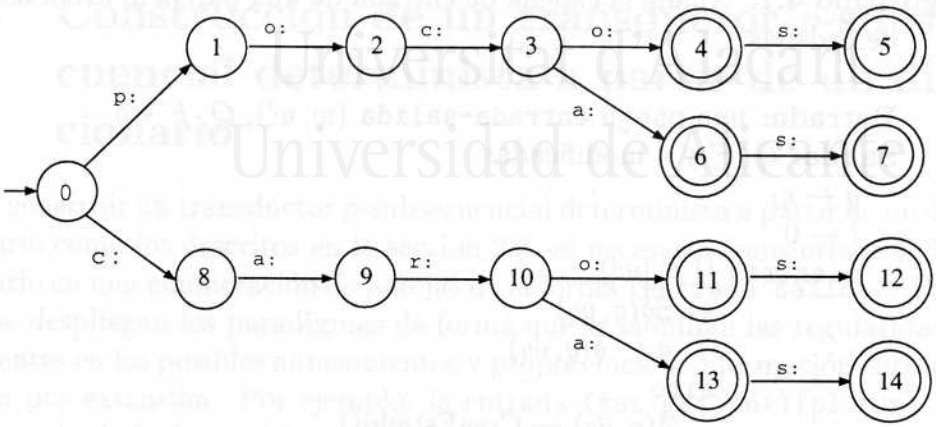
$$\widehat{p}(q) = \begin{cases} \text{PCML}[\bigcup_{\sigma\in\Sigma} \widehat{p}(\delta(q,\sigma)) \cup \tau(q)] & \text{si } q \in F \\ \text{PCML}[\bigcup_{\sigma\in\Sigma} \widehat{p}(\delta(q,\sigma))] & \text{si } q \notin F \end{cases} \quad (4.1)$$

De forma que

$$\lambda(q,\sigma) = \widehat{p}(q)^{-1}\widehat{p}(q\sigma)$$
$$\psi(q) = \widehat{p}(q)^{-1}\tau(q)$$

y se obtiene el transductor de prefijos adelantados correspondiente (definición 3.3). Para ello se aplica el algoritmo recursivo 4.2, que adelanta los

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002



Cadena ($w' = \tau(q)$)	Estado
poco<adv> poco<adj><m><sg>	4
poco<adj><m><pl>	5
poco<adj><f><sg>	6
poco<adj><f><pl>	7
caro<adj><m><sg>	11
caro<adj><m><pl>	12
caro<adj><f><sg>	13
caro<adj><f><pl>	14

Figura 4.5: Árbol aceptor de cadenas de entrada construido a partir de las entradas *poco*, *pocos*, *poca*, *pocas*, *caro*, *caros*, *cara*, *caras* y tabla de cadenas de salida asociadas a estados.

prefijos de salida, sobre el árbol aceptor de cadenas construido anteriormente, en el que se han inicializado las funciones de salida de la siguiente manera:

- $\lambda(q) = \emptyset$ si $q \notin F$;
- $\psi(q) = \tau(q)$ si $q \in F$;

de forma que el algoritmo las va modificando según va adelantando los prefijos hacia el estado inicial. Inicialmente se le pasa como parámetro el estado inicial del árbol aceptor de cadenas. El resultado de aplicar este algoritmo al árbol aceptor de cadenas de la figura 4.5 es el transductor de prefijos adelantados de la figura 4.6. Nótese que el transductor obtenido al aplicar esta operación no se corresponde con alineamientos de naturaleza lingüística,

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

sino con aquellos necesarios para el procesamiento determinista de la entrada y la obtención del prefijo más largo posible.

Universitat d'Alacant
Universidad de Alicante

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

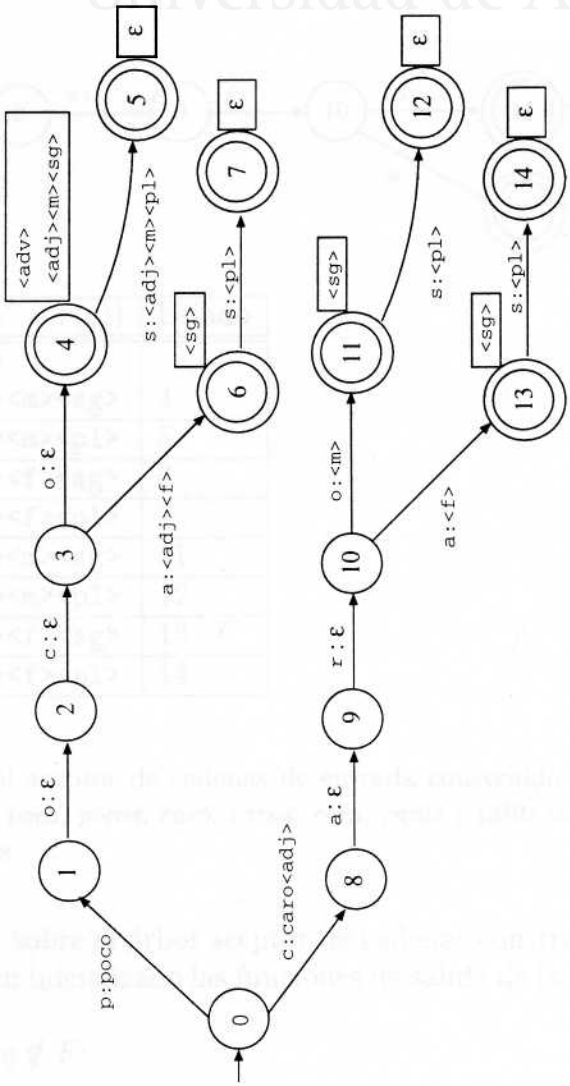


Figura 4.6: Transductor p-subsecuencial de prefijos adelantados correspondiente al árbol aceptor de cadenas y su tabla de la figura 4.5.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Volver al Indice/Tornar a l'index

Universitat d'Alacant
Universidad de Alicante

Algoritmo 4.2: Calcula el prefijo común más largo $\widehat{p}(q)$ de salida para un estado y actualiza las funciones λ y ψ del transductor

Entrada: q estado de un transductor p-subsecuencial acíclico
Salida: $\widehat{p}(q)$ prefijo de salida asociado al estado q

```
si (q ∈ F)
   p̂(q) ← PCML(⋃_{σ∈Σ} p̂(δ(q,σ)) ∪ τ(q)) [el PCML de un estado
                                            de aceptación]
sino
   p̂(q) ← PCML(⋃_{σ∈Σ} p̂(δ(q,σ))) [el PCML de un estado no final
                                    se calcula de forma recursiva]
   si (q = q_I) y (p̂(q) ≠ ϵ) [No se puede adelantar un prefijo antes
                              del estado inicial]
      para todo σ ∈ Σ
            λ(q,σ) ← p̂(δ(q,σ))
      fin_para
   sino
      para todo σ ∈ Σ
            λ(q,σ) ← [p̂(q)]^{-1} p̂(δ(q,σ))
      fin_para
   fin_si
fin_si
si q ∈ F [ q_I ∉ F ; no se acepta la cadena vacía]
   ψ(q) ← [p̂(q)]^{-1} τ(q) [actualiza la función de salida de los estados
                            de aceptación]
fin_si
devuelve p̂(q)
fin_algoritmo
```


Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Universitat d'Alacant
Universidad de Alicante

Capítulo 5

Algoritmos de minimización de transductores

5.1 Introducción

En este capítulo se presentan los algoritmos utilizados para minimizar los transductores construidos a partir de diccionarios que se han visto en el capítulo 4. Debido a que los transductores p-subsecuenciales únicamente se han utilizado para realizar un estudio comparativo entre los dos tipos de transductores se han utilizado dos algoritmos de minimización diferentes.

En la sección 5.2 se presenta el algoritmo utilizado para minimizar los transductores de letras, construidos como se explica en la sección 4.2, que consiste, a grandes rasgos, en realizar dos determinizaciones sobre el transductor de letras: la primera determinización afecta al transductor inverso; es decir, parte de los estados de aceptación hacia el estado inicial, y la segunda parte desde el estado inicial hacia los estados de aceptación, de forma que en la primera determinización se agrupan los sufijos comunes de las transducciones presentes en el transductor, mientras que en la segunda se agrupan los prefijos correspondientes.

En la sección 5.3 se presenta el algoritmo utilizado para minimizar los transductores p-subsecuenciales construidos como se ve en la sección 4.3. Este es un algoritmo clásico de minimización basado en clases de equivalencia que garantiza que el transductor resultante produce las mismas salidas para todos los prefijos y cadenas completas de entrada que el transductor original (es decir, con el mismo alineamiento entrada–salida).

Por último, en la sección 5.4 se realiza una comparación entre estos dos tipos de transductores mínimos para conocer la incidencia que tienen sus características cuando se aplican sobre datos reales.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

```
#etiquetas morfológicas
!simbol <pr>;

#diccionario
%dic
(por:por<pr>);
(con:con<pr>);
(para:para<pr>)
```

Figura 5.1: Ejemplo de diccionario morfológico

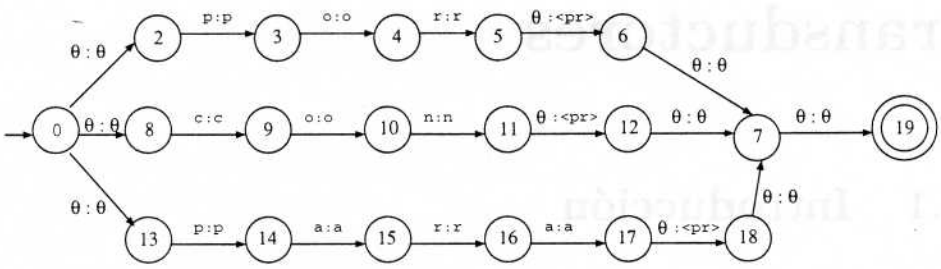


Figura 5.2: Transductor de letras original que corresponde al diccionario de la figura 5.1

5.2 Minimización de transductores de letras

El transductor de letras, que ha sido construido como se explica en la sección 4.2, consta de un camino por cada entrada del diccionario. Estos caminos están unidos mediante transiciones vacías por el principio a un estado inicial y por el final a uno o más estados de aceptación. Además, estos transductores pueden contener ciclos, debido a la posibilidad de construir transducciones mediante expresiones regulares. En la figura 5.2 se puede observar un ejemplo de este tipo de construcción que corresponde al diccionario (sin paradigmas ni expresiones regulares ni alfabeto del lenguaje, para simplificar) de la figura 5.1.

En esta sección se describe un algoritmo que tiene como entrada un transductor de letras indeterminista (TLI) de estas características, con N conjuntos (disjuntos) de estados de aceptación, $T = (Q, L, \delta, q_I, F_1, F_2, \ldots, F_N)$ y da como resultado un transductor de letras equivalente[1] $T' = (Q', L -$

[1]Puede ocurrir, en el caso general, que los conjuntos de estados de aceptación no sean disjuntos tras la minimización, porque en el transductor original una transducción determinada lleve desde el estado inicial a estados de aceptación pertenecientes a conjuntos

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

$\{(\theta,\theta)\}, \delta', q'_I, F'_1, F'_2, \ldots, F'_N)$, determinista, mínimo, y sin transiciones vacías.

El algoritmo 5.1 es una adaptación del descrito por van de Snepscheut (1993, p. 262–267) para autómatas finitos posiblemente indeterministas. Se ha usado este algoritmo en lugar de utilizar el algoritmo clásico de Hopcroft y Ullman (1979, p. 19–25, 68) para determinizar y minimizar el transductor, ya que se ha mostrado muy eficiente con transductores acíclicos o débilmente cíclicos[2], como son los utilizados en esta tesis. Este algoritmo consiste en realizar dos determinizaciones, de forma que minimiza el transductor en dos etapas:

- Primero se determiniza el transductor de letras inverso (resultante de cambiar el sentido de todas las flechas de las transiciones[3]), obteniendo un TL intermedio. En este paso se agrupan las regularidades presentes en los sufijos de las transducciones.

- A continuación se determiniza el resultado de invertir de nuevo las transiciones del TL intermedio. En este paso se agrupan las regularidades presentes en los prefijos de las transducciones.

Para ello el algoritmo 5.1 utiliza las siguientes funciones:

- $D_{\leftarrow}(A,l) = \{q : \delta(q,l) \cap A \neq \emptyset\}$ con $l \in L$ y $A \subseteq Q$ es el conjunto de estados del transductor original desde los que se alcanzan, mediante transiciones etiquetadas con l, estados de un subconjunto A.

- $C_{\leftarrow\theta}(A)$ es la clausura-θ inversa de un conjunto $A \subset Q$, es decir, el conjunto de todos los estados de Q accesibles desde A en cero o más transiciones inversas (θ,θ), o sea, el conjunto de todos los estados de Q desde los que se llega con cero o más transiciones vacías a algún estado de A, que se describe en el algoritmo 5.2.

- La función de transición del transductor intermedio que construye el algoritmo, resultante de la primera inversión y determinización es

$$\delta_{\leftarrow}(q_{\leftarrow}, l) = C_{\leftarrow\theta}(D_{\leftarrow}(q_{\leftarrow}, l)) \tag{5.1}$$

  donde $q_{\leftarrow}$ es un conjunto de estados de Q que constituye un estado del transductor intermedio y $l \in L - \{(\theta,\theta)\}$. Nótese que el resultado de

distintos; sin embargo, los TLI construidos a partir de diccionarios de esta tesis no tienen nunca esta propiedad.

[2]Informalmente, un transductor es débilmente cíclico cuando presenta pocos ciclos y de ámbito reducido, es decir, el ciclo abarca pocas transiciones.

[3]En realidad, cuando se construye el transductor de letras a partir del diccionario, se almacenan también las transiciones inversas, lo que efectivamente evita la operación de inversión.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Algoritmo 5.1: Determiniza y minimiza un transductor de letras mediante el procedimiento de doble inversión y determinización

Entrada: TLI $T = (Q, L, \delta, q_I, F_1, \ldots, F_N)$ con N tipos de estados de aceptación

Salida: TLD $T' = (Q', L - \{(\theta, \theta)\}, \delta', q'_I, F'_1, \ldots, F'_N)$ mínimo equivalente

$Q_{\leftarrow} \leftarrow \{C_{\leftarrow\theta}(F_1), \ldots, C_{\leftarrow\theta}(F_N)\}$; , [inicialización: el TL intermedio que
$t \leftarrow 0$, $R^{[0]} \leftarrow Q_{\leftarrow}$ tiene N estados iniciales]
repite [crea los estados $Q_{\leftarrow}$ del TL intermedio; primera etapa]
  $t \leftarrow t + 1$; $R^{[t]} \leftarrow \emptyset$
  para todo $l \in L - \{(\theta, \theta)\}$
    para todo $q_{\leftarrow} \in R^{[t-1]}$
      $R^{[t]} \leftarrow R^{[t]} \cup \{\delta_{\leftarrow}(q_{\leftarrow}, l)\}$ [ec. (5.1)]
    fin_para
  fin_para
  $D \leftarrow R^{[t]} - Q_{\leftarrow}$; $Q_{\leftarrow} \leftarrow Q_{\leftarrow} \cup R^{[t]}$;
hasta que $D = \emptyset$ [fin primera etapa]
$q'_I \leftarrow \emptyset$ [construcción del estado inicial del nuevo TLD; segunda etapa]
para todo $q_{\leftarrow} \in Q_{\leftarrow}$
  si $q_I \in q_{\leftarrow}$ entonces $q'_I \leftarrow q'_I \cup \{q_{\leftarrow}\}$
fin_para
$t \leftarrow 0$; $Q' \leftarrow \{q'_I\}$; $R^{[0]} \leftarrow Q'$ [inicialización]
repite [construye conjunto de estados Q' y función de transición δ' del TLD
  $t \leftarrow t + 1$; $R^{[t]} \leftarrow \emptyset$
  para todo $l \in L - \{(\theta, \theta)\}$
    para todo $q' \in R^{[t-1]}$
      $\delta'(q', l) \leftarrow \emptyset$
      para todo $q_{\leftarrow} \in Q_{\leftarrow}$
        si $\delta_{\leftarrow}(q_{\leftarrow}, l) \in q'$ entonces $\delta'(q', l) \leftarrow \delta'(q', l) \cup \{q_{\leftarrow}\}$
      fin_para
      $R^{[t]} \leftarrow R^{[t]} \cup \{\delta'(q', l)\}$
    fin_para
  fin_para
  $D \leftarrow R^{[t]} - Q'$; $Q' \leftarrow Q' \cup R^{[t]}$
hasta que $D = \emptyset$
para todo $i \in [1, N]$ [asigna estados de aceptación]
  $F'_i \leftarrow \emptyset$
  para todo $q' \in Q'$
    si $\exists q_{\leftarrow} \in q' : q' \cap F_i \neq \emptyset$ entonces $F'_i \leftarrow F'_i \cup \{q'\}$
  fin_para
fin_para

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Algoritmo 5.2: Realizar la clausura–θ inversa de un conjunto de estados

Entrada: Un conjunto de estados $A \subseteq Q$
Salida: Un conjunto de estados $C_{\leftarrow\theta}(A) \subseteq Q$
Método:

1. Inicializa $t \leftarrow 0$, $R_\theta^{[0]} \leftarrow A$, $C_{\leftarrow\theta}(A) \leftarrow \emptyset$
2. Haz
   $C_{\leftarrow\theta}(A) \leftarrow C_{\leftarrow\theta}(A) \cup R_\theta^{[t]}$
   $R_\theta^{[t+1]} \leftarrow D_{\leftarrow}(R_\theta^{[t]}, (\theta, \theta))$
   $t \leftarrow t + 1$
   mientras $R_\theta^{[t]} \not\subseteq C_{\leftarrow\theta}(A)$
3. Devuelve $C_{\leftarrow\theta}(A)$.

aplicar la función $\delta_{\leftarrow}$ puede ser el conjunto vacío: cuando esto ocurre, este estado representa el estado de absorción del transductor inverso intermedio.

Si aplicamos el algoritmo 5.1 al TL de la figura 5.2, en la primera etapa se genera el TL intermedio que se puede observar la figura 5.3.

En la segunda etapa de este algoritmo se obtiene el TLD mínimo equivalente al TLI de la figura 5.2, que se puede observar en la figura 5.4.

5.3 Minimización de transductores p-subsecuenciales

Esta sección describe la minimización de transductores p-subsecuenciales acíclicos construidos como se explica en la sección 4.3. Un ejemplo de esta construcción se puede ver en la figura 5.6, que corresponde a las entradas de la figura 5.5.

Para ello se aplica un algoritmo de minimización clásico basado en el establecimiento de clases de equivalencia (Aho et al. 1986, p. 142–143). En esta versión del algoritmo, en lugar de comprobar si dos estados son equivalentes, se comprueba si dos estados no son equivalentes, de forma que en cada iteración se refina la partición inicial (estados de aceptación y estados de no aceptación), hasta que no se puede refinar más.

Al aplicar el algoritmo 5.3 sobre el transductor de la figura 5.6 se obtiene el transductor mínimo que se puede observar en la figura 5.7.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

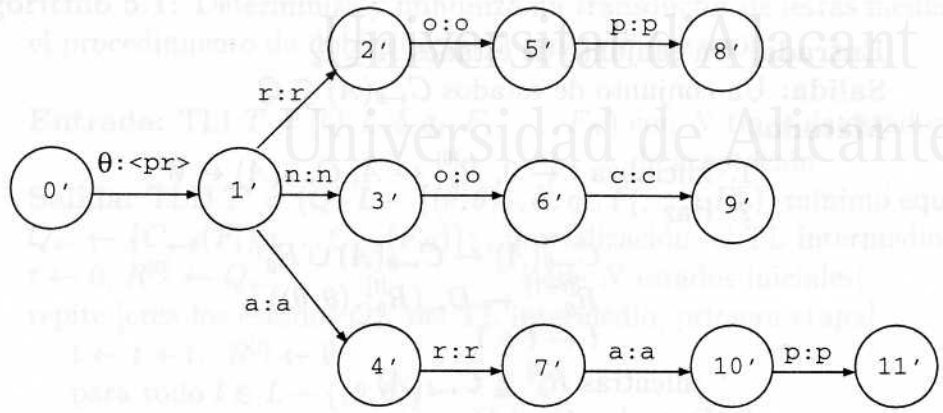


Figura 5.3: TL intermedio correspondiente al TLI de la figura 5.2 donde el estado $0'$ contiene el estado de aceptación del TLI original y los estados $8'$, $9'$ y $11'$ contienen el estado inicial.

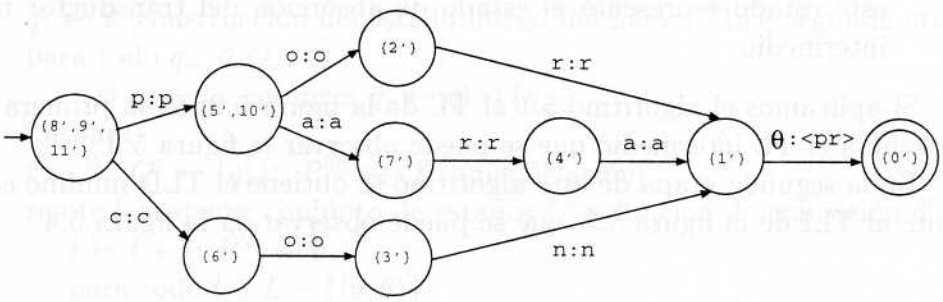


Figura 5.4: TLD mínimo construido a partir del TL intermedio de la figura 5.3

```
(adorno:adorno<n><m><sg>);
(adornos:adorno<n><m><pl>);
(adorno:adornar<vblex><pri><1><sg>);
(baremo:baremo<n><m><sg>);
(baremos:baremo<n><m><pl>);
(baremo:baremar<vblex><pri><1><sg>);
(grito:grito<n><m><sg>);
(gritos:grito<n><m><pl>);
(grito:gritar<vblex><pri><1><sg>)
```

Figura 5.5: Entradas de un diccionario dadas por extensión

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

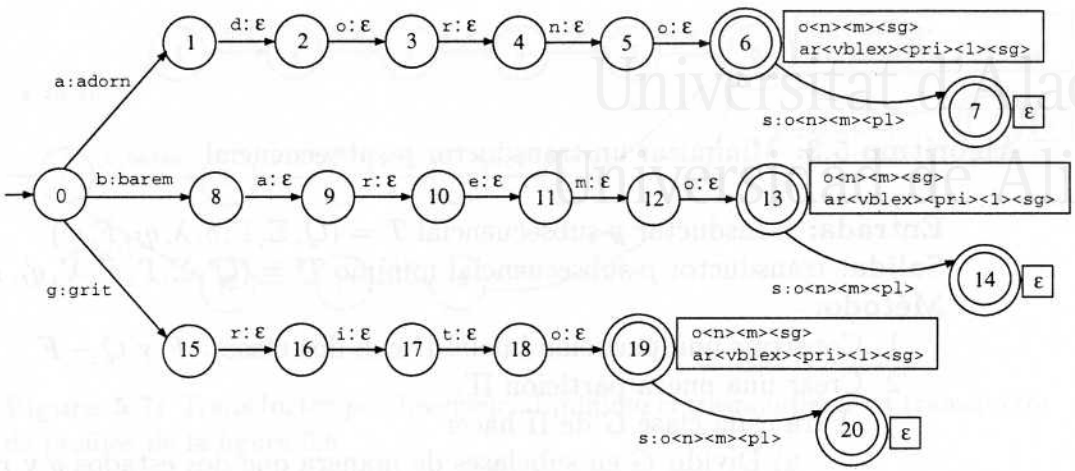


Figura 5.6: Transductor de prefijos p-subsecuencial correspondiente a las entradas de la figura 5.5.

Nótese que para añadir o eliminar una entrada en este tipo de transductores es necesario (en el peor caso) realizar por completo las operaciones de retrasar completamente los prefijos, añadir la nueva transducción, volver a adelantar los prefijos y minimizar el transductor para obtener un transductor que contemple esta modificación en el lenguaje aceptado (es decir, reconstruirlo completamente).

5.4 Comparación entre transductores de letras indeterministas y deterministas

El objetivo de esta sección es comparar dos tipos de transductores construidos a partir de diccionarios:

- Transductores que denominaremos TLD-L: transductores de letras deterministas con respecto al conjunto L de las etiquetas de sus transiciones construidos directamente a partir de diccionarios alineados tal como se indica en el apartado 4.2 y minimizados como se describe en el apartado 5.2, y

- Los transductores p-subsecuenciales construidos tal como se indica en el apartado 3.3 (es decir, sin preservar los alineamientos del diccionario) y minimizados como se describe en el apartado 5.3, convertidos convenientemente como se verá más abajo en transductores de letras

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Algoritmo 5.3: Minimizar un transductor p-subsecuencial

Entrada: transductor p-subsecuencial $T = (Q, \Sigma, \Gamma, \delta, \lambda, q_I, F, \psi)$
Salida: transductor p-subsecuencial mínimo $T' = (Q', \Sigma, \Gamma, \delta', \lambda', q'_I, F', \psi')$
Método:

1. Construir una partición inicial Π con dos clases: F y $Q - F$
2. Crear una nueva partición Π':
   Para cada clase G de Π hacer
   a) Dividir G en subclases de manera que dos estados q y r *no estarán en la misma subclase si cumplen alguna de las* siguientes condiciones:
      1. $\psi(q) \neq \psi(r)$, si $q \in F$, $r \in F$
      2. $\delta(q, \sigma)$ pertenece a una clase distinta en Π de la clase a la que pertenece $\delta(r, \sigma)$, para algún $\sigma \in \Sigma$
      3. $\lambda(q, \sigma) \neq \lambda(r, \sigma)$, para algún $\sigma \in \Sigma$

   b) Sustituir G en Π' por las subclases creadas
3. Si $\Pi' \neq \Pi$
   $\Pi \leftarrow \Pi'$
   Volver al paso 2.
4. Elegir uno de los estados en cada clase de Π' como representante de la clase; estos estados formarán Q'. Si el estado destino de alguna transición de algún $q \in Q'$ no es el representante de una clase, se modifica dicha transición de forma que apunte al estado representante de dicha clase; de esta manera $\delta'(q, \sigma) \in Q' \ \ \forall q \in Q', \forall \sigma \in \Sigma$.
   Para todo $q \in Q'$ se definen las funciones ψ' y λ' de la siguiente manera: $\psi'(q) = \psi(q)$ y $\lambda'(q, \sigma) = \lambda(q, \sigma)$. El estado inicial q'_I es el representante de la clase que contiene el estado q_I. El conjunto F' es el conjunto formado por los representantes que estuviesen originalmente en F.
5. Devolver $T' = (Q', \Sigma, \Gamma, \delta', \lambda', q'_I, F', \psi')$

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

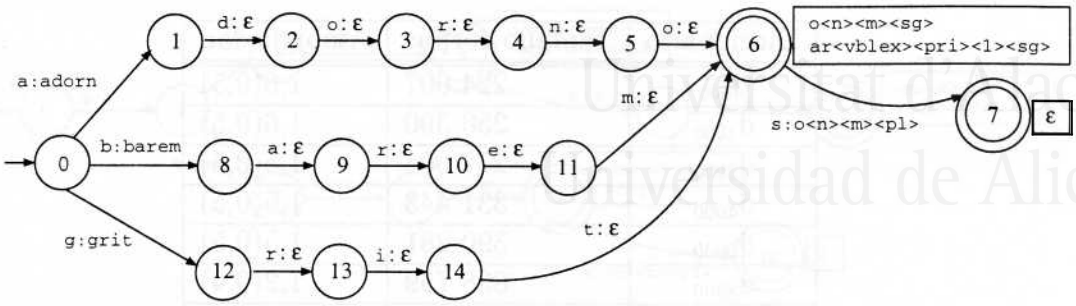


Figura 5.7: Transductor p-subsecuencial mínimo correspondiente al transductor de prefijos de la figura 5.6.

deterministas respecto del conjunto L que denominaremos TLCD-Σ o transductores de letras cuasideterministas respecto del alfabeto Σ. Son *cuasideterministas* respecto de Σ porque, como se verá, contienen pocos estados con transiciones salientes con el mismo símbolo de Σ, y estos pocos estados se encuentran sólo hacia el final del procesamiento para producir los sufijos de los conjuntos ψ de los transductores p-subsecuenciales.

Para esta comparación se han utilizado datos reales de palabras y sus frecuencias en corpus de texto, en una tarea de análisis morfológico. En concreto, para el estudio realizado en esta sección se ha usado un corpus de texto de alrededor de dos millones de palabras del castellano, en su mayor parte obtenidas por recopilación de artículos periodísticos. Para cada diccionario, se han eliminado del corpus las palabras que no eran reconocidas, con el fin de estudiar las palabras conocidas con sus frecuencias reales en los textos. El nombre de los diccionarios indica el tamaño aproximado del vocabulario que contienen, y se han construido de forma incremental, es decir, cada diccionario contiene el mismo vocabulario que el diccionario anterior, pero aumentado.

En la tabla 5.1 se pueden observar los resultados que se han obtenido al realizar un estudio sobre la ambigüedad léxica del lenguaje natural, definida como el número medio de formas léxicas por forma superficial que se da al analizar un corpus de texto. Para ello se cuentan todas las formas superficiales que hay en el texto (N_{FS}) y todas las formas léxicas que se generan al analizarlo (N_{FL}), de forma que la media de formas léxicas por forma superficial es $N_{\mathrm{FL}}/N_{\mathrm{FS}}$; la desviación típica correspondiente se indica en la tabla entre paréntesis. Esta ambigüedad se manifiesta en los transductores p-subsecuenciales como el cardinal de la función $\psi(q)$ de los estados de acep-

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Diccionario	Tamaño corpus	Ambigüedad
d_{500}	224 007	1,6(0,5)
d_{1000}	250 390	1,6(0,5)
d_{2000}	277 004	1,5(0,5)
d_{4000}	331 443	1,5(0,5)
d_{6000}	390 281	1,5(0,5)
d_{8000}	838 759	1,2(0,4)
d_{12000}	1 040 156	1,2(0,4)
d_{15000}	1 241 661	1,2(0,4)
d_{16500}	1 260 273	1,2(0,5)

Tabla 5.1: Corpus usados y el número medio (y desviación típica) de análisis por palabra (FL por FS) con diccionarios de distintos tamaños.

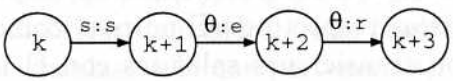


Figura 5.8: Secuencia de transiciones y estados generados a partir de la transición (s:ser).

tación del transductor, es decir, como el número de colas que concatena a su salida.

A continuación, sobre estos corpus, se ha realizado una comparación entre los TLD-L y los TLCD-Σ, construidos con los diccionarios que se han utilizado para filtrar cada corpus. Los transductores de letras cuasideterministas respecto del alfabeto de entrada (TLCD-Σ) que se han utilizado en este experimento se han obtenido a partir de los transductores p-subsecuenciales de la siguiente forma:

- Para cada transición del transductor p-subsecuencial que tenga asociada una cadena de salida de más de un símbolo, es decir, $\lambda(q, \sigma) = \gamma_1 \dots \gamma_n$ con $n > 1$, se generan $n - 1$ estados intermedios en el transductor de letras. Las etiquetas de transiciones de este nuevo camino son, en la primera, (σ, γ_1), y el resto $(\theta, \gamma_2) \dots (\theta, \gamma_n)$. Por ejemplo, a partir de la transducción *(s:ser)* se genera la secuencia de transiciones y estados que se puede ver en la figura 5.8.

- Para cada estado de aceptación $q \in F$ y para cada cola $\gamma_1\gamma_2 \dots \gamma_n \in \psi(q)$, se construye un camino de estados que va desde el estado q (que se marca como de no aceptación) hasta el único estado de aceptación del

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

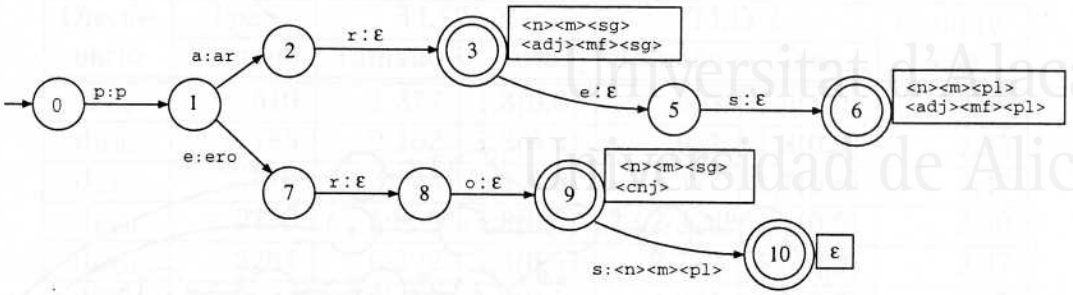


Figura 5.9: Ejemplo de transductor p-subsecuencial mínimo.

nuevo transductor, q_F, con las transiciones etiquetadas $(\theta, \gamma_1), (\theta, \gamma_2), \ldots (\theta, \gamma_n)$. Esta es la única fuente de no determinismo respecto de Σ del transductor de letras resultante, de ahí el nombre de cuasideterminista.

- Este transductor de cuasideterminista respecto de Σ o TLCD-Σ es minimizado respecto de L utilizando el algoritmo visto en la sección 5.2, de forma que se obtiene un nuevo TLCD-Σ, con los mismos alineamientos de entrada–salida que el p-subsecuencial, y mínimo. Esta operación de minimización es necesaria debido a que se generan transiciones intermedias como resultado de dividir las cadenas de salida del transductor p-subsecuencial en símbolos individuales, ya que en algunos casos se generan estados equivalentes.

Por ejemplo, el transductor de letras determinista que se observa en la figura 5.10 se ha obtenido a partir del transductor p-subsecuencial de la figura 5.9, y que se puede observar que no es mínimo. Al aplicarle el algoritmo de minimización antes mencionado se obtiene el TLCD-Σ mínimo que se puede ver en la figura 5.11.

La transformación de un transductor p-subsecuencial en un transductor de letras se ha hecho con el fin de poder realizar una comparación con los transductores de letras en los que se basa esta tesis, es decir, basada en la misma estructura de datos, ya que comparar transductores con estructuras tan dispares como son los transductores p-subsecuenciales y los transductores de letras no sería adecuado. De hecho, se podría decir que lo que realmente se compara es el transductor de letras resultante de preservar el alineamiento original del diccionario con el transductor de letras resultante del nuevo alineamiento resultante de la operación de avanzar al máximo los prefijos de las transducciones.

En la tabla 5.2 se pueden ver los resultados obtenidos al convertir cada diccionario en los dos tipos de transductores de letras. Con respecto al

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

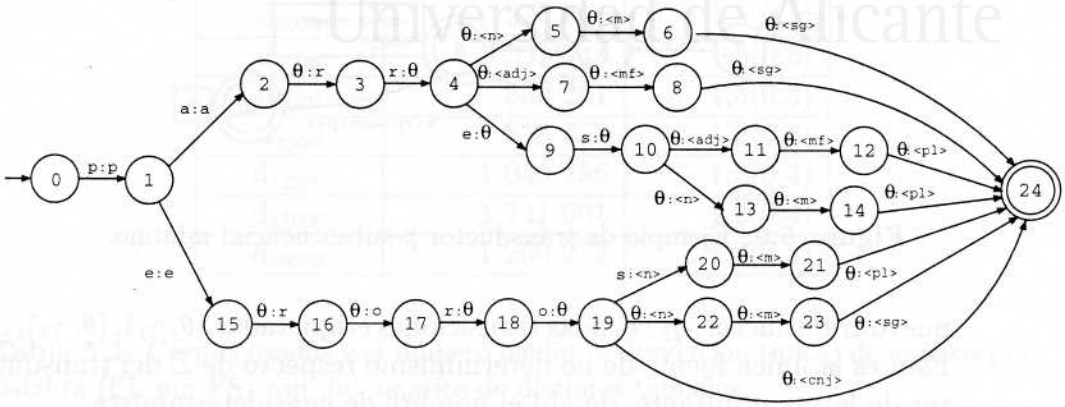


Figura 5.10: Transductor de letras cuasideterminista respecto del alfabeto de entrada (TLCD-Σ) correspondiente al transductor *p*-subsecuencial de la figura 5.9.

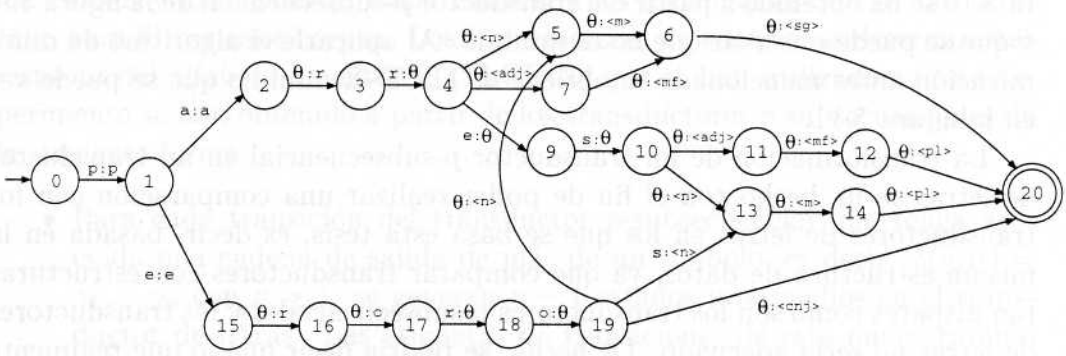


Figura 5.11: TLCD-Σ mínimo correspondiente al TLCD-Σ de la figura 5.10.

Diccio-	TpSS	TLCD-Σ		TLD-L		Compre-
nario	Tamaño	Tamaño	Tasa	Tamaño	Tasa	sión
d_{500}	510	1 377	1,3(0,6)	608	1,9(1,0)	2,26
d_{1000}	785	2 162	1,3(0,6)	921	1,8(0,9)	2,35
d_{2000}	1316	3 643	1,3(0,6)	1 477	1,7(0,9)	2,47
d_{4000}	2130	5 830	1,3(0,5)	2 324	1,6(0,9)	2,50
d_{6000}	2284	6 592	1,3(0,5)	2 779	1,6(0,9)	2,37
d_{8000}	3814	10 458	1,2(0,5)	4 071	1,5(0,8)	2,57
d_{12000}	4935	13 973	1,3(0,5)	5 487	1,6(0,9)	2,55
d_{15000}	5883	16 450	1,3(0,5)	6 467	1,5(0,9)	2,54
d_{16500}	5916	16 566	1,3(0,5)	6 537	1,5(0,9)	3,53

Tabla 5.2: Comparación de tamaños y tasa de indeterminismo (número medio de pares estado–salida vivos (algoritmo 5.2) que mantiene el transductor al analizar una palabra) entre TLD-L y TLCD-Σ. El número de estados del transductor p-subsecuencial original (TpSS) se da como referencia.

tamaño (número de estados) cabe destacar que la reducción de tamaño que muestran los TLD-L respecto de los TLCD-Σ es aproximadamente 2,5, y que el número de estados del TLD-L no es mucho mayor que el del transductor p-subsecuencial mínimo original (la ratio es aproximadamente 1,1). La diferencia de tamaños entre TLD-L y TLCD-Σ se puede observar más claramente en la figura 5.12. Otro dato relevante que se puede observar en esta tabla es el grado de indeterminismo (número medio de pares estado–salida vivos durante el análisis (algoritmo 3.1)) que muestran estos transductores al analizar el texto. La media, en este caso, muestra el número medio de pares estado–salida vivos que mantiene el transductor por durante el procesamiento de la entrada, indicándose entre paréntesis la desviación típica correspondiente. De esta comparación se deduce que el grado medio de indeterminismo que presentan los TLD-L no es mucho mayor que el que presentan los TLCD-Σ, éste último debido únicamente a los estados correspondientes a las colas del transductor p-subsecuencial correspondiente, ya que el resto del transductor es determinista respecto a la entrada. Recuérdese que el escaso indeterminismo de los transductores p-subsecuenciales se reduce a las colas, necesarias para tratar fenómenos como la homografía; de hecho, la tasa de indeterminismo que presentan los TLCD-Σ es del orden de la tasa de ambigüedad que se puede observar en la tabla 5.1.

Como ya se ha dicho, con esta comparación en realidad lo que se compara el resultado que se obtiene al utilizar un alineamiento lingüísticamente

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

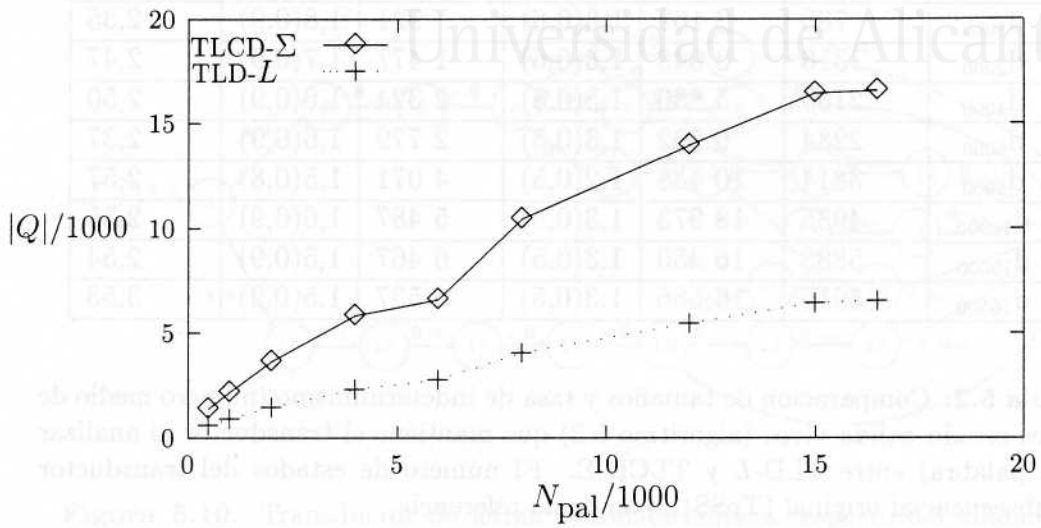


Figura 5.12: Comparación de tamaños entre TLD-L y TLCD-Σ.

motivado (que puede ser indeterminista) con un alineamiento automático determinista (de prefijos) para construir un TL. A tenor de estos datos, se deduce que el sacrificio temporal que supone utilizar TLD-L en lugar de TLCD-Σ es pequeño, a cambio de una gran mejora de la eficiencia espacial.

En la tabla 5.3 se indican los datos referentes a la longitud media de las cadenas emitidas por carácter leído en las transiciones de un transductor p-subsecuencial al procesar un texto, indicando la correspondiente desviación típica entre paréntesis, así como el tamaño de estos transductores. Para calcular la media se han sumado las longitudes de las cadenas emitidas por cada carácter procesado y se ha divido por el número total de caracteres procesados a través del transductor. Como se puede observar la media de la longitud de las cadenas emitidas es en general menor que 1, es decir, que se ha tenido que retrasar gran parte de la salida hasta los estados de aceptación, de forma que se confirman los fenómenos ilustrados en las figuras 1.5, 3.1 y 3.2, en las que se podía apreciar el retraso que se produce en las transducciones de salida de los transductores p-subsecuenciales.

En la misma tabla se muestra la longitud media de los sufijos finales ($\psi(q)$) en un transductor p-subsecuencial, es decir, la longitud media de las cadenas emitidas en los estados de aceptación, teniendo en cuenta que en un

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Volver al Indice/Tornar a l'index

Diccionario	Longitud media de las cadenas	
	Transiciones	Sufijos
d_{500}	0,4(1,0)	2,0(1,3)
d_{1000}	0,4(1,1)	1,9(1,2)
d_{2000}	0,5(1,2)	1,9(1,2)
d_{4000}	0,6(1,2)	1,8(1,2)
d_{6000}	0,7(1,3)	2,5(3,4)
d_{8000}	0,7(1,2)	1,8(1,2)
d_{12000}	0,7(1,2)	2,1(2,3)
d_{15000}	0,7(1,2)	2,1(2,1)
d_{16500}	0,7(1,2)	2,2(2,4)

Tabla 5.3: Longitud media (y desviación típica) de las cadenas emitidas en las transiciones $\lambda(q, \sigma)$ y de los sufijos emitidos en los estados de aceptación ($\psi(q)$) por un transductor p-subsecuencial.

estado de aceptación pueden emitirse varias cadenas de longitud diferente. Los resultados son compatibles con los de la longitud media de las cadenas emitidas en las transiciones, ya que la longitud media de las cadenas emitidas en los estados de aceptación es en general mayor que 1.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

Capítulo 6

Construcción y mantenimiento incremental de transductores de letras mínimos

6.1 Introducción

Durante el mantenimiento de un sistema de traducción automática hay dos operaciones muy comunes: la adición de una entrada a un diccionario (transductor), correspondiente a una nueva palabra en el vocabulario, y la eliminación de una entrada (por ejemplo, cuando se detecta un error en el analizador morfológico). Inicialmente, en el sistema de TA interNOSTRUM estas operaciones se realizaban añadiendo y eliminando las entradas de los diccionarios y recompilándolos en un transductor de letras mínimo. Este procedimiento era demasiado lento para ser conveniente. Más concretamente, no se compilaba el diccionario completo; en su lugar, los diccionarios eran divididos en secciones, compilados por separado, y los correspondientes TL mínimos eran unidos en un sólo transductor, de forma que la adición o eliminación de una entrada afectaba únicamente a la recompilación de una sección concreta del diccionario y a la unión de todos los TL. Este procedimiento era mejor que el anterior, pero demasiado lento todavía. En algunas situaciones, el tiempo real de adición y eliminación de transducciones puede ser crucial; por ejemplo, cuando se depuran los módulos léxicos de un sistema de traducción automática o cuando un usuario quiere añadir palabras nuevas para su utilización inmediata en una aplicación concreta.

En este capítulo[1] se muestra un método eficiente y simple para modificar un TEF mínimo de modo que se añada o se elimine una transducción

[1]Este capítulo es una traducción y ampliación del artículo Garrido-Alenda et al. 2002.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

del lenguaje aceptado por el TEF. Los algoritmos son aplicables a un tipo particular de TEF llamados transductores de letras deterministas aumentados (TLDA). Estos transductores son un tipo concreto de transductores de letras (definición 3.4) y son una forma conveniente de implementar analizadores morfológicos ya que:

- Analizan el texto de forma simultánea a su segmentación, utilizando un carácter de preanálisis, en unidades motivadas morfológicamente (que pueden estar compuestas por más de una palabra);
- Se pueden manipular fácilmente como autómatas finitos deterministas (AFD) sobre un alfabeto de pares de símbolos entrada–salida.
- Las operaciones de adición y eliminación de entradas no modifican los alineamientos de entrada/salida existentes (al contrario de lo que ocurre con los transductores p-subsecuenciales).

6.2 Definiciones matemáticas

6.2.1 Transductores de letras deterministas aumentados

Los transductores de letras deterministas aumentados (TLDA) utilizados se definen como un transductor de letras (definición 3.4), teniendo en cuenta que el conjunto de estados de aceptación, F, se define mediante dos funciones que asignan conjuntos de validación a cada estado. Es decir:

Definición 6.1 (Transductor de letras determinista aumentado) *Un transductor de letras determinista aumentado se define como*

$$T = (Q, L, \delta, q_I, \xi_s, \xi_w) \tag{6.1}$$

donde

- Q *es el conjunto finito de estados*
- L *el conjunto de etiquetas de las transiciones del tipo que se define en la ecuación (6.2)*
- $q_I \in Q$ *el estado inicial*
- $\delta : Q \times L \rightarrow Q$ *la función de transición*

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

- $\xi_s : Q \rightarrow 2^{\Sigma \cup \{\$\}}$ *es la función que asigna un* conjunto de validación fuerte *(o* preanálisis fuerte*)* $\xi_s(q)$ *a cada estado, con el símbolo* \$ *como marca de fin de fichero,*

- $\xi_w : Q \rightarrow 2^{\Sigma \cup \{\$\}}$ *es la función que asigna un* conjunto de validación débil *(o* preanálisis débil*)* $\xi_w(q)$ *a cada estado, con el símbolo* \$ *como marca de fin de fichero y*

- $\xi_s(q) \cap \xi_w(q) = \emptyset$ *para todo* $q \in Q$ *(el uso de los conjuntos de validación se aclarará en la sección 7.2).*

Se define el conjunto de etiquetas de las transiciones del transductor como el conjunto

$$L = ((\Sigma \cup \{\theta\}) \times \Gamma) \cup (\Sigma \times (\Gamma \cup \{\theta\})) \tag{6.2}$$

donde

- Σ *es el alfabeto de los símbolos de entrada,*

- Γ *el alfabeto de símbolos de salida, y*

- θ *representa el símbolo vacío.*

Los estados tales que $\xi_s(q) \cup \xi_w(q) \neq \emptyset$ se llamarán estados de *aceptación*; cada conjunto de estados de aceptación con los mismos valores de $\xi_s(q)$ y $\xi_w(q)$ se corresponden con los diferentes tipos de estados de aceptación de F posteriores a la minimización (sección 5.2); es decir, los TLDA son una especificación de los transductores de letras deterministas respecto de L.

De acuerdo con la definición de L en los TLDA, las etiquetas de las transiciones entre estados pueden ser de tres tipos: (σ, γ), significa que se lee un símbolo $\sigma \in \Sigma$ y se escribe un símbolo $\gamma \in \Gamma$; (σ, θ) significa que lee un símbolo pero no se escribe nada y (θ, γ) significa que no lee nada pero se escribe un símbolo. Las transiciones del tipo (θ, θ), presentes en la definición 3.4, no existen en este tipo de transductores, ya que pueden evitar el determinismo en el transductor.

En el análisis morfológico, los símbolos de Σ son aquellos que se encuentren en el texto, y los símbolos de Γ son aquellos necesarios para formar los lemas y la representación de su información morfológica. El preanálisis únicamente es necesario para el análisis morfológico; la transferencia léxica y la generación morfológica, que trabajan con formas léxicas previamente delimitadas, no necesitan las condiciones de preanálisis, de forma que los TLDA utilizados para estos procesos se puede considerar que tienen $\xi_s(q) = \Sigma \cup \{\$\}$, es decir, que son transductores de letras que tienen los clásicos estados de aceptación incondicional y de no aceptación (Roche y Schabes 1997).

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Aunque los TLDA son *deterministas* respecto de L, en general son indeterministas respecto de Σ[2]. La entrada indeterminista es sustancial para la codificación de muchas regularidades tal como las expresan los lingüistas y permiten representar la ambigüedad del lenguaje de forma directa, mientras que los TEF como los transductores p-subsecuenciales vistos en la sección 3.3, que se determinizan mediante un realineamiento de las transducciones y, por tanto, destruyen los alineamientos motivados lingüísticamente, fuerzan un modelo de ambigüedad que produce diferentes sufijos de salida después de alcanzar un estado de aceptación, como se puede ver en la figura 4.6. Por otra parte, los experimentos realizados con diccionarios y corpus reales, muestran que la entrada indeterminista (es decir, el número medio de transducciones vivas durante el proceso) es muy baja y está entorno a 2 (sección 5.4).

Aquí se define δ para que sea una función *completa*; el TLDA correspondiente se llamará *completo*, lo que no implica pérdida de generalidad ya que cualquier TLDA puede hacerse completo añadiendo un nuevo estado de *absorción* $\perp$ a Q, de forma que todas las transiciones no definidas apunten a éste, $\delta(\perp, l) = \perp$ para todo $l \in L$, y $\xi_s(\perp) = \xi_w(\perp) = \emptyset$. La utilización de TLDA completos es conveniente para la exposición teórica; las implementaciones reales del transductor y los algoritmos correspondientes no tienen por qué contener una representación explícita del estado de absorción y sus transiciones entrantes y salientes.

Para los TLDA completos, la correspondencia extendida $\delta^* : Q \times L^* \to Q$ (la extensión de δ a las transducciones de L^*) se define simplemente como:

Definición 6.2 (Función de transición extendida de un TLDA) *Se define la extensión de la función de transición de un TLDA a las transducciones del lenguaje L^* como*

$$\begin{aligned} \delta^*(q, \epsilon) &= q \\ \delta^*(q, xl) &= \delta(\delta^*(q, x), l) \quad \forall l \in L, \forall x \in L^* \end{aligned} \tag{6.3}$$

donde ϵ es la cadena vacía sobre L.

Los lenguajes de transducciones que acepta cada transductor T para cada símbolo de preanálisis $\sigma \in \Sigma \cup \{\$\}$ se definen como sigue:

Definición 6.3 (Lenguaje fuertemente aceptado-σ por T) *El lenguaje $\mathcal{L}^s_\sigma(T) \subseteq L^*$ fuertemente aceptado-σ por un TLDA T, es decir, el lenguaje*

[2]Por ejemplo, si el lenguaje de entrada es $\Sigma = \{a\}$, el lenguaje de salida es $\Gamma = \{x, z\}$, el conjunto de etiquetas de las transiciones es $L = \{(a, x), (a, z)\}$ y se define la función de transición del transductor como $\delta(q_1, (a, x)) = q_2$ y $\delta(q_1, (a, z)) = q_1$, el transductor resultante es indeterminista respecto del lenguaje Σ, pero determinista respecto de L.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

de las transducciones fuertemente aceptadas por T con σ como símbolo de validación se define como

$$\mathcal{L}^s_\sigma(T) = \{t : \sigma \in \xi_s(\delta^*(q_I, t))\} \tag{6.4}$$

Definición 6.4 (Lenguaje débilmente aceptado-σ por T) *El lenguaje $\mathcal{L}^w_\sigma(T) \subseteq L^*$ débilmente aceptado-σ por un TLDA T, es decir, el lenguaje de las transducciones débilmente aceptadas por un T con σ como símbolo de validación se define como*

$$\mathcal{L}^w_\sigma(T) = \{t : \sigma \in \xi_w(\delta^*(q_I, t))\} \tag{6.5}$$

donde σ es un símbolo leído de la entrada que se utiliza para validar la transducción que el transductor ha construido hasta el momento de leer dicho símbolo. Nótese que según la definición 6.1, para cada $\sigma \in \Sigma \cup \{\$\}$, $\mathcal{L}^s_\sigma(T) \cap \mathcal{L}^w_\sigma(T) = \emptyset$. Las familias de lenguajes derechos correspondientes a cada estado q, $\mathcal{R}^s_\sigma(T, q)$ y $\mathcal{R}^w_\sigma(T, q)$, se definen de forma análoga sustituyendo q_I por q:

Definición 6.5 (Lenguaje derecho fuerte del estado q) *El lenguaje $\mathcal{R}^s_\sigma(T, q) \subseteq L^*$ fuerte derecho del estado q, es decir, el lenguaje derecho de las transducciones fuertemente aceptadas desde el estado q de un TLDA se define como*

$$\mathcal{R}^s_\sigma(T, q) = \{t : \sigma \in \xi_s(\delta^*(q, t))\} \tag{6.6}$$

Definición 6.6 (Lenguaje derecho débil del estado q) *El lenguaje $\mathcal{R}^w_\sigma(T, q) \subseteq L^*$ débil derecho del estado q, es decir, el lenguaje derecho de las transducciones débilmente aceptadas desde el estado q de un TLDA se define como*

$$\mathcal{R}^w_\sigma(T, q) = \{t : \sigma \in \xi_w(\delta^*(q, t))\} \tag{6.7}$$

Es muy fácil construir un transductor de letras aumentado (TLA) no determinista a partir de un diccionario morfológico insertando las transducciones individuales correspondientes a cada entrada en el diccionario morfológico, como `(panes:pan<n><m><sg>)`, alineado por ejemplo como `(p:p)(a:a)(n:n)(e:<n>)(θ:<m>)(s:<pl>)` a partir del estado inicial. Ya que los TLA son isomorfos a los automátas de estados finitos, estos pueden determinizarse respecto al alfabeto L para convertirlos en un TLDA y ser minimizados utilizando versiones adaptadas (sección 5.2) de algoritmos existentes para autómatas finitos (Hopcroft y Ullman 1979, p. 68).

Un TLDA mínimo es aquel en el que no hay dos estados equivalentes. De forma análoga a un autómata finito determinista, estados equivalentes son

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

aquellos que tienen los mismos conjuntos de lenguajes derechos de transducciones, es decir, para dos estados q y r de un TLDA T:

$$q \equiv r \Leftrightarrow \forall \sigma \in \Sigma \cup \{\$\}\ \mathcal{R}^s_\sigma(T,q) = \mathcal{R}^s_\sigma(T,r) \wedge \mathcal{R}^w_\sigma(T,q) = \mathcal{R}^w_\sigma(T,r) \quad (6.8)$$

Nótese que la minimización no puede cambiar los alineamientos realizados originariamente entre las formas superficiales y las formas léxicas; esto se debe a que, a diferencia de lo que ocurre en los algoritmos de minimización para TEF p-subsecuenciales de la definición 3.2 (Mohri 1997), la minimización opera a nivel de los pares del alfabeto L.

6.2.2 EL TLDA de transducción única

En muchos casos, el diccionario morfológico crecerá a través de la adición de una transducción $t \in L^*$ —correspondiente a un nuevo par (forma superficial, forma léxica)— que tiene que ser fuertemente aceptada, pero condicionada a un conjunto de preanálisis $S^t \subseteq \Sigma \cup \{\$\}$. Por tanto es conveniente definir el *TLDA* (completo) *de transducción única* para una transducción t fuertemente aceptada con el conjunto de preanálisis S^t, denotado como T^t:

Definición 6.7 (TLDA de transducción única fuertemente aceptada) *El TLDA canónico de una única transducción fuertemente aceptada se define como*

$$T^t = (Q^t, L, \delta^t, q^t_I, \xi^t_s, \xi^t_w) \quad (6.9)$$

tal que $\mathcal{L}^s_\sigma(T^t) = \{t\}$ *para todo* $\sigma \in S^t$. *Este TLDA*[3] *tiene*

- $Q^t = \Pr(t) \cup \{\perp^t\}$, *donde* $\Pr(t)$ *es el conjunto de todos los prefijos de la transducción* t *y* $\perp^t$ *denota el estado de absorción,*
- $q^t_I = \epsilon$,
- $\xi^t_w(x) = \emptyset$ *para todo* $x \in Q^t$,
- ξ^t_s *tal que*

$$\xi^t_s(x) = \begin{cases} S^t & si\ \ x = t \\ \emptyset & en\ otro\ caso \end{cases}$$

- δ^t *se define:*

$$\forall x \in Q^t,\ l \in L:\ \delta^t(x,l) = \begin{cases} xl & si\ \ x, xl \in \Pr(t) \\ \perp^t & en\ otro\ caso \end{cases}$$

[3] El TLDA de una sola transducción se puede definir de forma análoga para la aceptación débil.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Nótese que el TLDA de transducción única para una transducción t tiene $|Q^t| = |t| + 2$ estados.

6.3 El TLDA producto

Dados dos TLDA $T^{(1)}$ y $T^{(2)}$, conviene definir el *producto* de estos dos transductores.

Definición 6.8 (TLDA producto) *El TLDA producto T' de dos TLDA $T^{(1)}$ y $T^{(2)}$ se define como*

$$T' = (Q', L, \delta', q'_I, \xi'_s, \xi'_w) \tag{6.10}$$

donde

- $Q' = Q^{(1)} \times Q^{(2)}$
- $\delta' : Q' \times L \to Q'$ *se define como*

  $\forall l \in L,\ \forall (q^{(1)}, q^{(2)}) \in Q',$
  $\delta'((q^{(1)}, q^{(2)}), l) = (\delta^{(1)}(q^{(1)}, l), \delta^{(2)}(q^{(2)}, l))$

- $q'_I = (q_I^{(1)}, q_I^{(2)})$
- ξ'_s *y* ξ'_w *quedan por definir según la finalidad del transductor producto. En general será función de las correspondientes funciones de* $T^{(1)}$ *y* $T^{(2)}$.

  $\xi'_s = \Xi_s(\xi_s^{(1)}(q), \xi_s^{(2)}(q), \xi_w^{(1)}(q), \xi_w^{(2)}(q))$
  $\xi'_w = \Xi_w(\xi_w^{(1)}(q), \xi_w^{(2)}(q), \xi_s^{(1)}(q), \xi_s^{(2)}(q))$

Por tanto las funciones ξ'_s y ξ'_w se definen en función del lenguaje que se desea que acepte el transductor producto.

Por ejemplo, si se desea que el nuevo TLDA acepte todas las transducciones fuertes que aceptaban $T^{(1)}$ y $T^{(2)}$ (quitando las correspondientes débiles, si se ha producido alguna coincidencia), es decir, si se desea que acepte $\forall \sigma \in \Sigma \cup \{\$\}$ los lenguajes:

$$\begin{aligned} \mathcal{L}_\sigma^s(T') &= \mathcal{L}_\sigma^s(T^{(1)}) \cup \mathcal{L}_\sigma^s(T^{(2)}) \\ \mathcal{L}_\sigma^w(T') &= (\mathcal{L}_\sigma^w(T^{(1)}) - \mathcal{L}_\sigma^s(T^{(2)})) \cup (\mathcal{L}_\sigma^w(T^{(2)}) - \mathcal{L}_\sigma^s(T^{(1)})) \end{aligned} \tag{6.11}$$

las funciones que asignan los conjuntos de validación se definen $\forall (q^{(1)}, q^{(2)}) \in Q'$ como:

$$\begin{aligned} \xi'_s((q^{(1)}, q^{(2)})) &= \xi_s^{(1)}(q^{(1)}) \cup \xi_s^{(2)}(q^{(2)}) \\ \xi'_w((q^{(1)}, q^{(2)})) &= [\xi_w^{(1)}(q^{(1)}) - \xi_s^{(2)}(q^{(2)})] \cup [\xi_w^{(2)}(q^{(2)}) - \xi_s^{(1)}(q^{(1)})] \end{aligned} \tag{6.12}$$

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Estas elecciones son fácilmente demostrables si se define el lenguaje de las transducciones que llevan al estado q.

Definición 6.9 (Lenguaje izquierdo del estado q) *Se define el lenguaje izquierdo del estado q de un TLDA como*

$$\mathcal{H}(q) = \{t \in L^* : \delta^*(q_I, t) = q\}. \tag{6.13}$$

Es fácil demostrar que

$$\mathcal{H}'((q^{(1)}, q^{(2)})) = \mathcal{H}^{(1)}(q^{(1)}) \cap \mathcal{H}^{(2)}(q^{(2)}); \tag{6.14}$$

ya que los transductores $T^{(1)}$ y $T^{(2)}$ son completos se cumple:

$$\bigcup_{q^{(1)} \in Q^{(1)}} \mathcal{H}^{(1)}(q^{(1)}) = \bigcup_{q^{(2)} \in Q^{(2)}} \mathcal{H}^{(2)}(q^{(2)}) = L^* \tag{6.15}$$

Por tanto $\forall \sigma \in \Sigma$ el lenguaje fuertemente aceptado-σ por $T^{(1)}$ se puede definir en los siguientes términos:

$$\begin{aligned}
&\mathcal{L}^s_\sigma(T^{(1)}) = \textstyle\bigcup_{q^{(1)} \in Q^{(1)}:\, \sigma \in \xi^{(1)}_s(q^{(1)})} \mathcal{H}^{(1)}(q^{(1)}) = \\
&\textstyle\bigcup_{q^{(1)} \in Q^{(1)}:\, \sigma \in \xi^{(1)}_s(q^{(1)})} \mathcal{H}^{(1)}(q^{(1)}) \cap L^* = \\
&\textstyle\bigcup_{q^{(1)} \in Q^{(1)}:\, \sigma \in \xi^{(1)}_s(q^{(1)})} \mathcal{H}^{(1)}(q^{(1)}) \cap \bigcup_{q^{(2)} \in Q^{(2)}} \mathcal{H}^{(2)}(q^{(2)}) = \\
&\textstyle\bigcup_{q^{(1)} \in Q^{(1)}:\, \sigma \in \xi^{(1)}_s(q^{(1)})} \bigcup_{q^{(2)} \in Q^{(2)}} \mathcal{H}^{(1)}(q^{(1)}) \cap \mathcal{H}^{(2)}(q^{(2)}) = \\
&\textstyle\bigcup_{q^{(1)} \in Q^{(1)}:\, \sigma \in \xi^{(1)}_s(q^{(1)})} \bigcup_{q^{(2)} \in Q^{(2)}} \mathcal{H}'((q^{(1)}, q^{(2)})) = \\
&\textstyle\bigcup_{(q^{(1)}, q^{(2)}) \in Q^{(1)} \times Q^{(2)}:\, \sigma \in \xi^{(1)}_s(q^{(1)})} \mathcal{H}'((q^{(1)}, q^{(2)}))
\end{aligned} \tag{6.16}$$

donde se ha hecho uso de las igualdades (6.14) y (6.15). De igual forma se puede definir $\mathcal{L}^s_\sigma(T^{(2)})$, de manera que la unión de estos lenguajes se puede expresar como:

$$\begin{aligned}
&\forall \sigma \in \Sigma,\ \mathcal{L}^s_\sigma(T^{(1)}) \cup \mathcal{L}^s_\sigma(T^{(2)}) = \\
&\textstyle\bigcup_{(q^{(1)}, q^{(2)}) \in Q':\, \sigma \in \xi^{(1)}_s(q^{(1)})} \mathcal{H}'((q^{(1)}, q^{(2)})) \cup \\
&\textstyle\bigcup_{(q^{(1)}, q^{(2)}) \in Q':\, \sigma \in \xi^{(2)}_s(q^{(2)})} \mathcal{H}'((q^{(1)}, q^{(2)})) = \\
&\textstyle\bigcup_{(q^{(1)}, q^{(2)}) \in Q':\, \sigma \in (\xi^{(1)}_s(q^{(1)}) \cup \xi^{(2)}_s(q^{(2)}))} \mathcal{H}'((q^{(1)}, q^{(2)}))
\end{aligned} \tag{6.17}$$

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Es decir, el lenguaje fuertemente aceptado-σ por el nuevo transductor T', se define como:

$$\mathcal{L}^s_\sigma(T') = \bigcup_{(q^{(1)},q^{(2)})\in Q':\ \sigma\in\xi'_s((q^{(1)},q^{(2)}))} \mathcal{H}'((q^{(1)},q^{(2)})) \qquad (6.18)$$

donde para todo $\sigma \in \Sigma$:

$$\xi'_s((q^{(1)},q^{(2)})) = \xi^{(1)}_s(q^{(1)}) \cup \xi^{(2)}_s(q^{(2)}).$$

La función $\xi'_w((q^{(1)},q^{(2)}))$ se obtiene siguiendo un proceso análogo.

Si se desea que el nuevo TLDA acepte todas las transducciones (fuertes y débiles) que aceptaba $T^{(1)}$, excepto las transducciones (fuertes y débiles) aceptadas por $T^{(2)}$, es decir, que acepte $\forall\sigma \in \Sigma \cup \{\$\}$ los lenguajes:

$$\begin{aligned}\mathcal{L}^s_\sigma(T') &= \mathcal{L}^s_\sigma(T^{(1)}) - \mathcal{L}^s_\sigma(T^{(2)})\\ \mathcal{L}^w_\sigma(T') &= \mathcal{L}^w_\sigma(T^{(1)}) - \mathcal{L}^w_\sigma(T^{(2)})\end{aligned} \qquad (6.19)$$

las funciones que asignan los conjuntos de validación se definen $\forall(q^{(1)},q^{(2)}) \in Q'$ como:

$$\begin{aligned}\xi'_s((q^{(1)},q^{(2)})) &= \xi^{(1)}_s(q^{(1)}) - \xi^{(2)}_s(q^{(2)})\\ \xi'_w((q^{(1)},q^{(2)})) &= \xi^{(1)}_w(q^{(1)}) - \xi^{(2)}_w(q^{(2)})\end{aligned} \qquad (6.20)$$

La adición y la eliminación de transducciones se ven en la siguiente sección como un caso particular del producto, donde $T^{(1)}$ es el transductor original y $T^{(2)}$ el de transducción única correspondiente a la transducción t que se desea añadir o eliminar. Para ello, se establecerá primero qué tipo de estados se generan en el producto en cada caso y se les asignará a continuación la función ξ.

6.4 Adición de una transducción

Dado un TLDA T, es fácil construir un nuevo TLDA T' completo tal que acepte todas las transducciones débiles y fuertes aceptadas por T más una nueva transducción t fuertemente aceptada con un conjunto de validación S^t; esto es; $\forall\sigma \in \Sigma \cup \{\$\}$,

$$\begin{aligned}\mathcal{L}^s_\sigma(T') &= \mathcal{L}^s_\sigma(T) \cup \begin{cases}\{t\} & \text{si } \sigma \in S^t\\ \emptyset & \text{otro caso}\end{cases}\\ \mathcal{L}^w_\sigma(T') &= \mathcal{L}^w_\sigma(T) - \begin{cases}\{t\} & \text{si } \sigma \in S^t\\ \emptyset & \text{otro caso}\end{cases}\end{aligned} \qquad (6.21)$$

El nuevo TLDA T' se define como el producto de los transductores T y T^t. Antes de discutir las funciones ξ'_w y ξ'_s, es conveniente observar que

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

los estados de T', obtenidos mediante la aplicación de la definición (6.8), se pueden separar en cuatro grupos (la nomenclatura está inspirada en la utilizada por (Carrasco y Forcada 2002; Daciuk et al. 2000)):

- Estados de la forma $(q, \perp^t)$, con $q \in Q - \{\perp\}$, equivalentes a aquellos estados de T que no son alcanzables por ningún prefijo de t; estos se llamarán estados *intactos* porque tienen la misma estructura de transiciones que sus equivalentes en T; esto es, si $\delta(q, l) = r$ entonces $\delta'((q, \perp^t), l) = (r, \perp^t)$. Por tanto, tendrán los mismos lenguajes derechos $\mathcal{R}^s_\sigma(T', (q, \perp^t)) = \mathcal{R}^s_\sigma(T, q)$, y $\mathcal{R}^w_\sigma(T', (q, \perp^t)) = \mathcal{R}^w_\sigma(T, q)$, ya que todas sus transiciones de salida tienen como destino un estado intacto; además, como el transductor T era mínimo, cada estado $(q, \perp^t)$ tiene un lenguaje derecho diferente; es decir, dos estados intactos nunca podrán ser fusionados durante el proceso de minimización (los estados intactos solo pueden ser eliminados si son inalcanzables, como se describe más adelante). Para un transductor (diccionario) grande T, éstos son la gran mayoría de estados (el número de estados intactos está entre $|Q| - |t| - 1$ y $|Q|$); por tanto en la práctica es conveniente considerar T' como una modificación de T y, por ello, será tratado como tal en los algoritmos siguientes.

- Estados de la forma (q, x) con $q \in Q - \{\perp\}$ y $x \in \Pr(t)$ y tales que $\delta^*(q_I, x) = q$; este tipo de estados se llamarán estados *clonados*; en particular, el nuevo estado inicial $q'_I = (q_I, \epsilon)$ es también un estado clonado, ya que $\delta^*(q_I, \epsilon) = q_I$. Los estados restantes de $(Q - \{\perp\}) \times \Pr(t)$ —la gran mayoría de los estados de $Q \times Q^t$— pueden ser descartados con seguridad ya que son inalcanzables desde el nuevo estado inicial q'_I. Los estados clonados son versiones modificadas de los estados originales $q \in Q - \{\perp\}$: todas sus transiciones de salida apuntan a los correspondientes estados intactos en Q', $(\delta(q, l), \perp^t)$, excepto para la transición con el símbolo l: $xl \in \Pr(t)$, que ahora apunta al correspondiente estado clonado $(\delta(q, l), xl)$, es decir:

$$\delta'((q, x), l) = \begin{cases} (\delta(q, l), xl) & \text{si } xl \in \Pr(t) \\ (\delta(q, l), \perp^t) & \text{en otro caso} \end{cases} \tag{6.22}$$

  Hay como máximo $|t| + 1$ estados clonados.

- Estados de la forma $(\perp, x)$ con $x \in \Pr(t)$. Estos estados se llamarán estados *de cola*; los estados de este tipo aparecen sólo si en el transductor original $\delta^*(q_I, x) = \perp$ para algún $x \in \Pr(t)$. Hay como máximo $|t|$ estados de cola.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

q'	intactos $((q, \perp^t))$	clonados $((q, x))$	de cola $((\perp, x))$
$\xi'_s(q') =$	$\xi_s(q)$	$\xi_s(q) \cup S^t$ si $x = t$ $\xi_s(q)$ otro caso	S^t si $x = t$ $\emptyset$ otro caso
$\xi'_w(q') =$	$\xi_w(q)$	$\xi_w(q) - S^t$ si $x = t$ $\xi_w(q)$ otro caso	$\emptyset$

Tabla 6.1: Tabla de conjuntos de preanálisis para la adición de una transducción.

- Un estado de absorción $\perp' = (\perp, \perp^t)$ con $\xi'_s(\perp') = \xi'_w(\perp') = \emptyset$.

El nuevo TLDA T', que para diccionarios grandes es sólo un poco más grande que T, tiene los conjuntos de preanálisis que se especifican en la tabla 6.1, que se deducen de las ecuaciones (6.12) y (6.21).

Claramente, con esta elección los lenguajes derechos de los estados intactos de T' son los mismos de sus correspondientes en T: $\mathcal{R}^s_\sigma(T', (q, \perp^t)) = \mathcal{R}^s_\sigma(T, q)$ y $\mathcal{R}^w_\sigma(T', (q, \perp^t)) = \mathcal{R}^w_\sigma(T, q)$, para todo $\sigma \in \Sigma \cup \{\$\}$; puesto que el TLDA original era ya mínimo, esto significa que la minimización no afectará a los estados intactos, que normalmente son la mayoría de Q', ni al estado de absorción. Esta es la principal razón de la eficiencia de los algoritmos presentados aquí: la minimización se puede llevar a cabo en un número pequeño de operaciones. No es difícil demostrar que la minimización puede llevarse a cabo inicializando un registro R con todos los estados intactos y el estado de absorción y comprobar, uno por uno, los estados de cola y clonados con los estados de R (empezando por el último estado de cola $(\perp, t)$, o si no existe, el último estado clonado (q, t), e ir descendiendo en $\Pr(t)$) y añadirlos al registro si no se encuentra un estado equivalente en R (realizar esta comprobación hacia atrás evita tener que comprobar la equivalencia de estados visitando sus descendientes recursivamente). Esta es la parte más costosa de los algoritmos y tiene una complejidad asintótica de $O(|Q||t|)$, si se considera que $|R|$ es del orden de $|Q|$. La minimización (que incluye la eliminación de los estados no alcanzables de T) se muestra como parte del algoritmo de adición de una transducción.

El registro R de estados, que no necesita minimización, se inicializa con Q (realmente $(Q \times \{\perp_t\})$). El algoritmo 6.1 tiene tres fases:

- Primero, se construyen y añaden a Q los estados de cola y clonados utilizando la función clonar() para todos los prefijos de t. La función devuelve un estado clonado (con todas sus transiciones creadas) si el argumento no es un estado de absorción (es de $Q - \{\perp\}$), o un estado de cola si opera sobre el estado de absorción $\perp \in Q$; en ambos casos, se actualizan los conjuntos de preanálisis de acuerdo a la tabla 6.1.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

- En segundo lugar, aquellos estados intactos que se han vuelto inalcanzables por haberse designado q'_I como nuevo estado inicial son borrados de Q y de R y el estado inicial se reemplaza por su clon (los estados inalcanzables son simplemente aquellos que no tienen transiciones de entrada tal como han sido construidos por el algoritmo o como consecuencia de borrar otros estados inalcanzables). Nótese que únicamente los estados intactos en $\delta(q_I, x)$ para algún $x \in \Pr(t)$ pueden llegar a ser inalcanzables como resultado de haber sido clonados; por tanto, si los estados se visitan en orden ascendente de la longitud de x, la función inalcanzable() sólo tiene que comprobar la ausencia de transiciones entrantes.

```
funcion equiv(p,q)
    si (ξs(p) ≠ ξs(q)) o (ξw(p) ≠ ξw(q)) entonces
        devuelve falso
    para todo símbolo σ ∈ Σ
        si δ(p,σ) ≠ δ(q,σ)
            devuelve falso
        fin_si
    fin_para
    devuelve cierto
fin_funcion
```

Figura 6.1: La función equiv().

```
funcion reemplazar_registrar(q)
    si ∃p ∈ R : equiv(p,q) entonces
        fusionar(p,q)
    si no
        R ← R ∪ {q}
    fin_si
fin_funcion
```

Figura 6.2: La función reemplazar_registrar().

- Finalmente, los estados clonados y de cola se comparan (empezando por el último estado) con los del registro R utilizando la función

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Algoritmo 6.1: Añadir una transducción a un TLDA manteniéndolo mínimo

Entrada: $T = (Q, L, \delta, q_I, \xi_s, \xi_w)$ (mínimo, completo),
$t \in L^*$, con conjunto de preanálisis fuerte S^t
Salida: $T' = (Q', L, \delta', q'_I, \xi'_s, \xi'_w)$ mínimo, completo,
y tal que cumple las condiciones (6.21)

$R \leftarrow Q$ [inicialización registro]
$q'_I \leftarrow$ clonar(q_I) [clona el estado inicial;
actualiza los conjuntos de validación]
$q_{\text{ult}} \leftarrow q'_I$
para $i = 1$ hasta $|t|$
    $q \leftarrow$ clonar$(\delta^*(q_I, t_1 \cdots t_i))$ [crea los estados clonados y de cola;
    actualiza los conjuntos de validación]
    $\delta(q_{\text{ult}}, t_i) \leftarrow q$
    $q_{\text{ult}} \leftarrow q$
fin_para
$i \leftarrow 1$
$q_{\text{act}} \leftarrow q_I$
mientras($i \leq |t|$ e inalcanzable(q_{act}))
    $q_{\text{sig}} \leftarrow \delta(q_{\text{act}}, t_i)$
    $Q \leftarrow Q - \{q_{\text{act}}\}$ [borra los estados inalcanzables de Q
    y actualiza las transiciones en δ]
    $R \leftarrow R - \{q_{\text{act}}\}$ [los borra también del registro]
    $q_{\text{act}} \leftarrow q_{\text{sig}}$
    $i \leftarrow i + 1$
fin_mientras
si inalcanzable(q_{act})
    $Q \leftarrow Q - \{q_{\text{act}}\}$
    $R \leftarrow R - \{q_{\text{act}}\}$
fin_si
$q_I \leftarrow q'_I$ [sustituye el estado inicial]
para $i = |t|$ hasta 1
    reemplazar_registrar$(\delta^*(q_I, t_1 \cdots t_i))$ [comprueba los estados clonados
    y de cola uno por uno]
fin_para
devuelve $T = (Q, L, \delta, q_I, \xi_s, \xi_w)$

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

reemplazar_registrar(), que es una adaptación de la versión no recursiva del algoritmo 2 de Daciuk et al. (2000) y que se puede ver en la figura 6.2. Si el estado argumento q tiene un estado equivalente p en R, la función fusionar(p, q) se encarga de redireccionar a p aquellas transiciones que llegan a q; si no, simplemente se añade q al registro. La equivalencia entre estados se comprueba con la función equiv(p, q), que puede verse en la figura 6.1, que comprueba la equivalencia de los estados comparando:

1. Si los conjuntos de validación ξ_s y ξ_w son iguales para ambos estados.
2. Si sus transiciones de salida llegan a los mismos estados en R.

Nótese que las transiciones de salida no pueden llegar a estados distintos que sean equivalentes, ya que no hay estados equivalentes en el registro ($\forall p, q \in R, \text{equiv}(p, q) \Rightarrow p = q$) y que la minimización hacia atrás garantiza que los estados no tienen transiciones a estados que no pertenezcan al registro.

Por último, se devuelve el nuevo TLDA (mínimo). En la implementación real, los estados de absorción no son almacenados explícitamente; ello conlleva pequeñas diferencias en la implementación de las funciones clonar() y reemplazar_registrar().

En la siguiente sección se ilustra el funcionamiento de este algoritmo con un ejemplo.

6.4.1 Un ejemplo

Por ejemplo, al diccionario de la figura 6.3, que ya ha sido transformado en el TLDA mínimo de la figura 6.5 (en el que, por motivos de claridad, no se han dibujado ni el estado de absorción, ni las transiciones que llegan a él, ni los conjuntos de preanálisis, que son todos iguales a aquellos caracteres que no estén definidos en el alfabeto del lenguaje aceptado, para los estados marcados como de aceptación y vacíos para los demás), se desea añadir la transducción (`paso:paso<n><m><sg>`), cuyo TLDA[4] de transducción única se puede ver en la figura 6.4; esta transducción tiene como conjunto de preanálisis

[4]Los estados de este transductor se han reetiquetado con números 1', 2', etc. debido a que las etiquetas correspondientes a los prefijos usados en la definición 6.7 son excesivamente largas.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

```
!alfabet a-zçáéíóúàèìòùäëïöüâêîôûñ
#etiquetas morfológicas
!simbol <pr>;
!simbol <n>;
!simbol <f>;
!simbol <m>;
!simbol <sg>;

#diccionario
%dic
(para:para<pr>);
(pasa:pasa<n><f><sg>);
(pase:pase<n><m><sg>)
```

Figura 6.3: Ejemplo de diccionario morfológico.

fuerte, S^t, el conjunto de todos aquellos caracteres que no sean alfabéticos. [5]

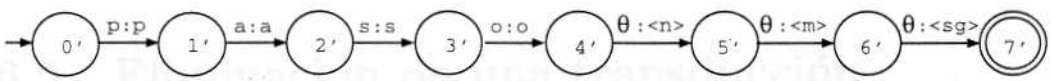


Figura 6.4: TLDA de transducción única para la transducción `paso:paso<n><m><sg>`.

La aplicación de la primera fase del algoritmo para la adición de una transducción da como resultado el transductor sin minimizar de la figura 6.6. Este transductor tiene, además del conjunto de estados intactos $\{(0, \perp^t), ..., (11, \perp^t)\}$, cuatro estados clonados $\{(0, 0'), (1, 1'), (2, 2'), (4, 3')\}$ y cuatro estados de cola $\{(\perp, 4'), (\perp, 5'), (\perp, 6'), (\perp, 7')\}$.

Al aplicar la segunda fase del algoritmo se considera el estado $(0, 0')$ como el nuevo estado inicial del transductor, de forma que los estados dibujados con un trazo más grueso en la figura 6.6 se convierten en estados inalcanzables (no tienen transiciones de entrada) y serán eliminados en el siguiente orden: $(0, \perp^t)$, $(1, \perp^t)$, $(2, \perp^t)$ y $(4, \perp^t)$.

Por último, en la tercera fase, se comprueba si los estados clonados y de cola tienen algún estado equivalente en R, empezando por el estado $(\perp, 7')$, que es equivalente al estado $(11, \perp^t)$ del registro: por tanto las transiciones que llegan a $(\perp, 7')$ son redirigidas al estado $(11, \perp^t)$. De esta forma se

[5] Es el mismo conjunto de preanálisis que tienen el resto de transducciones fuertemente aceptadas del transductor.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

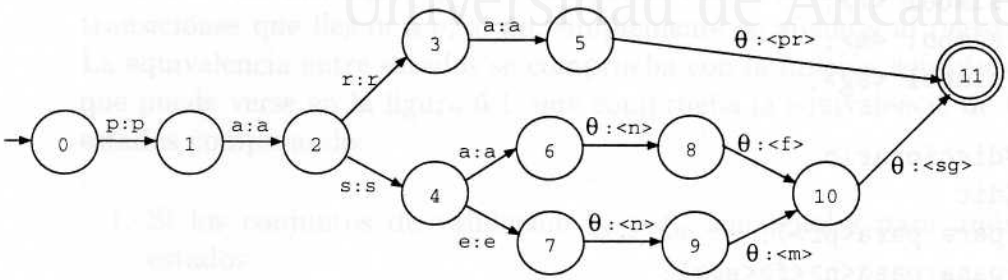


Figura 6.5: TLDA mínimo correspondiente al diccionario de la figura 6.3.

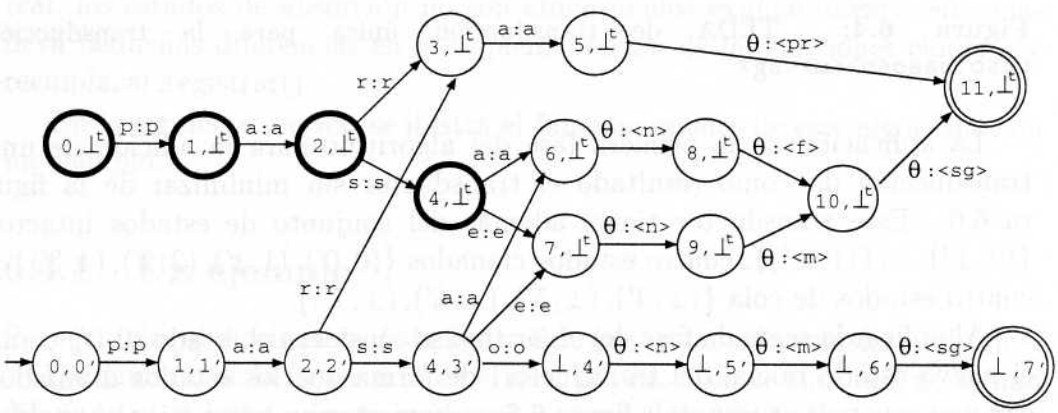


Figura 6.6: Transductor sin minimizar después de añadir la transducción de la figura 6.4 al TLDA de la figura 6.5. Los estados dibujados con un trazo más grueso son estados que se convierten en inalcanzables, y que serán eliminados en el siguiente orden: $(0, \perp^t)$, $(1, \perp^t)$, $(2, \perp^t)$ y $(4, \perp^t)$.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

comprueba que los estados $(\perp, 6')$, $(\perp, 5')$ y $(\perp, 4')$ son equivalentes, respectivamente, a los estados $(10, \perp^t)$, $(9, \perp^t)$, $(7, \perp^t)$ en este orden, y que los estados $(4, 3')$, $(2, 2')$, $(1, 1')$ y $(0, 0')$ no tienen estados equivalentes en el registro y son añadidos a R en este orden, con lo que se obtiene el TLDA mínimo que se puede ver en la figura 6.7, después de renumerar convenientemente los estados.

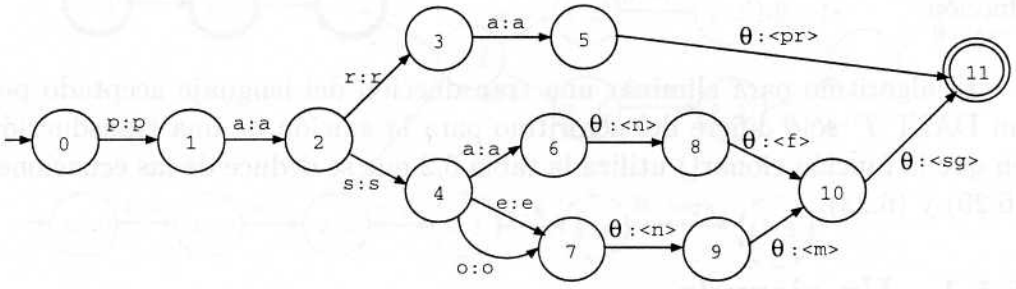


Figura 6.7: TLDA mínimo después de añadir una transducción correspondiente al TLDA de la figura 6.6.

6.5 Eliminación de una transducción

De nuevo, dado un TLDA T, es fácil construir un TLDA T' completo tal que acepte todas las transducciones fuertes y débiles aceptadas por T menos la transducción t fuertemente aceptada con un conjunto de validación S^t, es decir; $\forall \sigma \in \Sigma \cup \{\$\}$,

$$\begin{aligned} \mathcal{L}^s_\sigma(T') &= \mathcal{L}^s_\sigma(T) - \begin{cases} \{t\} & \text{si } \sigma \in S^t \\ \emptyset & \text{otro caso} \end{cases} \\ \mathcal{L}^w_\sigma(T') &= \mathcal{L}^w_\sigma(T) \end{aligned} \tag{6.23}$$

El TLDA resultante tiene el mismo conjunto de estados alcanzables en Q' que el de la sección anterior, y por tanto las mismas propiedades cercanas a la minimalidad; por tanto, ya que se supone $t \in \mathcal{L}^s_\sigma(T)$ para todo $\sigma \in S^t$, no se formarán estados de cola (de hecho, si $t \notin \mathcal{L}^s_\sigma(T)$, se crea una cola de no aceptación con todos sus estados eventualmente equivalentes a $(\perp, \perp^t)$).

Nótese que si una transducción t con un conjunto de validación S^t se elimina de un TLDA, es posible que t sea aún fuertemente aceptada con un conjunto de validación menor (situación improbable en el mantenimiento normal de un diccionario, ya que los conjuntos de preanálisis diferentes suelen ser pocos).

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

q'	intactos $((q, \perp^t))$	clonados $((q, x))$	de cola $((\perp, x))$
$\xi'_s(q') =$	$\xi_s(q)$	$\xi_s(q) - S^t$ si $x = t$ $\xi_s(q)$ otro caso	$\emptyset$
$\xi'_w(q') =$	$\xi_w(q)$	$\xi_w(q)$	$\emptyset$

Tabla 6.2: Tabla de conjuntos de preanálisis para la eliminación de una transducción.

El algoritmo para eliminar una transducción del lenguaje aceptado por un DALT T' sólo difiere del algoritmo para la adición de una transducción en que la función clonar() utiliza la tabla 6.2 que se deduce de las ecuaciones (6.20) y (6.23).

6.5.1 Un ejemplo

Por ejemplo, si se desea eliminar la transducción correspondiente a la entrada `(para:para<pr>)` del TLDA de la figura 6.5 (cuyo TLDA de una sola transducción se puede ver en la figura 6.8), el transductor resultante de aplicar la primera fase del algoritmo se puede ver en la figura 6.9. Nótese que el estado $(11, 5')$ tiene como $\xi_s = \emptyset$, ya que se desea eliminar la cadena. Al aplicar la segunda fase del algoritmo se considera el estado $(0, 0')$ como el nuevo estado inicial del transductor, por tanto los estados $(0, \perp^t)$, $(1, \perp^t)$, $(2, \perp^t)$, $(3, \perp^t)$ y $(5, \perp^t)$ (dibujados con un trazo más grueso) se convierten en estados inalcanzables y son borrados por este orden. A continuación se comprueban los estados que corresponden al sufijo de la transducción (dibujados con un trazo discontinuo); para ello se recorre la lista de estados clonados empezando por el último, el estado $(11, 5')$, y se comprueba si es equivalente a alguno de los estados del registro R, de forma que se encuentra que es equivalente al estado de absorción $(\perp, \perp^t)$ y se fusiona con él (es decir, desaparecen del diagrama); de esta forma se fusionan también los estados $(5, 4')$ y $(3, 3')$ por este orden. Por último, en la tercera fase del algoritmo, se comprueba si los estados clonados restantes, $(2, 2')$, $(1, 1')$ y $(0, 0')$ tienen algún estado equivalente en R: como no se encuentra ningún estado equivalente, estos estados son añadidos al registro en este orden. El TLDA mínimo resultante se puede ver en la figura 6.10.

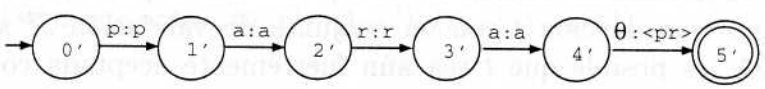


Figura 6.8: TLDA de transducción única para la transducción `para:para<pr>`.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

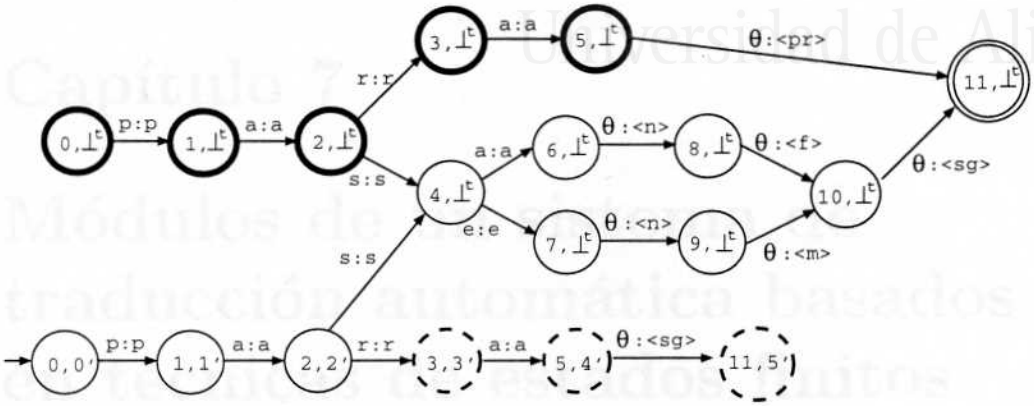


Figura 6.9: Transductor sin minimizar después de eliminar la transducción de la figura 6.8 del transductor de la figura 6.5. Los estados dibujados con un trazo más grueso son estados que se convierten en inalcanzables, y que serán eliminados en el orden en que aparecen. Los estados dibujados con trazo discontinuo son estados del sufijo de la transducción que serán finalmente equivalentes al estado de absorción, $(\perp, \perp^t)$ (no expresado explícitamente), y se fusionarán con él.

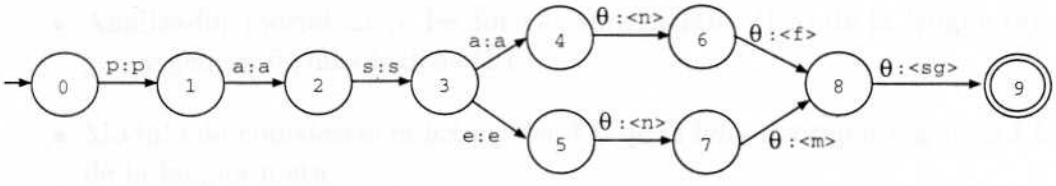


Figura 6.10: TLDA mínimo después de eliminar una transducción correspondiente al TLDA de la figura 6.9.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Volver al Indice/Tornar a l'index

Ya que el algoritmo para eliminar una transducción consultará previamente si la transducción para borrar estaba en $\mathcal{L}^s_\sigma(T)$ para algún σ, la función clonar() generará sólo estados clonados y no estados de cola.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Universitat d'Alacant
Universidad de Alicante

Capítulo 7

Módulos de un sistema de traducción automática basados en técnicas de estados finitos

7.1 Introducción

En este capítulo se describen los módulos que forman parte de un sistema de traducción automática por transferencia morfológica avanzada y que obtienen los datos lingüísticos de un transductor como los descritos en la sección 6.2.1 para realizar el proceso correspondiente. Cabe destacar que una de las opciones en el diseño de de estos módulos es que se comunican con los demás a través de flujo de texto por dos razones: primero, para un diagnóstico de errores fácil durante el desarrollo del sistema y segundo porque es una elección natural cuando se utiliza un sistema operativo orientado a texto como es Linux (o, más genéricamente, Unix). Estos módulos son:

- Analizador morfológico: lee formas superficiales (FS) de la lengua origen y genera formas léxicas (FL).
- Módulo de transferencia léxica: lee FL de la lengua origen y genera FL de la lengua meta.
- Generador morfológico: lee FL de la lengua meta y genera sus correspondientes FS.
- Postgenerador: lee FS de la lengua meta y genera FS de la lengua meta sobre las que ha aplicado reglas ortográficas concernientes a la contracción, apostrofación y guionado.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Cada módulo está compuesto por un programa, codificado en el lenguaje ANSI C, y por un fichero binario que contiene el transductor mínimo correspondiente al diccionario pertinente a cada caso. Su funcionamiento básico consiste en leer el fichero binario, de forma que guarda en memoria el transductor (en el apéndice A se indica la estructura utilizada), y espera a leer texto de la entrada (la cual puede ser un fichero o la entrada estándar) para procesarlo.

Cabe resaltar que para generar el transductor del analizador y generador morfológico de una misma lengua se utiliza un mismo diccionario. Para ello se invierte el orden de las parejas de entrada que forman el diccionario, de forma que las cadenas de entrada pasan a ser de salida, y las cadenas de salida son de entrada.

7.2 Analizador morfológico

El transductor mínimo que se obtiene a partir de un diccionario morfológico concreto se almacena en un fichero binario. El analizador morfológico lee el fichero que contiene el TL y utiliza esta información para analizar un texto en la lengua origen, produciendo para cada palabra reconocida todas sus posibles *formas léxicas* (FL), es decir, sus análisis morfológicos.

El analizador es el primer módulo del sistema de traducción automática; es el responsable de reconocer una entrada (levemente preprocesada para encapsular la información de formato del texto) y producir una salida adecuada para ser procesada por el resto del sistema. Las tareas que realiza el analizador morfológico son:

- Segmentar la entrada en FS y blancos; la extensión de las FS la determina el mismo proceso de análisis (ver más adelante). Como blancos se consideran: el espacio en blanco, el retorno de carro, el tabulador y la información de formato encapsulada por el módulo desformateador del sistema interNOSTRUM (sección 9.4).

- Producir todos los análisis morfológicos (todas las FL posibles) para cada FS reconocida en el diccionario.

- Marcar convenientemente las FS desconocidas (no incluidas en el diccionario).

- Emitir sin modificar aquellos caracteres que no sean reconocidos en el diccionario, es decir, aquellos que no pueden formar parte de palabras o expresiones regulares reconocidas por el diccionario, y tampoco son

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

considerados en él como signos de puntuación (por ejemplo "&" o "%"). Esta operación está implícita en el proceso de análisis.

Para ello, el analizador considera una cadena $w' \in \Gamma^*$ como una transducción *fuerte* (respectivamente *débil*) de una cadena de entrada $w \in \Sigma^*$, si hay como mínimo un camino desde el estado inicial q_I a un estado q tal que forma la transducción (w, w') al concatenar las etiquetas de las transiciones que forman parte del camino y el símbolo de entrada siguiente a w se encuentra en el conjunto de validación $\xi_s(q)$ (resp. $\xi_w(q)$) (sección 6.2.1). En general, puede haber más de una transducción válida para una cadena[1] w.

División en unidades léxicas dirigido por el análisis: Los conjuntos de validación definidos en los TLDA a través de las funciones ξ_s y ξ_w (capítulo 6) permiten a los analizadores morfológicos *tokenizar* la entrada (segmentarla en formas superficiales susceptibles de ser analizadas) y analizarla de forma simultánea (de forma similar a los escáneres léxicos generados por la utilidad `lex` de Unix, Lesk 1975). Los TLDA analizan la entrada manteniendo tres conjuntos análogos:

- Un CSP (*conjunto de salidas parciales*). El CSP inicial contiene el par (q_I, ϵ); un nuevo CSP se construye a partir del anterior después de cada símbolo de entrada y contiene todos los pares (q, z) (estado, cadena de salida) formados por el último estado q alcanzado y su cadena parcial de salida asociada $z \in \Gamma^*$.

- Dos CSV (conjunto de *salidas validadas*), el CSV *fuerte* y el CSV *débil*, que inicialmente están vacíos, contienen los pares (estado, cadena de salida) formados por los estados de aceptación de validación fuerte y débil más recientemente validados y sus cadenas de salida asociadas.

La posición de entrada p del símbolo con el que se alcanza los conjuntos de estados de aceptación en cada CSV también se almacena. Después de leer cada símbolo de entrada σ,

- se construye un nuevo CSP a partir del anterior (el nuevo CSP contiene $(q', z\gamma)$ si (q, z) estaba en el CSP anterior y hay un $\gamma \in (\Gamma \cup \{\epsilon\})$ tal que $\delta^*(q, (\sigma, \gamma)) = q'$); y

- los pares (q, z) del anterior CSP tales que $\sigma \in \xi_s(q)$ (resp. $\sigma \in \xi_w(q)$) (pares validados) se utilizan para sobreescribir el CSV fuerte (resp.

[1] En análisis, esto se correspondería a la *ambigüedad léxica* que presentan los *homógrafos*: formas superficiales que tienen dos o más formas léxicas, por tanto, dos o más análisis morfológicos.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

débil) y su posición p (si ningún par (q, z) es validado, el CSV y p permanecen intactos).

Se lee de la entrada hasta que todos los elementos del actual CSP contienen el estado de absorción $\perp$; entonces, se escriben las cadenas de salida correspondientes al CSV fuerte, y si está vacío, se escriben las cadenas de salida correspondientes al CSV débil; finalmente, se reinicia el TLDA al estado inicial y al carácter inmediatamente posterior a la posición p del último CSV. Si todos los elementos del CSP actual contienen el estado de absorción y los conjuntos CSV (fuerte y débil) están vacíos, se emite el primer símbolo leído de la entrada sin ninguna modificación y se reinicia el TLDA al estado inicial y al carácter inmediatamente posterior[2].

Los estados de aceptación de validación fuerte ($q : \xi_s(q) \neq \emptyset$) se utilizan, por ejemplo, para identificar palabras "normales" usando un conjunto de validación con todos los símbolos que no forman parte de las palabras (de manera que el analizador no pare en `bar` si la entrada es `barra`); en particular, los estados de aceptación fuertes incondicionales ($\xi_s(q) = \Sigma \cup \{\$\}$)se utilizan para identificar tokens como las marcas de puntuación (`:`, `-`), espacios en blanco o formas apostrofadas (`d'`, `l'`). Por otra parte, los estados de aceptación de validación débil ($q : \xi_w(q) \neq \emptyset$) se utilizan para identificar "palabras desconocidas" (como `frifo`) y producir algún tipo de transducción *adivinada* (débil)[3]. Como se ha dicho, si una transducción fuerte ha sido aceptada, las transducciones débiles se ignoran.

Este funcionamiento de izquierda a derecha, que reconoce el patrón más largo hace muy fácil tratar las unidades multipalabra (UMP) (variables o invariables) de la entrada: si una UMP no está completa, el estado de aceptación alcanzado podrá corresponder a una unidad más pequeña, que será identificada y cuya transducción se escribirá en la salida (por ejemplo, si el diccionario contiene "`dar`" y la UMP "`dar clase`", cuando lea "`dar calabazas`", la UMP "`dar clase`" abortará en la primera "`a`" de "`calabazas`", la transducción de "`dar`" se escribirá en la salida y el analizador reanudará el proceso con el texto restante "`calabazas`").

El algoritmo 7.1 que se emplea para realizar el análisis morfológico es una adaptación del algoritmo 3.1. Dicha adaptación viene dada por la presencia de los dos tipos de salida que proporciona el TLDA (fuerte y débil), ya que existe una precedencia en la salida; además en esta adaptación se considera una entrada formada por más de una FS, mientras que el algoritmo anterior

[2]Esto permite emitir sin modificar aquellos caracteres que no se consideran inicio de una FS.

[3]En los analizadores de esta tesis, las transducciones de las palabras desconocidas son las mismas palabras precedidas de un indicador al efecto

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

sólo se analizaba una FS individuales.

7.3 Módulo de transferencia léxica

La finalidad de este módulo es traducir las *formas léxicas* (FL) de la lengua origen a FL de la lengua meta. Las FL que este módulo recibe han sido generadas por el analizador morfológico, que puede producir más de una FL para una FS determinada (como en el caso de los homógrafos), y procesadas por un desambiguador de categorías léxicas (descrito superficialmente en la sección 9.2) que proporciona una sola FL; de esta forma la entrada de este módulo está convenientemente dividida en FL, una por cada FS del texto.

Este módulo lee el transductor de letras, que se ha creado a partir de un diccionario bilingüe (sección 2.3) y procesado como se ha descrito en los capítulos 4 y 5, de un fichero binario y a continuación comienza a procesar la entrada. Un ejemplo de posible entrada para este módulo sería la siguiente:

```
venir<vblex><ifi><3><sg>/ de<pr>/ comer<vblex><inf>/
```

que corresponde a uno de los posibles análisis de la frase *vino de comer* en lenguaje natural.

El procesamiento de la entrada se realiza usando una adaptación del algoritmo 1 utilizado para el análisis morfológico. Las modificaciones realizadas se deben a:

- El tipo de entrada que espera este módulo ya que, como se ha comentado anteriormente, la entrada estará presegmentada en FL y con las palabras desconocidas marcadas, por tanto no necesita realizar preanálisis (sección 6.2.1).

- El funcionamiento intrínseco de este módulo, diferente al del analizador morfológico en el hecho de que los diccionarios no especifican aquí la correspondencia la FLs completas sino entre los prefijos de las mismas que varían al pasar de una lengua a otra (ya que, para muchas formas, hay un sufijo que se copia directamente a la salida). Por ejemplo, considérense las cuatro formas que puede tomar la palabra castellana *alto*, para las cuales se dan, en la tabla 7.1, las formas superficiales y léxicas en ambos idiomas. Puede verse que para efectuar la traducción de cualquiera de ellas sólo hay que traducir el prefijo `alto<adj>` por el prefijo `alt<adj>` y copiar el sufijo restante (la información sobre la flexión de esa forma en particular, por ejemplo `<f><sg>`) sin modificarlo. Por ello:

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Algoritmo 7.1: Analizar morfológicamente un texto utilizando un TLDA

Entrada: $T = (Q, L, \delta, q_I, \xi_s, \xi_w)$ (mínimo),
texto $\sigma[1]\sigma[2]\ldots$

Salida: El texto analizado

$\mathrm{CSP}^{[0]} \leftarrow \widehat{C}_\varepsilon((q_I, \epsilon))$; $\mathrm{CSV}_s \leftarrow \emptyset$; $\mathrm{CSV}_w \leftarrow \emptyset$; $t \leftarrow 0$
mientras $(\sigma[t+1] \neq \$)$
  $\mathrm{CSP}^{[1]} \leftarrow \emptyset$
  $(\forall (q, x) \in \mathrm{CSP}^{[0]},\ \forall \gamma \in (\Gamma \cup \{\epsilon\}))$
    $(r, x\gamma) \leftarrow (\delta(q, (\sigma[t], \gamma)), x\gamma)$
    $\mathrm{CSP}^{[1]} \leftarrow \mathrm{CSP}^{[1]} \cup \widehat{C}_\varepsilon((r, x\gamma))$
  $V_s \leftarrow \emptyset$; $V_w \leftarrow \emptyset$
  $\forall (q, x) \in \mathrm{CSP}^{[1]}$
    si $(\sigma[t+1] \in \xi_s(q))$
      $V_s \leftarrow V_s \cup \{(q, x)\}$
    si $(\sigma[t+1] \in \xi_w(q))$
      $V_w \leftarrow V_w \cup \{(q, x)\}$
  si $V_s \neq \emptyset$
    $\mathrm{CSV}_s \leftarrow V_s$; $p \leftarrow t$
  si no
    si $V_w \neq \emptyset$
      $\mathrm{CSV}_w \leftarrow V_w$; $p \leftarrow t$
  fin_si
  si $(q = \perp \quad \forall (q, x) \in \mathrm{CSP}^{[1]})$
    si $(\mathrm{CSV}_s \neq \emptyset)$
      $\forall (q, x) \in \mathrm{CSV}_s$
        emitir x
      $t \leftarrow p$; $\mathrm{CSV}_s \leftarrow \emptyset$
    si no
      si $(\mathrm{CSV}_w \neq \emptyset)$
        $\forall (q, x) \in \mathrm{CSV}_w$
          emitir x
        $t \leftarrow p$; $\mathrm{CSV}_w \leftarrow \emptyset$
      si no
        emitir $\sigma[t+1]$
      fin_si
    $\mathrm{CSP}^{[0]} \leftarrow \widehat{C}_\varepsilon((q_I, \epsilon))$
    fin_si
  si no
    $\mathrm{CSP}^{[0]} \leftarrow \mathrm{CSP}^{[1]}$
  fin_si
  $t \leftarrow t + 1$

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

FSLO	FLLO	FLLM	FSLM
alto	alto<adj><m><sg>	alt<adj><m><sg>	alt
altos	alto<adj><m><pl>	alt<adj><m><pl>	alts
alta	alto<adj><f><sg>	alt<adj><f><sg>	alta
altas	alto<adj><f><pl>	alt<adj><f><pl>	altes

Tabla 7.1: Traducción de las cuatro formas del adjetivo castellano *alto* al catalán *alt* (FSLO: forma superficial en lengua origen; FLLO: forma léxica en lengua origen; FLLM: forma léxica en lengua meta; FSLM: forma superficial en lengua meta).

- En el diccionario sólo se especifican correspondencias entre prefijos de formas léxicas.
- El transductor sólo lee de la forma léxica en lengua original un prefijo (el más largo que contiene en su diccionario), escribe el prefijo de salida correspondiente, y finalmente copia el resto de la forma léxica sin modificarlo.

Por tanto los cambios realizados en el algoritmo son:

- Inicialmente se comprueba si el carácter a procesar es la marca de palabra desconocida; en tal caso se llama a la subrutina `Paldesco()` que lee la palabra y la emite. Esta operación se podría haber implementado dentro del TLDA, pero resulta más eficiente copiar la palabra de la entrada a salida, que procesarla dentro de un transductor.
- En el proceso de una FL se pueden presentar los siguientes casos:
  - La FL de entrada es completamente procesada en el transductor: en este caso se emite la transducción del CSV (recuérdese que en este caso sólo existe un conjunto de aceptación fuerte).
  - La FL de entrada es procesada sólo parcialmente en el transductor, es decir, se procesa únicamente el prefijo de la FL: en este caso se emite la transducción almacenada en el CSV (la transducción obtenida hasta el momento[4]) y se emite el sufijo de la FL que queda por procesar, que se obtiene invocando a la subrutina `Sufijo()` que, además de copiarlo a la salida, actualiza el valor de p a la posición inmediatamente posterior al último símbolo de la

[4]Cuando se llega a que todos los estados del transductor son el estado de absorción, todavía queda por procesar parte de la FL de entrada.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

entrada que forma parte del sufijo. Esto es posible debido a las FL están perfectamente delimitadas por la derecha.

Por ejemplo, la FL del castellano `postre<n><m><sg>/` se procesa completamente debido a que, por causa del cambio de género y número que se produce en la traducción al catalán, se especifica la FL completamente (`postres<n><f><pl>/`). En cambio, la FL `alto<adj><f><pl>/` se procesa de forma parcial hasta la categoría léxica `<adj>`, ya que la información sobre género y número es idéntica en ambas lenguas, como se puede observar en la tabla 7.1.

7.4 Generador morfológico

El objetivo de este módulo es realizar el proceso inverso al realizado por el analizador morfológico pero en la lengua meta. Es decir, obtener una *forma superficial* (FS) a partir de cada forma léxica de entrada. Para ello, al igual que los módulos anteriores, lee un transductor de letras, creado a partir de un diccionario morfológico (sección 2.3) y procesado como se describe en los capítulos 4 y 5, de un fichero binario, que contiene toda la información de un diccionario morfológico.

La entrada de este módulo consiste en una sucesión de formas léxicas, en este caso proporcionadas por el módulo de transferencia léxica. El funcionamiento de este módulo es una simplificación del algoritmo 7.1, ya que únicamente tiene un conjunto de salidas validadas (CSV), debido a que no necesita realizar preanálisis (sección 6.2.1), de forma que su funcionamiento consiste en procesar la FL de entrada completa y emitir la transducción correspondiente del CSV, o emitir un carácter de la entrada si todos los estados del conjunto de salidas parciales (CSP) son el estado de absorción y el CSV está vacío. Este módulo, al igual que el módulo de transferencia léxica, cuando detecta que el carácter a procesar es la marca de palabra desconocida, invoca una subrutina `Paldesco()` que lee la palabra de la entrada y la copia a la salida.

7.5 Postgenerador: aplicación de reglas ortográficas

La finalidad de este módulo es aplicar ciertas reglas ortográficas relativas a la lengua meta, especificadas en un diccionario de reglas. Un ejemplo de este tipo de reglas son las reglas de apostrofación de artículos, preposiciones,

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

etc. del catalán. Por ejemplo, la frase del castellano *las pruebas para el instituto*, se traduce al catalán como *les proves per el institut*, antes de ser tratada por un módulo para aplicar las reglas de contracción y apostrofación del catalán. Para obtener una traducción correcta se debe aplicar la regla de la contracción y apostrofación:

`per` + `el` + <*consonante*> → `pel` <*consonante*>
`per` + `el` + <*vocal*> = `per l'`<*vocal*>

de forma que la traducción final obtenida sería *les proves per l'institut.*

Este módulo también lee un transductor de letras de un fichero binario, que contiene toda la información sobre las reglas ortográficas a aplicar.

La entrada de este módulo consiste en formas superficiales de la lengua meta proporcionadas por el generador morfológico, con la particularidad de que algunas de las FS están marcadas con un símbolo especial para indicar que es posible que se les haya de aplicar alguna de las reglas ortográficas (esta marca se incluye en la entrada del diccionario morfológico de la lengua meta, de manera que forma parte de la FS correspondiente a una determinada FL). En el ejemplo anterior, las formas marcadas serían `per` y `el`.

El TLDA en el que se basa este módulo únicamente tiene transducciones fuertemente aceptadas, al igual que los dos módulos anteriores; es decir, para todos los estados q del transductor, $\xi_s(q) = \Sigma \cup \{\$\}$ para todo $\sigma \in \Sigma$ y su funcioncionamiento es básicamente igual que el del generador morfológico, salvo por el tipo de entrada.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Universitat d'Alacant
Universidad de Alicante

Capítulo 8

Módulo de transferencia estructural

Aunque las lenguas involucradas en el sistema de TA sean ambas muy similares en su sintaxis, a menudo es necesario realizar operaciones que requieren la identificación de estructuras sintácticas y su manipulación (bien transformando las palabras de la estructura identificada o, bien cambiando la estructura misma, o ambas cosas a la vez) como ya se comentó en la sección 1.1.5. Esta operación, que se realiza en la fase de transferencia, puede organizarse, cuando las lenguas no son muy diferentes, en torno a *patrones* que representan secuencias de formas léxicas de la lengua origen (FLLO) de longitud fija; una secuencia de FL sigue un cierto patrón cuando contiene la secuencia de categorías léxicas especificada en el patrón. El sistema contiene un catálogo de patrones que sabe como procesar. Los patrones no son "frases" ni constituyentes en un sentido sintáctico estricto, porque son sencillos y sin estructura, pero la detección de patrones es un avance respecto al mero análisis morfológico y puede ser considerado como una forma rudimentaria de análisis sintáctico, de ahí el nombre de *transferencia morfológica avanzada* que se usa para referirse a los sistemas de TA que trata esta tesis (sección 1.2).

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: se escoge el patrón más largo que coincida, se procesa la secuencia seleccionada (mirar más abajo), y el proceso continua 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} (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

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

se solapan. Por tanto, el procesamiento se realiza de izquierda a derecha y en fragmentos bien definidos, de forma similar al *chunking* (Abney 1991), que consiste en dividir un texto en segmentos de palabras que están relacionadas sintácticamente.

El procesamiento de patrones toma la secuencia de FLLO detectada y construye, utilizando el módulo de transferencia léxica (sección 7.3), una secuencia de formas léxicas en la lengua meta (FLLM) que puede haber sido reordenada, y de la que se pueden haber eliminado o a la que se pueden haber añadido FL. La información sobre la flexión de las FLLM es generada a partir de la concordancia observada dentro de la secuencia, si es necesario. Por ejemplo, el patrón castellano artículo–adjetivo–nombre (por ejemplo ser "una señal inequívoca") se convierte en una secuencia sin reordenar en catalán ("un senyal inequívoc"), después de propagar el género masculino en catalán (era femenino en castellano) del sustantivo "senyal" al artículo y al adjetivo. Además, el módulo de transferencia puede mantener información de "estado" para asegurar la propagación de información entre patrones como por ejemplo, la necesaria para asegurar la concordancia de número entre sujeto y verbo. Esta información de estado se actualiza después del procesamiento de cada patrón.

Naturalmente, un catálogo finito de secuencias "congeladas" de longitud fija no puede cubrir todas las posibles formas que un cierto constituyente (p.e. un sintagma nominal) pueda tomar, pero si los patrones son escogidos de manera que cubran los fenómenos más comunes, se puede obtener una calidad razonable de traducción, en particular cuando las lenguas origen y meta son sintácticamente similares. Este diseño —que favorece los patrones más largos— permite a los desarrolladores construir primero un sistema palabra por palabra (que es como funciona por defecto el módulo de transferencia salvo que un patrón sea encontrado) y añadir patrones incrementalmente con la confianza de que, si los patrones están bien escogidos, solo se pueden producir mejoras en la calidad de la traducción.

Otra limitación de este sistema que se debe mencionar es la dificultad de modelizar con él fenómenos complejos de larga distancia, a pesar de contar con una estructura para propagar información de un patrón a otro.

8.1 Especificación del módulo de transferencia

En lugar de programar el módulo de transferencia directamente en C o en un lenguaje similar, se ha diseñado un lenguaje de alto nivel (con palabras

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

clave en catalán) que puede ser usado por un lingüista, después de algún entrenamiento, para codificar:

- los patrones de FLLO que tienen que ser detectados para su procesamiento;
- la extracción de características gramaticales como el lema, género o número de las FLLO;
- la manipulación de FLLO y el montaje de sus traducciones (consultadas en un diccionario bilingüe), y de otras FLLM generadas internamente (si es necesario) para producir el patrón meta.

Este lenguaje también admite comentarios que tienen el mismo formato que los utilizados para los diccionarios (sección 2.3).

8.1.1 Estructura del programa

Los programas codificados en este lenguaje se componen de dos secciones bien diferenciadas: una primera sección donde se realizan todas las declaraciones y una segunda sección donde se encuentran los patrones que considera el módulo de transferencia y las acciones (transformaciones) asociadas a ellos, es decir, las *reglas* patrón–acción.

La sección de declaraciones: En esta sección se incluyen las definiciones de los elementos que forman parte de una secuencia de FLLO; es decir, las categorías léxicas que se usarán para clasificar las FLLO en patrones y el formato de los espacios en blanco que las separan. También se definen los atributos de las categorías léxicas y las variables necesarias para pasar información de un patrón a otro. Las distintas declaraciones de esta sección son las siguientes:

- La palabra clave `separa` y el símbolo `:=` seguidos de una expresión regular que define los espacios en blanco que pueden ser encontrados entre formas léxicas. Este material será usado para reconstruir el formato después de la generación. Un ejemplo sencillo de este tipo de declaración es

```
separa:="[ \n\t]+"
```

que declara como separador entre formas léxicas cualquier secuencia de uno o más espacios en blanco, retornos de carro o tabuladores.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

- Una o más *declaraciones de categorías léxicas* consistentes en la palabra clave `catlex`, un identificador, el símbolo `:=` y una expresión regular que seleccionará las formas léxicas que serán tratadas como una categoría particular. Cabe 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 plurales). Por ejemplo si se desea detectar las formas léxicas correspondientes a la categoría léxica *adjetivo* porque forma parte un patrón sobre el que se aplica una regla, la declaración sería

```
catlex adj:="<adj>(<ind>|<sup>|<itg>|<pos>)*"
```

  de forma que detectaría las formas léxicas de adjetivos con el formato utilizado en el sistema de TA interNOSTRUM (capítulo 9) de los tipos indefinido, superlativo, interrogativo, posesivo y aquellos que no tienen subcategoría léxica especificada. Ejemplos de FL compatibles con esta declaración serían `rojo<adj><m><sg>` o `rojo<adj><sup><m><sg>`.

- Una o más *declaraciones de atributos*, cada una de las cuales consiste en la palabra clave `atribut`, un identificador, el símbolo `:=` y una expresión regular que describe las subcadenas que representan un cierto atributo (como puede ser el *género* o el *número*) en una forma léxica. Por ejemplo, la declaración de los valores que puede tomar el atributo *número* de una categoría léxica, como la de *adjetivo*, sería

```
atribut num:={"<sg>","<pl>","<sp>","<ND>"}
```

  que indica que puede ser singular, plural, válido para singular y plural (como por ejemplo la palabra salvavidas del castellano) o que tiene el número sin determinar. Este último valor sólo aparece para FLLM del sistema interNOSTRUM (capítulo 9) cuando se traduce una palabra que en la lengua origen es válida para el singular y el plural, pero en la lengua meta tiene formas diferentes para cada valor (como es el caso de crisis en castellano, que al traducirla al catalán tiene dos posibles valores: *crisi* o *crisis*)[1].

- Una o más *declaraciones de variables de estado*, cada una de las cuales consiste en la palabra clave `estat` y un identificador. Las variables de

[1]Igualmente existen en interNOSTRUM los valores `<m>` (masculino), `<f>` (femenino), `<mf>` (válido para masculino y femenino) y `<GD>` (género sin determinar) para el atributo de género.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

estado son utilizadas para transferir valores de atributos activos (mirar la palabra clave `activa` más abajo) de un patrón a otros que le sigan o como variables de uso interno a una regla.

El lema de las formas léxicas está definido de forma implícita. Como parte del lema se considera todo aquel carácter alfanumérico o de signo de puntuación (".", ";", etc.) que se encuentre inmediatamente delante de las etiquetas; el lema se almacenará en el atributo implícito `lem` de cada forma léxica. Existe otro atributo implícito, `tot`, que contiene toda la forma léxica.[2]

La sección de reglas: Esta sección es una secuencia de reglas patrón–acción donde cada regla tiene tres partes:

- La definición del patrón que será detectado. Este consiste en la palabra clave `detecta` seguida de una secuencia de una o más categorías léxicas de las previamente declaradas con `catlex`. Cabe destacar que, si la secuencia de FLLO detectada en la entrada es reconocida por dos patrones diferentes, en primer lugar, prevalece, como ya se ha dicho, el más largo y, en segundo lugar, para patrones de la misma longitud, prevalece el primer patrón definido. Un ejemplo de patrón a detectar para aplicar sobre él acciones de concordancia y reordenamiento de elementos sería

  ```
  detecta cuyo nombre adj
  ```

  que detectaría la secuencia de FLLO del castellano formada por el relativo *cuyo* (en cualquiera de sus formas) seguida de un nombre y un adjetivo, y llamaría de forma implícita al módulo de transferencia léxica para obtener la secuencia de FLLM correspondiente en la lengua meta.

- La declaración de aquellos atributos que son relevantes para una regla particular y que por tanto han de ser activados, es decir, extraídos de las FLLO y FLLM (menos los atributos `lem` y `tot` que se activan de forma implícita). Los nombres de los atributos siguen a la palabra clave `activa`. Además de los atributos definidos con `atribut`, las categorías léxicas definidas con `catlex` también pueden ser extraídas de las FLLO y FLLM para ser manipuladas en las reglas cuando ello sea necesario. Para ello también se deberán activar con `activa`.

[2]Este atributo se utiliza cuando se quiere generar una forma léxica completa que no ha sido modificada.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

- Finalmente vienen las acciones a realizar para el patrón, encapsuladas entre llaves y escritas utilizando la sintaxis descrita en el siguiente apartado. Estas acciones pueden consistir, por ejemplo, para el patrón anterior, en establecer la nueva concordancia en la lengua meta, ya que en castellano el relativo posesivo *cuyo* concuerda con el nombre al que acompaña, mientras que en catalán, *el qual*, concuerda con su antecedente, para lo cual es necesario disponer de una variable de estado que proporcione esta información sobre el sintagma nominal anterior. Además se pueden generar FLLM nuevas (el determinante que acompaña al nombre) y se produce un reordenamiento de las FL: el patrón resultante en catalán tiene la forma `det nombre adj el_qual`.

8.1.2 El lenguaje de las acciones

Las acciones son secuencias de sentencias especificadas en un lenguaje de alto nivel que recuerda a C, y que está especificado por la siguiente gramática:

Stats	→	Stat ; Stats
Stats	→	Stat
Stat	→	{ Stats }
Stat	→	`si` (Cond) Stat
Stat	→	`si` (Cond) Stat `altrament` Stat
Stat	→	`envia` Expr
Stat	→	**$ num** `.orig.` **id** := Expr
Stat	→	**$ num** `.meta.` **id** := Expr
Stat	→	**id** := Expr
Stat	→	ε
Cond	→	Cond `o` Conj
Cond	→	Conj
Conj	→	Conj `i` SimCon
Conj	→	SimCon
SimCon	→	(Cond)
SimCon	→	`no` SimCon
SimCon	→	Expr **compop** Expr
Expr	→	Expr SimplExp
Expr	→	SimplExp
SimplExp	→	`net` SimplExp
SimplExp	→	**$ num** `.orig.` **id**
SimplExp	→	**$ num** `.meta.` **id**
SimplExp	→	**&** **num**
SimplExp	→	**string**
SimplExp	→	**id**

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

donde:

- La construcción `si (...)...altrament...` es la sentencia condicional equivalente a la de cualquier lenguaje de programación.
- El símbolo `$` seguido de un número positivo i (**num**) representa la i-ésima FLLO o FLLM en el patrón actual. Por ejemplo si se está tratando el patrón `nombre adj`, `$1` representa la FLLO (FLLM) del nombre y `$2` la del adjetivo.
- El símbolo `&` seguido de un número positivo i (**num**) representa el espacio en blanco encontrado entre la i-ésima y la $(i+1)$-ésima FLLO. En el ejemplo de patrón anterior únicamente se detecta un espacio en blanco, que se encuentra entre la forma léxica del `nombre` y la del `adjetivo`, y se representa como `&1`.
- El operador `:=` asigna el valor que devuelve la parte derecha a la parte izquierda.
- La orden `envia` manda una expresión a la salida estándar. Esta expresión está formada por concatenación de los diferentes elementos que pueden formar una FLLM. Por ejemplo, si una FLLM está formada por los elementos `$1.meta.lem` y `$1.meta.adj`, para generarla se utiliza la instrucción `envia $1.meta.lem $1.meta.adj`, que genera la concatenación de dichos elementos.
- El componente léxico **compop** representa uno de los operadores de comparación de cadenas `==` (igual) o `!=` (diferente).
- Los operadores `no`, `i` y `o` son los operadores lógicos *negación*, *conjunción* y *disyunción* respectivamente.
- El componente léxico **string** es una cadena entre comillas dobles. Por ejemplo, cuando en interNOSTRUM (capítulo 9) se desea comparar si género de una FL es masculino, se compara con la cadena `"<m>"`.
- El operador `.orig.` se utiliza para seleccionar la información en la forma léxica de la lengua origen. Por ejemplo, la construcción `$2.orig.gen` se referiría al atributo `gen` de la segunda FLLO en el patrón detectado.
- El operador `.meta.` se utiliza para seleccionar la información en la forma léxica de la lengua meta. Por tanto `$2.meta.gen` se referiría al atributo `gen` de la FLLM obtenida tras procesar la segunda FLLO del patrón a través del módulo de transferencia léxica (la consulta al diccionario bilingüe está implicita en todas las acciones).

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

- La concatenación de cadenas se realiza simplemente escribiendo las expresiones de cadena una al lado de otra, como se ha visto en la instrucción `envia`.

- Un identificador en una expresión de cadena representa el contenido de la variable de estado correspondiente. Por ejemplo, si en una instrucción `envia` se utiliza un identificador de una variable de estado, se está concatenando el contenido de dicha variable.

8.1.3 Un ejemplo

A continuación se muestra como se combinan todos estos elementos para formar un programa y su funcionamiento. Tómese, por ejemplo, la frase *Nos regalaron el postre, con cuyo sabor nos deleitamos.* del castellano. En la traducción al catalán la palabra *postre* cambia de género y número, por tanto se debe asegurar la concordancia del determinante que le precede. A continuación se encuentra el relativo *cuyo* que concuerda con el nombre al que precede, *sabor*, pero en su traducción al catalán, *de les quals*, debe concordar con su antecedente, *postre*. También aparece un nuevo elemento, un determinante, y se realiza un reordenamiento, obteniendo la traducción: *Ens van regalar les postres, amb el sabor de les quals ens vam delectar.*

A continuación se da como ejemplo el fichero que contiene las reglas que detectan los patrones `determinante nombre` y `preposición cuyo nombre`, que realizan estas acciones[3]:

```
# DECLARACIONES
# Espacios, retornos de carro y cualquier cosa entre [[...]] son
# espacios en blanco
separa:="([ \n\t]|\[\[([^\]]|[^\]])*]])+";

# Los nombres contienen la cadena "<n>" seguida opcionalmente por
# "<acr>" (indicador de acrónimos)
catlex nombre:="<n>(<acr>)?";

# Los determinantes pueden estar subcategorizados: <det><def>,
# <det><ind>, ...
catlex det:= "<det>(<def>|<ind>|<dem>|<pos>)";

catlex cuyo:="<rel><aa>";
catlex prep:="<pr>";
```

[3]Este fichero de reglas es válido para el sistema de TA interNOSTRUM (sección 9.3).

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

```
# Interesa extraer género y número, donde GD significa "género meta
# por determinar" y ND significa "número meta por determinar"
atribut gen:={"<m>","<f>","<mf>","<GD>"};
atribut nbr:={"<sg>","<pl>","<sp>","<ND>"};

# las variables genero y numero se utilizan para pasar información
# sobre el género y número distintos patrones. Las variables auxg
# y auxn se utilizan como variables temporales para guardar el
# género y número en la regla "prep cuyo nombre"
estat genero, numero, auxg, auxn;

#REGLAS
# Una regla simple para asegurar la concordancia de número y género
# entre determinante y nombre
detecta det nombre
{

# Extrae género, número y categorías
activa gen nbr det nombre;

# Hace algo solo si ambas partes concuerdan (esto es, forman una unidad
# en la LO)
si (  ( ( ($1.orig.gen!="<mf>")
          i($2.orig.gen!="<mf>")
          i($1.orig.gen==$2.orig.gen) )
        o( ($1.orig.gen=="<mf>")
          o($2.orig.gen=="<mf>") ) )
    i ( ( ($1.orig.nbr!="<sp>")
          i($2.orig.nbr!="<sp>")
          i($1.orig.nbr==$2.orig.nbr) )
        o( ($1.orig.nbr=="<sp>")
          o($2.orig.nbr=="<sp>") ) ) )

# Hace algo solo si el género o el número cambia en la traducción
# El género y número del determinante puede cambiar de <m>/<f> o
# <sg>/<pl> a <mf> o <sp>, respectivamente; por ejemplo, la
# traducción de ningún<det><ind><m><sg> es cap<det><ind><mf><sp>
si (  ($1.orig.gen!=$1.meta.gen)
    o ($1.orig.nbr!=$1.meta.nbr)
    o ($2.orig.gen!=$2.meta.gen)
    o ($2.orig.nbr!=$2.meta.nbr) )
```


Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

```
# Si el género del nombre está por determinar, lo toma del determinante,
# si éste tiene un género definido. Si no, toma el género masculino. Si
# ambos están por determinar, ambos adoptan el masculino.
si ($2.meta.gen=="<GD>")
 {
   si (($1.meta.gen!="<mf>") i
      ($1.meta.gen!="<GD>"))
        $2.meta.gen:=$1.meta.gen
   altrament
    {
        $2.meta.gen:="<m>";
        si ($1.meta.gen=="<GD>")
           $1.meta.gen:="<m>"
    }
 }
# Si no, se establece la concordancia por propagación de género a
# partir del nombre al determinante si ninguno es ambiguo.
altrament
 {
   si ($2.meta.gen!="<mf>")
    {
      si ($1.meta.gen!="<mf>")
        $1.meta.gen:=$2.meta.gen
    }
   altrament
     si ($1.meta.gen=="<GD>")
        $1.meta.gen:="<m>"
 };

# Ahora el número. Si el número en la LM está por determinar para
# el nombre pero está definido para el determinante, toma éste a
# partir de aquí; si no, adopta el singular.
si ($2.meta.nbr=="<ND>")
 {
    si ($1.meta.nbr!="<sp>")
      $2.meta.nbr:=$1.meta.nbr
    altrament
      $2.meta.nbr:="<sg>"
 }
# Si ambos están definidos, se transfiere el número del nombre al
# determinante
altrament
 {
```


Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

```
   si ($2.meta.nbr!="<sp>")
     si ($1.meta.nbr!="<sp>")
        $1.meta.nbr:=$2.meta.nbr
 };

# Se guarda el género y número del nombre detectado en esta regla
# para utilizarlo en una regla posterior
genero := $2.meta.gen;
numero := $2.meta.nbr;


# Escribe el patrón meta. Realmente no hacen falta dos
# instrucciones "envia"; se ha optado por esta solución por
# cuestiones de claridad de código
envia $1.meta.lem $1.meta.det $1.meta.gen $1.meta.nbr &1;
envia $2.meta.lem $2.meta.nombre $2.meta.gen $2.meta.nbr;
}

# Regla para asegurar la concordancia de número y género entre el
# relativo cuyo y su antecedente, y entre el nombre que sigue al
# relativo y el determinante que se genera
detecta prep cuyo nombre
{

# Extrae género, número y categorías
activa gen nbr prep cuyo nombre;

# Propaga el género del antecedente (regla anterior) al relativo
# cuyo
si (genero!="<mf>")
      $2.meta.gen:=genero
altrament
      $2.meta.gen:="<m>";

# Determina el género meta del nombre detectado en este patrón
si ($3.meta.gen=="<GD>")
      $3.meta.gen:=$2.orig.gen;

# Propaga el número del antecedente al relativo cuyo
si (numero!="<sp>")
      $2.meta.nbr:=numero
altrament
      $2.meta.nbr:="<sg>";
```


Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

```
# Determina el número meta del nombre detectado en este patrón
si ($3.meta.nbr=="<ND>")
      $3.meta.nbr:=$2.orig.nbr;

# Determina el género y número del determinante que se genera
# en esta regla
si ($3.meta.gen=="<mf>")
    auxg:=$2.orig.gen
altrament
    auxg:=$3.meta.gen;
si ($3.meta.nbr=="<sp>")
    auxn:=$2.orig.nbr
altrament
    auxn:=$3.meta.nbr;

# Escribe el patrón meta
envia $1.meta.lem $1.meta.prep  &1;
envia "el<det><def>" auxg auxn  &2;
envia $3.meta.lem $3.meta.nom $3.meta.gen $3.meta.nbr &2;
envia "el_qual"  $2.meta.cuyo $2.meta.gen $2.meta.nbr;
}
```

A continuación se realiza una breve descripción del ejemplo. En primer término está la sección de declaraciones donde se encuentra:

- La declaración del separador de palabras: se considera que cualquier secuencia de blancos, retornos de carro, tabuladores o información encapsulada entre `[[...]]` es un blanco en el sistema de TA interNOSTRUM (sección 9.4). Esta información es utilizada al final del proceso de traducción para devolver el formato al documento traducido.

- Las categorías léxicas que se utilizarán para detectar todos los patrones que contenga el fichero. En este caso: `nombre` (que puede ser o no un acrónimo), `det` (determinante definido, indefinido, demostrativo o posesivo), el relativo `cuyo` y `prep` (preposición).

- Los atributos que se necesitarán a lo largo de todo el fichero. Concretamente: género, que puede ser masculino, femenino, válido para masculino y femenino o por determinar y el atributo número, que puede ser singular, plural, válido para singular y plural o por determinar. Los valores por determinar ("`<GD>`" y "`<ND>`") únicamente aparecen en

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

las FLLM que proporciona el módulo de transferencia léxica. Estos valores los asigna cuando la FLLO a traducir tiene un único valor para el atributo correspondiente pero su FLLM puede optar a dos valores diferentes, como ya se ha dicho en la página 112.

- Las variables `genero`, `numero`, `auxg`, `auxn` que se utilizan para pasar información de un patrón a otro.

A continuación se encuentran las reglas para detectar los patrones:

- Regla `detecta det nombre`: En primer lugar realiza una llamada implícita al módulo de transferencia léxica para disponer de las FLLM correspondientes al patrón detectados. A continuación extrae de las FLLO y FLLM los atributos y categorías relativas a esta regla (`gen`, `nbr`, `det` y `nombre`). En el ejemplo *Nos regalaron el postre ...*, se detectaría la secuencia de FLLO correspondiente a *el postre*.

  La primera condición se utiliza para procesar solo aquellos patrones que formen un sintagma, es decir, que concuerden en género y número en la lengua origen. [4]

  La siguiente condición comprueba si se ha producido algún cambio en el género en la traducción del nombre o del determinante, ya que si no es así las subsiguientes comprobaciones son innecesarias. En el caso de *el postre* sí se produce un cambio de género y número en la traducción de *postre* del castellano (masculino, singular) al catalán *postres* (femenino, plural). En el siguiente paso se comprueba si el género del nombre (`$2`), que se considera como núcleo del patrón, ha quedado por definir en la traducción; si fuera así se consideraría el determinante (`$1`) como fuente para asignar el género al nombre, comprobando previamente que no esté por determinar (`<GD>`) o sea válido para masculino y femenino (`<mf>`), en cuyo caso se asigna el valor masculino como valor por defecto. En este ejemplo *postres* tiene género femenino, por tanto se pasa al `altrament` correspondiente. Dentro de éste se comprueba si el género del nombre y del determinante son `<mf>`; como no es así se le asigna al género del determinante el género del nombre.

[4] En el caso del patrón `det nombre` esto no sería necesario, si el desambiguador léxico no fallara algunas veces, ya que es imposible que en textos bien formados un determinante aparezca aislado; por otra parte, la comprobación de la concordancia es imprescindible en patrones como `nombre adj`, ya que son posibles frases como *leí libros cansada* en los que el nombre y el adjetivo no forman un sintagma y no tienen por qué concordar. Aún así, este filtro fallaría en frases como *las mujeres cosían almohadas cansadas*, que en catalán es *les dones cosien coixins cansades* y no *les dones cosien coixins cansats*.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Las comprobaciones que se realizan respecto al número son equivalentes, por tanto se asigna el número del nombre al número del determinante.

A continuación se almacena en las variables `genero` y `numero` el género y número resultantes del nombre para poder utilizarlos posteriormente. Por último, se realiza la composición del patrón de salida; para ello, en primer lugar se genera la traducción del determinante, seguido del espacio en blanco intermedio, y después la del nombre.

- Regla `detecta prep cuyo nombre`: En primer lugar se comprueba que el género almacenado en la variable que proporciona el género a la categoría `cuyo` no sea `<mf>` (en el caso del ejemplo es `<f>` ya que procede de la regla anterior), de forma que se asigna este género al atributo correspondiente de la FLLM `cuyo`. A continuación se comprueba si el `nombre` de este patrón tiene el género por definir como resultado de la traducción, en cuyo caso se le asignaría el género que tenía en la lengua origen el relativo *cuyo* que le precedía en la lengua origen.

  Las comprobaciones y operaciones realizadas respecto al número son equivalentes. A continuación se define el género y número que tendrá el determinante que acompaña a este nombre, y que se genera en esta regla. Por último, se realiza la composición del patrón. En primer lugar se genera la traducción de la preposición, a continuación se crea el nuevo determinante; para ello se concatena el lema y su categoría léxica en forma de cadena ("`el<det><def>`") al género y número determinados anteriormente a este efecto. Después se genera la traducción del nombre al que acompaña (con lo cual se está reordenando el patrón). Por último se genera la traducción elegida para `cuyo` en este contexto, "`el_qual`", y se le concatena el género y número correspondiente.

8.2 El compilador

El compilador `MorphTrans` que compila esta especificaciones para generar módulos de transferencia estructural ha sido desarrollado en Linux utilizando las herramientas clásicas de construcción de compiladores `yacc` (`bison`) y `lex` (`flex`). El compilador lee un fichero fuente `MorphTrans` (`.trf`) como los que se acaba de describir y escribe un fichero `lex` que es convertido a continuación en un fichero en C, compilado y enlazado con el módulo de transferencia léxica (sección 7.3) para producir un programa ejecutable. El programa en `lex` (Lesk 1975) está construido de forma que cada patrón se convierte en una expresión regular que es buscada en la entrada y guardada en una

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Volver al Indice/Tornar a l'index

cadena; como parte de la acción asociada, esta cadena se segmenta en formas léxicas y espacios en blanco; para ello se utiliza la librería `regex.h` de C, que permite extraer un segmento de una cadena que coincida con una expresión regular dada, utilizando la función `regexec`. La expresión regular que se le pasa como parámetro se obtiene transformando la cadena que contenga dicha expresión en un registro de tipo `regex_t` a través de la función `regcomp` de la misma librería. El paso siguiente es la traducción de cada forma léxica, ya almacenada de manera individual. El enlace con el módulo de transferencia léxica se hace a través de una función C con prototipo

```
char * Bilingue( char * ) ;
```

que toma una FLLO y proporciona una FLLM que es su equivalente designada para la lengua meta.

A continuación las formas léxicas, tanto las FLLO como las FLLM, son procesadas para extraer los atributos utilizando sus definiciones regulares de forma equivalente a como se han extraído las formas léxicas.

Por último, con los datos ya almacenados de forma separada para poder operar sobre ellos, se manipulan las formas léxicas de acuerdo con el código escrito por el lingüista, convenientemente traducido a C.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Universitat d'Alacant
Universidad de Alicante

Capítulo 9

Aplicación a un caso real: El sistema de traducción automática interNOSTRUM

El trabajo de esta tesis se enmarca en el proyecto "Desarrollo de un sistema de traducción automática del castellano al balear, catalán y valenciano", financiado por la Caja de Ahorros del Mediterráneo y la Universidad de Alicante. Este capítulo reproduce la descripción del sistema que se detalla en los artículos de Canals-Marote et al. (2001a) y Canals-Marote et al. (2001b).

Como resultado de este proyecto se ha desarrollado el sistema interNOSTRUM (disponible en línea en `http://internostrum.com/`), un sistema de traducción automática indirecta que utiliza la estrategia de transferencia morfológica avanzada descrita e implementada en esta tesis. El sistema interNOSTRUM consiste en ocho módulos (figura 9.1): un módulo desformateador (que separa el texto de la información de formato HTML o RTF), dos módulos de análisis (analizador morfológico y desambiguador léxico categorial), dos módulos de transferencia (módulo de transferencia léxica y módulo de transferencia estructural), dos módulos de generación (generador morfológico y postgenerador) y el módulo reformateador (que reintegra la información de formato original al texto traducido). La autora de esta tesis ha participado en el desarrollo de este sistema, asumiendo la realización de los siguientes programas o módulos:

- El analizador morfológico (sección 7.2).
- Los módulos de transferencia: el módulo de transferencia léxica (sección 7.3) y el módulo de transferencia estructural (capítulo 8).
- Los módulos de generación: el generador morfológico (sección 7.4) y el

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

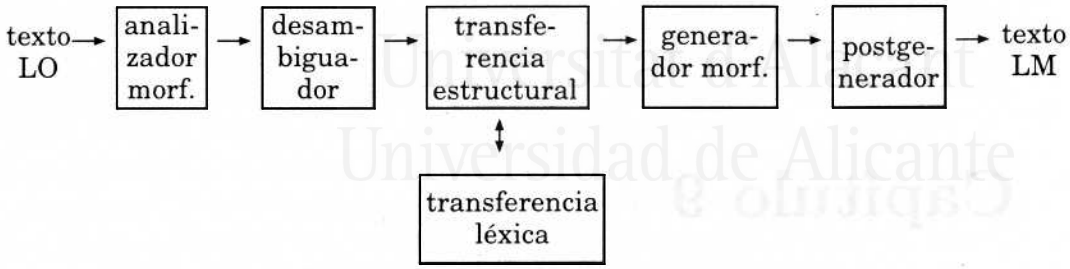


Figura 9.1: Los módulos lingüísticos de interNOSTRUM (los módulos desformateador y reformateador no aparecen en la figura).

postgenerador (sección 7.5).

- Los compiladores necesarios para generar los datos en un formato eficaz para su utilización en el sistema, a partir de los datos lingüísticos facilitados por los lingüistas:
  - Compilador de diccionarios (secciones 4.2 y 5.2).
  - Compilador `MorphTrans` (sección 8.2).
- Los prototipos de herramientas para el mantenimiento incremental de los diccionarios (capítulo 6).

Cabe destacar que una de las opciones en el diseño de interNOSTRUM es que todos los módulos se comunican con los demás a través de flujo de texto por dos razones: primero, para facilitar el diagnóstico de errores durante el desarrollo del sistema y segundo, porque es una elección natural cuando se utiliza un sistema operativo orientado a texto como es Linux (o, más genéricamente, Unix); esto causa una pequeña cantidad de reanálisis a la entrada de cada módulo, ya que la entrada y salida entre módulos consiste en secuencias de formas léxicas que se componen de lema (secuencia de caracteres) y características morfológicas (secuencia de etiquetas como las descritas en la página 28). Aun así, la velocidad obtenida actualmente en interNOSTRUM sobrepasa ampliamente las mil palabras por segundo en un PC típico de 1998 (Pentium II 400 MHz).

9.1 Módulos basados en tecnología de estados finitos

Cuatro de los módulos de interNOSTRUM, a saber, el analizador morfológico, el módulo de transferencia léxica, el generador morfológico y el postgenera-

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

Tabla 9.1: Generación automática de los módulos de interNOSTRUM a partir de datos lingüísticos

LENGUA	DATOS LINGÜÍSTICOS	PROGRAMA GENERADOR	MÓDULO DE interNOSTRUM
LO	diccionario morfológico (sección 2.3)	compilador de analizadores morfológicos (secciones 4.2 y 5.2)	analizador morfológico (sección 7.2)
LO	corpus analizado morfológicamente	entrenamiento del desambiguador	desambiguador
LO, LM	diccionario bilingüe (sección 2.3)	compilador de diccionarios bilingües (secciones 4.2 y 5.2)	módulo de transferencia léxica (sección 7.3)
LO, LM	reglas de procesamiento de patrones (sección 8.1.1)	compilador de reglas de procesamiento patrones (sección 8.2)	módulo de transferencia estructural (sección 8.1)
LM	diccionario morfológico (sección 2.3)	compilador de generadores morfológicos (secciones 4.2 y 5.2)	generador morfológico (sección 7.4)
LM	reglas de guionado y apostrofación (sección 2.3)	compilador de postgeneradores (secciones 4.2 y 5.2)	postgenerador (sección 7.5)

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

dor, están basados en los *transductores de estados finitos* (TEF) descritos en esta tesis (secciones 5.2 y 6.2.1). Esto permite que el procesamiento alcance velocidades del orden de 10 000 palabras por segundo, velocidad que es prácticamente independiente al tamaño de los diccionarios.

El analizador morfológico utiliza un transductor que se genera automáticamente (secciones 4.2 y 5.2) a partir del *diccionario morfológico* (DM) para la lengua origen (LO). El DM contiene los lemas (formas canónicas o de base de las palabras con flexión), los paradigmas de flexión y las relaciones entre ellos (sección 2.3). El módulo (sección 7.2) lee las formas *superficiales* (FS) y escribe, para cada una, una o más *formas léxicas* (FL) que consisten en un lema, una categoría léxica e información de flexión.

El módulo de transferencia léxica, el cual es invocado por el módulo de procesamiento de patrones (véase más abajo), utiliza un transductor que se genera automáticamente a partir de un fichero que contiene las correspondencias bilingües entre lemas. El módulo lee una FL en LO y escribe su FL equivalente en la lengua meta (LM) (sección 7.3).

El generador morfológico básicamente lleva a cabo la tarea inversa del analizador morfológico, pero referida a la LM. El transductor que lee el generador morfológico se genera a partir del DM de la LM (sección 7.4).

El postgenerador : Aquellas FS meta involucradas en el guionado y la apostrofación (tales como los pronombres proclíticos, los artículos, algunas preposiciones, etc.) activan este módulo que normalmente está *dormido* (copiando carácter a carácter la entrada en la salida). El transductor que lee el postgenerador se genera a partir de un fichero que contiene las reglas correspondientes para la LM (sección 7.5).

La división del texto origen en palabras presenta aspectos no triviales (en **interNOSTRUM** se usan conjuntamente la regla del segmento más largo y los procedimientos de preanálisis, tal como se explica en la sección 7.2). Por una parte, existen grupos de palabras que no pueden traducirse palabra por palabra, y deben ser tratados como *unidades multipalabra* (UMP); las UMP que tienen un número fijo de palabras se incorporan a los diccionarios morfológicos y al bilingüe; para ello, los módulos correspondientes son capaces de tratar tanto las UMP invariables como las que tienen flexión; su uso permite, además, evitar algunos de los problemas de traducción causados por la *homografía*, la *polisemia* o por estructuras no composicionales tales como

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

algunas locuciones y colocaciones (sección 2.1.2)[1]. Ejemplos de este tipo de construcciones son: Cast. *con cargo a* → Cat. *a càrrec de*; Cast. *por adelantado* → Cat. *per endavant*; Cast. **echar** *de menos* → Cat. **trobar** *a faltar*; Cast. **tener** *que* + infinitivo → Cat. **haver** *de* + infinitivo. En los dos últimos ejemplos (una construcción modal y una locución), la UMP tiene un elemento susceptible de flexión (indicado en negrita). En los diccionarios hay diferentes tipos de UMP en función de las FL que se generan en el análisis de las mismas y en función de si son invariables o tienen flexión. Ejemplos de los cuatro tipos básicos de UMP se pueden ver en la tabla 9.2.

Por otra parte, existen combinaciones de ciertas formas verbales y pronombres enclíticos que se escriben como una sola palabra en castellano; estas combinaciones presentan transformaciones ortográficas tales como cambios en la acentuación o pérdida de consonantes: Cast. *dámelo* = *da* + *me* + *lo* → Cat. *dóna* + *me* + *lo* = *dóna-me'l*; Cast. *presentémonos* = *presentemos* + *nos* → Cat. *presentem* + *nos* = *presentem-nos*. Algunos ejemplos de como se representan estas formas superficiales como formas léxicas se muestran en la tabla 9.3. El analizador morfológico se ocupa de resolver todos estos casos si los paradigmas correspondientes se introducen convenientemente en los diccionarios morfológicos.

9.2 El desambiguador léxico categorial

La mayoría de las ambigüedades léxicas están dentro de dos grandes grupos: la *homografía* (cuando una FS tiene más de una FL o análisis posible) y la *polisemia* (cuando la FS tiene una sola FL pero el lema puede tener más de una interpretación). El módulo de desambiguación léxica o *desambiguador léxico categorial*[2] (en inglés *part-of-speech tagger*) utiliza un modelo oculto de Markov basado en bigramas y trigramas (secuencias de dos o tres categorías léxicas) para resolver aquellos homógrafos que presentan ambigüedad categorial. Los parámetros del modelo reflejan las estadísticas de aparición conjunta de categorías observadas en un corpus de referencia; el desambiguador calcula en una pasada la desambiguación más probable de cada frase.

Una de las principales contribuciones a la actual tasa de error (alrededor del 3% en interNOSTRUM) son los pocos errores que aparecen en algunos homógrafos *difíciles* pero *frecuentes*, como puede ser *una* (artículo/verbo), *para* (verbo/preposición) y *como* (conjunción/verbo). Afortunadamente otros homógrafos frecuentes son más fáciles de desambiguar. Cuando se anali-

[1]El uso de la regla del segmento máslargo favorece a las UMP frente a las palabras simples.

[2]El programador de este módulo es Raül Canals Marote.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

	Una única FL	Secuencia de más de una FL
con flexión	real decreto → `real_decreto<n><m><sg>`	dar clase → `dar<vblex><inf>+clase<n><f><sg>`
	reales decretos → `real_decreto<n><m><pl>`	dar clases → `dar<vblex><inf>+clase<n><f><pl>`
	bien ajeno → `bien_ajeno<n><m><sg>`	doy clase → `dar<vblex><pri><1><sg>+clase<n><f><sg>`
	bienes ajenos → `bien_ajeno<n><m><pl>`	doy clases → `dar<vblex><pri><1><sg>+clase<n><f><pl>`
invariables	a rabiar → `a_rabiar<adv>`	así como → `así<adv>+como<pr>`
	como máximo → `como_máximo<adv>`	para después → `para<pr>+después<adv>`

Tabla 9.2: Ejemplos de los diferentes tipos de UMP que contiene el diccionario morfológico de castellano de interNOSTRUM. En las FL los caracteres '_' y '+' se consideran como un espacio en blanco de la entrada, pero el carácter '+' permite al módulo de transferencia léxica segmentar una secuencia de más de una FL unidas con '+' en las FL que la componen, con el fin de traducirlas de forma independiente.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

FS	FL
jugáosla	jugar<vblex><imp><2><pl> + os<prn><atn><2><mf><pl> +lo<prn><atn><3><f><sg>
tomándoselos	tomar<vblex><ger> + se<prn><atn><ref><3><mf><sp> +lo<prn><atn><3><m><pl>
traerles	traer<vblex><inf> + le<prn><atn><3><mf><pl>
fés-t'ho	fer<vblex><imp><2><sg> + et<prn><atn><2><mf><sg> +ho<prn><atn><3><nt>
crida'ns	cridar<vblex><imp><2><sg> + ens<prn><atn><1><mf><pl>

Tabla 9.3: Ejemplos de la representación de formas verbales con pronombres enclíticos en interNOSTRUM.

za el catalán se encuentran problemas similares. La polisemia no se trata (únicamente en algunos casos mediante el uso de unidades multipalabra de longitud fija que representan colocaciones seguras): el diccionario bilingüe siempre proporciona el mismo equivalente para cada lema; se da el caso de que los errores debidos a la polisemia son mucho menos frecuentes que aquellos que se deben a la homografía. El problema de la polisemia se puede evitar en aplicaciones concretas mediante el uso de un *lenguaje controlado* adaptado a los tipos de texto correspondientes. En el proyecto interNOSTRUM se está diseñando un castellano controlado para aplicaciones bancarias y administrativas.

9.3 El módulo de transferencia estructural

A pesar del gran parecido entre el castellano y el catalán, existen bastantes divergencias gramaticales: por ejemplo, existen divergencias de género y número que afectan a la concordancia — Cast. *la deuda contraída* (fem.) → Cat. *el deute contret* (masc.)—; construcciones de relativo con *cuyo*, ausente en catalán — Cast. *la cuenta cuyo titular es el asegurado* → Cat. *el compte el titular del qual és l'assegurat*[3] —, o cambios preposicionales — Cast. *en Londres* → Cat. *a Londres* —.

Estas divergencias se tratan utilizando las reglas gramaticales oportunas. Como se ha dicho más arriba, interNOSTRUM utiliza una solución similar a la de algunos sistemas de TA comerciales: se basa en la detección y tratamiento de secuencias predefinidas de categorías léxicas (*patrones* o *fragmentos* bien definidos) que pueden considerarse como sintagmas rudimentarios:

[3]Las reglas que realizan estas concordancias se pueden ver en la sección 8.1.3.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

por ejemplo, **art.–subst.** o **art.–subst.–adj.** son dos sintagmas nominales válidos. Las secuencias conocidas por el módulo de transferencia estructural (capítulo 8, Garrido-Alenda y Forcada 2001) constituyen su *catálogo de patrones.*

Este módulo se genera automáticamente a partir de un fichero que contiene las reglas que especifican los patrones y sus acciones asociadas. Es el más lento del sistema (velocidades del orden de mil palabras por segundo en un Pentium II de 400 MHz), comparado con las decenas de miles de palabras por segundo del resto de los módulos. Actualmente interNOSTRUM dispone de dos catálogos de reglas patrón–acción, uno para cada sentido de la traducción (castellano-catalán y catalán-castellano):

- El catálogo para la traducción del castellano al catalán contiene los siguientes patrones:
  - `preposicion infinitivo`: Esta regla cambia la preposición *en* por *a* cuando precede a un verbo en infinitivo. Por ejemplo, la traducción de *consiste en calcular* es *consisteix a calcular.*
  - `preposicion toponimo`: Esta regla cambia la preposición *en* por *a* cuando precede a nombre propio de lugar (por ejemplo, el nombre de una ciudad). Por ejemplo la frase *vive en Madrid* se traduce por *viu a Madrid.*
  - `determinante nombre`: Realiza las concordancias necesarias. Similares a este patrón hay cinco más:
    * `determinante nombre adjetivo`
    * `determinante nombre adverbio adjetivo`
    * `determinante adjetivo nombre`
    * `determinante adverbio nombre`
    * `determinante numeral nombre`
  - Los patrones `contraccion nombre`, `contraccion nombre adjetivo` y `contraccion adjetivo nombre` realizan funciones similares, en el caso de que el determinante sea el artículo de una contracción *del* o *al.*
  - `nombre adjetivo`: Realiza las concordancias necesarias. Similares a este patrón hay dos modalidades más: `nombre adverbio adjetivo` y `adjetivo nombre`.
  - `cuyo nombre`: Realiza las concordancias, generación de formas léxicas y reordenamientos necesarios de los sintagmas nominales iniciados por *cuyo.* Análogos a este patrón hay tres más:

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

`cuyo adjetivo nombre`, `cuyo nombre adjetivo` y `preposicion cuyo nombre`. Un ejemplo completo de la aplicación del patrón `preposicion cuyo nombre` se puede ver en la sección 8.1.3.

- `preterito`: A partir de la forma léxica correspondiente al pretérito indefinido de un verbo (por ejemplo *canté*) genera las dos formas léxicas del pretérito perifrástico del catalán (*vaig cantar*).

- El catálogo para la traducción de catalán a castellano contiene los siguientes patrones:
  - `preposicion infinitivo`: Esta regla cambia la preposición *en* por *al* cuando precede a un verbo en infinitivo que no sea modal (por ejemplo el verbo *haver de* es modal en catalán). Por ejemplo la frase *aniré en acabar* se traduce como *iré al acabar*.
  - `determinante nombre`: Realiza las concordancias necesarias. Similares a este patrón hay cuatro más:
    * `determinante nombre adjetivo`
    * `determinante nombre adverbio adjetivo`
    * `determinante adjetivo nombre`
    * `determinante adverbio nombre`
  - Los patrones `contraccion nombre`, `contraccion nombre adjetivo` y `contraccion adjetivo nombre` realizan funciones similares, en el caso de que el determinante sea el artículo de una contracción *del* o *al*.
  - `nombre adjetivo`: Realiza las concordancias necesarias. De este patrón hay dos modalidades más: `nombre adverbio adjetivo` y `adjetivo nombre`.
  - `auxiliar infinitivo`: Transforma las dos formas léxicas del pretérito perifrástico (por ejemplo *vaig entrar*) en la forma léxica única correspondiente al pretérito indefinido de un verbo (*entré*).

9.4 El desformateador y el reformateador

Ambos módulos[4] están escritos en `lex` y `C++`; el reformateador es mucho más sencillo. Hay tres versiones de cada módulo: la versión para textos ASCII, la versión para textos RTF y la versión para textos HTML. El desformateador se utiliza para identificar y separar los comandos de formato del texto que

[4]El programador de estos módulos es Sergio Ortiz Rojas.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Volver al Indice/Tornar a l'index

se quiere traducir. La información sobre formato, las imagenes incluídas en el documento, etc. se encapsulan (entre dobles corchetes "`[[...]]`") para formar los denominados *superblancos*, que los restantes módulos ven como simples espacios en blanco entre palabras. Por ejemplo,

```
<A HREF="http://www.ua.es/spv/">Servicio de promoción
del valenciano</A>
```

se encapsula como

```
[[<A HREF="http://www.ua.es/spv/">]]Servicio de promoción
del valenciano[[</A>]]
```

Los segmentos de material de formato muy grandes (> 8 kB), por ejemplo los que contienen gráficos, se escriben en ficheros temporales cuyo nombre único se envia entre corchetes en lugar de los datos. Una versión especial del desformateador convierte los URL de las etiquetas HTML "`<A HREF=...>`" en URL redirigidos a través de interNOSTRUM con el fin de permitir traducciones en tiempo real durante la navegación[5]. Por ejemplo, si al traducir con interNOSTRUM una página HTML de castellano al catalán, se encuentra la URL

```
<A HREF="http://www.elpais.es">
```

este módulo la transforma en

```
<A HREF="http://internostrum.com/tradurl.php?dir=es-ca&url=
http://www.elpais.es">
```

[5]También permite el tratamiento de muchos tipos de páginas con *frames*.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Universitat d'Alacant
Universidad de Alicante

Capítulo 10

Conclusiones

En esta tesis se ha descrito un sistema para generar traductores automáticos (completos a falta de un módulo de desambiguación léxica) que alcanzan velocidades de más de 1000 palabras por segundo en la traducción, sobre un Pentium II a 400 MHz, y que se pueden integrar fácilmente dentro de otros productos informáticos que en algún momento requieran de esta opción, especialmente aquellos relacionados con la comunicación a través de la red. De hecho, además de traducir textos en formato ASCII, RTF o HTML, interNOSTRUM está integrado en tres de estos servicios actualmente:

- Correo electrónico (`http://internostrum.com/mail.php`). Este servicio permite enviar un mensaje electrónico escrito en castellano o catalán traducido al catalán o castellano, respectivamente, con la posibilidad de revisar la traducción antes de enviarla.

- *Chat* (`http://internostrum.com/xat.php`). Este servicio permite mantener una conversación electrónica bilingüe; es decir, el usuario elige la lengua que utilizará, y las respuestas de sus contertulios le llegan en este idioma, sea cual sea la lengua (castellano o catalán) que utilicen sus contertulios.

- Navegación traducida (`http://internostrum.com/navegar.php`). Con este servicio se puede navegar a través de la red de forma que la traducción se hace automática e implícitamente (sección 9.4); por ejemplo, se puede leer un periódico electrónico en catalán como `vilaweb.com` en castellano.

También se describen métodos eficientes para mantener los diccionarios en los que se basa el sistema de traducción automática.

Estos traductores se pueden generar para cualquier par de lenguas sobre las que se tengan los datos lingüísticos en los formatos correspondientes, con

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

lo que se obtienen resultados aceptables (a nivel de asimilación y diseminación de información) para un par de lenguas que tengan una sintaxis similar, como pueden ser el italiano, el portugués o el catalán.

Como trabajo futuro se prevé realizar las siguientes tareas:

- Diseñar un desambiguador de categorías léxicas basado en reglas que se genere también a partir de una fuente de datos lingüísticos, de forma que se pueda obtener un sistema de traducción automática completo a partir de datos lingüísticos.
- Diseñar un sistema que permita alinear automáticamente las parejas FS–FL de los diccionarios morfológicos y las parejas FL–FL de los diccionarios bilingües, de manera que el transductor resultante sea el más pequeño posible. Por ejemplo en el diccionario bilingüe se tienen muchas traducciones que consisten en meras variantes ortográficas, sobre las que se puede aplicar un alineamiento automático basado en la distancia de edición de Levenshtein (Wagner y Fischer 1974).
- Con respecto al módulo de transferencia estructural:
  - Generalizarlo todo lo posible para que pueda ser utilizado en cualquier sistema de TA basado en la estrategia de transferencia morfológica. Para ello es necesario que todos los componentes de una forma léxica puedan ser definidos por el usuario; por tanto falta la posibilidad de poder definir el lema, la terminación de cualquier forma léxica y el formato de una forma léxica genérica en la sección de declaraciones.
  - Añadir operadores al lenguaje diseñado. Por ejemplo se ha pensado en incluir un operador que permita saber si un elemento pertenece a un conjunto de cadenas.
  - Mejorar la velocidad actual de este módulo.

10.1 Publicaciones relacionadas

A continuación se expone la lista de publicaciones que recogen de forma parcial el material descrito en esta tesis doctoral:

Canals, R., Garrido, A., Guardiola, M., Iturraspe, A., Montserrat, S., Pastor, H. y Forcada, M. (2000). Herramientas para la construcción de sistemas de traducción automática: aplicación al par castellano-catalán. IV Congreso de Lingüística General, Cádiz, España, 3-6.04.2000.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Canals-Marote, R., Esteve-Guillen, A., Garrido-Alenda, A., Guardiola-Savall, M., Iturraspe-Bellver, A., Montserrat-Buendia, S., Ortiz-Rojas, S., Pastor-Pina, H., Perez-Antón, P., y Forcada, M. (2001a). El sistema de traducción automática castellano-catalán interNOSTRUM. *Procesamiento del Lenguaje Natural*, 27:151–156. XVII Congreso de la Sociedad Española de Procesamiento del Lenguaje Natural, Jaén, Spain, 12-14.09.2001.

Canals-Marote, R., Esteve-Guillen, A., Garrido-Alenda, A., Guardiola-Savall, M., Iturraspe-Bellver, A., Montserrat-Buendia, S., Ortiz-Rojas, S., Pastor-Pina, H., Perez-Antón, P., y Forcada, M. (2001b). The Spanish-Catalan machine translation system interNOSTRUM. Proceedings of MT Summit VIII: Machine Translation in the Information Age p. 73–76. Santiago de Compostela, Spain, 18-22.09.2001.

Canals R., Esteve, A., Garrido, A., Guardiola, M.I., Iturraspe-Bellver, A., Montserrat, S., Pérez-Antón, P., Ortiz, S., Pastor, H., y Forcada, M. (2001) interNOSTRUM: a Spanish-Catalan Machine Translation System. *Machine Translation Review*, 11:21–25. En línea: `http://www.bcs.org.uk/siggroup/nalatran/mtreview/mtr-11/mtr-11-9.htm`.

Garrido-Alenda, A., Iturraspe-Bellver A., Montserrat S., Pastor-Pina, H., y Forcada, M. (1999). A compiler for morphological analysers and generators based on finite-state transducers. *Procesamiento del Lenguaje Natural*, 25:93–98. XV Congreso de la Sociedad Española de Procesamiento del Lenguaje Natural, Lleida, España, 8–10.09.1999.

Garrido, A., Iturraspe A., Montserrat S., Pastor, H. y Forcada, M. (2000). Generación automática de lematizadores de textos catalanes antiguos basados en técnicas de estados finitos. IV Congreso de Lingüística General, Cádiz, España, 3-6.04.2000.

Garrido-Alenda, A. y Forcada, M. L. (2001). 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.

Garrido-Alenda, A., Forcada, M. L., y Carrasco, R. C. (2002). Incremental construction and maintenance of morphological analysers based on augmented letter transducers. En *Proceedings of TMI 2002 (Theoretical and Methodological Issues in Machine Translation, Keihanna/Kyoto, Japan, March 2002)*, pages 53–62.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Volver al Indice/Tornar a l'index

Garrido-Alenda, A. y Forcada, M. L. (2002). Comparing nondeterministic and quasideterministic finite-state transducers built from morphological dictionaries. (enviado al XVIII Congreso de la Sociedad Española de Procesamiento del Lenguaje Natural).

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Universitat d'Alacant
Universidad de Alicante

Apéndice A

Estructura utilizada para guardar un TLDA

Cada estado del transductor se guarda en una estructura, `estado`, con los siguientes campos:

`numero` : entero que contiene el número de estado.

`nonulas` : puntero a la primera posición de un *array* de estructuras de tipo `trans` (estructura de las transiciones) que contiene las transiciones del tipo $(\sigma : \gamma)$ o $(\sigma : \theta)$ (página 43).

`numnonul` : entero que contiene el número de transiciones de este tipo que tiene el estado.

`nulas` : puntero a la primera posición de un *array* de estructuras de tipo `trans` que contiene las transiciones del tipo $(\theta : \gamma)$.

`numnul` : entero que contiene el número de transiciones de este tipo que tiene el estado.

El correspondiente código en C de esta estructura es:

```
typedef struct {
                int numero;
                trans *nonulas;
                trans *nulas;
                int numnul;
                int numnonul;
              } estado;
```

donde la estructura `trans` de las transiciones tiene los siguientes campos:

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

`lectura` : entero corto que contiene el símbolo que lee la transición.

`genera` : puntero a la primera posición de un *array* de estructuras de tipo `destinos` que contiene los símbolos de escritura y los estados destino correspondientes al símbolo de lectura.

`numgen` : entero que contiene el número de símbolos diferentes que se escriben al leer el símbolo `lectura` desde el estado actual.

El correspondiente código en C de esta estructura es:

```
typedef struct {
                    short int lectura;
                    destinos *genera;
                    int numgen;
                } trans;
```

donde la estructura `destinos` de cada transición tiene los siguientes campos:

`escribe` : entero corto que contiene el símbolo que escribe la transición.

`destino` : entero que contiene el estado destino de la transición.

El correspondiente código en C de esta estructura es:

```
typedef struct {
                     short int escribe;
                     int destino;
                } destinos;
```

El transductor se guarda en memoria como un *array* de estados, en el que cada estado apunta al primer elemento de cada uno de los dos *arrays* de transiciones (de lectura/escritura y de sólo escritura), que a su vez contienen un *array* de símbolos que se escribe la transición y su estado destino. Para ello el transductor se almacena en un puntero a la estructura `estado`, para el cual se reserva memoria dinámica en función del número de estados del transductor, de forma que se utiliza como un *array* de elementos del tipo `estado`. De igual manera se reserva memoria dinámica para los campos `nonulas` y `nulas` de cada estado, que serán utilizados como *arrays* de elementos del tipo `trans` dentro de cada estado. Por último, dentro de cada transición se reserva memoria dinámica para el número de símbolos que se escriben al leer un símbolo determinado (grado de indeterminismo del transductor) de forma que se utiliza como un *array* de elementos del tipo `destinos`.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Volver al Indice/Tornar a l'index

De esta forma el acceso a un estado del transductor viene dado por el número del estado (no hay que realizar ninguna búsqueda), y si desde un estado se pasa a otro sin leer ningún símbolo, tampoco hay que realizar búsquedas ya que consiste en recorrer el *array* de `destinos` de la transición `nulas` emitiendo los símbolos de escritura que tenga almacenados y cambiando al estado destino correspondiente.

Para la búsqueda del símbolo leído en el *array* de transiciones se realiza una búsqueda dicotómica en ejecución. Para realizar esta búsqueda se genera el *array* de transiciones en el orden alfabético de los símbolos leídos y se empieza buscando el símbolo por la mitad de la longitud del *array*, de forma que si el símbolo buscado es menor que el valor de esa posición, se busca en la mitad izquierda del *array*, en caso contrario se busca en la mitad derecha del mismo. Una optimización de esta operación es la búsqueda precompilada, de forma que se genera la estructura de datos para que simule un árbol ternario; el hijo izquierdo contiene el menor valor, el hijo derecho contiene el mayor valor, y el hijo medio contiene el valor que se busca (Bentley y Sedgewick 1997). Esta mejora no se ha llegado a implementar porque en los transductores obtenidos a partir de diccionarios reales, hay muy pocos estados que tengan más de dos transiciones.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

Apéndice B

Índice de símbolos utilizados

Σ	$\rightarrow$	alfabeto de entrada (página 34)
Γ	$\rightarrow$	alfabeto de salida (página 34)
τ	$\rightarrow$	transducción secuencial y p-subsecuencial (páginas 15, 17)
w	$\rightarrow$	una cadena (página 17)
p	$\rightarrow$	número máximo de sufijos de una transducción p-subsecuencial (página 17)
μ_0	$\rightarrow$	cadena de salida inicial (página 16)
σ	$\rightarrow$	$\sigma \in \Sigma$ (página 34)
γ	$\rightarrow$	$\gamma \in \Gamma$ (página 43)
$\zeta(w, \sigma)$	$\rightarrow$	extensión añadida a una cadena de salida al leer σ (página 16)
T	$\rightarrow$	transductor (páginas 34, 36, 42, 80)
q_I	$\rightarrow$	estado inicial de un transductor (páginas 33, 34, 35, 36, 43, 80)
δ	$\rightarrow$	función transición de un transductor (páginas 33, 34, 36, 43, 80)
Q	$\rightarrow$	conjunto finito de estados de un transductor (páginas 34, 36, 43, 80)
λ	$\rightarrow$	función de salida de un transductor secuencial (páginas 34, 35, 36)
F	$\rightarrow$	conjunto de estados de aceptación de un transductor (páginas 34, 35, 36, 43)
ψ	$\rightarrow$	función de salida que proporciona el conjunto de sufijos correspondiente a un estado de aceptación en un transductor p-subsecuencial (páginas 35, 37)

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Compiladores de datos linguisticos para la generacion de modulos de estados finitos para la traduccion... Alicia Garrido Alenda.

Volver al Indice/Tornar a l'index

L	$\rightarrow$	conjunto de etiquetas de las transducciones de un transductor de letras (páginas 43, 81)
ϵ	$\rightarrow$	cadena vacía
θ	$\rightarrow$	símbolo vacío
q	$\rightarrow$	un estado de un transductor
ξ_s	$\rightarrow$	función que asigna un conjunto de validación fuerte a un estado de un TLDA (página 81)
ξ_w	$\rightarrow$	función que asigna un conjunto de validación débil a un estado de un TLDA (página 81)
$\perp$	$\rightarrow$	estado absorción de un TLDA (página 82)
$\mathcal{L}_\sigma^s$	$\rightarrow$	lenguaje de las transducciones fuertemente aceptadas en un TLDA (página 83)
$\mathcal{L}_\sigma^w$	$\rightarrow$	lenguaje de las transducciones débilmente aceptadas en un TLDA (página 83)
$\mathcal{R}_\sigma^s$	$\rightarrow$	familia de los lenguajes derechos fuertemente aceptados por un estado de un TLDA (página 83)
$\mathcal{R}_\sigma^w$	$\rightarrow$	familia de los lenguajes derechos débilmente aceptados por un estado de un TLDA (página 83)
\$	$\rightarrow$	marca de fin de fichero (página 81)
$\mathcal{H}$	$\rightarrow$	lenguaje de L^+ que lleva a un estado en un TL (página 86)
S^t	$\rightarrow$	conjunto de validación (página 84)

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

Apéndice C

Índice de abreviaturas utilizadas

AFD	autómata finito determinista
AFN	autómata finito no determinista
FL	forma léxica
FLLO	forma léxica de la lengua origen
FLLM	forma léxica de la lengua meta
FS	forma superficial
RALM	representación abstracta en la lengua meta
RALO	representación abstracta en la lengua origen
TA	traducción automática
TEF	transductor de estados finitos
TL	transductor de letras
TLA	transductor de letras aumentado
TLD	transductor de letras determinista
TLDA	transductor de letras determinista aumentado
TLD-L	transductor de letras determinista respecto del conjunto de etiquetas de sus transiciones
TLCD-Σ	transductor de letras cuasi determinista respecto del alfabeto de entrada
TLI	transductor de letras indeterminista
UMP	unidad multipalabra

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Universitat d'Alacant
Universidad de Alicante

Bibliografía

Abney, S. (1991). Parsing by chunks. En Berwick, R., Abney, S., y Tenny, C., editores, *Principle-Based Parsing: Computation and Psycholinguistics*, p. 257–278. Kluwer Academic Publishers, Boston.

Aho, A. V., Sethi, R., y Ullman, J. D. (1986). *Compilers: Principles, Techniques and Tools*. Addison-Wesley, Reading, Massachusetts.

Allegranza, V., Krauwer, S., y Steiner, E. (1991). Introduction [Eurotra Special Issue]. *Machine Translation*, 6(2/3):61–71.

Allen, J. (1995). *Natural Language Understanding*. Benjamin/Cummings, Redwood City, CA, 2nd. edition.

Arnold, D. (1993). Sur la conception du transfert. En Buillon, P. y Clas, A., editores, *La traductique*, p. 64–76. Les Presses de l'Université de Montréal, Montréal.

Arnold, D., Balkan, L., Meijer, S., Humphreys, R., y Sadler, L. (1994). *Machine translation: An introductory guide*. NCC Blackwell, Oxford. Disponible en `http://clwww.essex.ac.uk/~doug/MTbook/`.

Bennett, W. S. y Slocum, J. (1985). The LRC Machine Translation System. *Computational Linguistics*, 11(2-3):111–121.

Bentley, J. y Sedgewick, B. (1997). Fast algorithms for sorting and searching strings. En *Proceedings of the 8th SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, p. 360–369.

Canals-Marote, R., Esteve-Guillen, A., Garrido-Alenda, A., Guardiola-Savall, M., Iturraspe-Bellver, A., Montserrat-Buendia, S., Ortiz-Rojas, S., Pastor-Pina, H., Perez-Antón, P., y Forcada, M. (2001a). El sistema de traducción automática castellano-catalán interNOSTRUM. *Procesamiento del Lenguaje Natural*, 27:151–156. XVII Congreso de la Sociedad Española de Procesamiento del Lenguaje Natural, Jaén, Spain, 12-14.09.2001.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Canals-Marote, R., Esteve-Guillen, A., Garrido-Alenda, A., Guardiola-Savall, M., Iturraspe-Bellver, A., Montserrat-Buendia, S., Ortiz-Rojas, S., Pastor-Pina, H., Perez-Antón, P., y Forcada, M. (2001b). The Spanish-Catalan machine translation system interNOSTRUM. En *Proceedings of MT Summit VIII: Machine Translation in the Information Age*, p. 73–76. Santiago de Compostela, Spain, 18–22.09.2001.

Carrasco, R. C. y Forcada, M. L. (2002). Incremental construction and maintenance of minimal finite-state automata. *Computational Linguistics*, 28(2):207–216.

Daciuk, J., Mihov, S., Watson, B. W., y Watson, R. E. (2000). Incremental construction of minimal acyclic finite-state automata. *Computational Linguistics*, 26(1):3–16.

Díaz-Ilarraza, A., Mayor, A., y Sarasola, K. (2000). Reusability of Wide-Coverage Linguistic Resources in the Construction of a Multilingual Machine Translation System. En *MT 2000: Machine Translation and Multilingual Applications in the New Millennium*, p. 12–1–12–9, Exeter, UK.

Díaz-Ilarraza, A., Mayor, A., y Sarasola, K. (2001). Inclusión del par castellano-euskara en un prototipo de traducción automática multilingüe. En *Proceedings of the Second International Workshop on Spanish Language Processing and Language Technologies (SLPLT-2)*, p. 107–111, Jaén.

Forcada, M. (2000). Learning machine translation strategies using commercial systems: discovering word-reordering rules. En *MT 2000: Machine Translation and Multilingual Applications in the New Millennium*, p. 7–1–7–8, Exeter, UK.

Garrido-Alenda, A. y Forcada, M. L. (2001). 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. XVII Congreso de la Sociedad Española de Procesamiento del Lenguaje Natural, Jaén, España, 12-14.09.2001.

Garrido-Alenda, A., Forcada, M. L., y Carrasco, R. C. (2002). Incremental construction and maintenance of morphological analysers based on augmented letter transducers. En *Proceedings of TMI 2002 (Theoretical and Methodological Issues in Machine Translation, Keihanna/Kyoto, Japan, March 2002)*, p. 53–62.

Hopcroft, J. E. y Ullman, J. D. (1979). *Introduction to automata theory, languages, and computation.* Addison-Wesley, Reading, MA.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Hutchins, J. (1999). The development and use of machine translation systems and computer-based translation tools. En *International Symposium on Machine Translation and Computer Language Information Processing*, Beijing, China.

Hutchins, W. y Somers, H. (1992). *An Introduction to Machine Translation.* Academic Press, London.

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

Maas, H. D. (1987). The MT system SUSY. En King, M., editor, *Machine Translation Today: The State of the Art*, p. 209–246, Edinburgh. Edinburgh University Press.

Mitamura, T., Nyberg, E., y Carbonell, J. (1991). An efficient interlingua translation system for multilingual document production. En *Proceedings of Machine Translation Summit III*, Washington, DC.

Mohri, M. (1994). Minimization of sequential transducers. En Crochemore, M. y Gusfield, D., editores, *Proceedings of the 5th Annual Symposium on Cmbinatorial Pattern Matching (CPM 94)*, volume 807, p. 151–163, Berlin-NY. Springer-Verlag. June 5-8 1994.

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Mori, T., Matsuo, M., y Nakagawa, H. (1999). Zero-subject resolution using linguistic constraints and defaults: The case of Japanese instruction manuals. *Machine Translation*, 14(3-4):231–245.

Nakaiwa, H. (1999). Automatic extraction of rules for anaphora resolution of Japanese zero pronouns in Japanese-English machine translation from aligned sentence pairs. *Machine Translation*, 14(3-4):247–279.

Nyberg, E. y Mitamura, T. (2000). The KANTOO machine translation environment. En *Proceedings of AMTA-2000*, p. 192–195.

Oncina, J., García, P., y Vidal, E. (1993). Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458.

Radford, A., Atkinson, M., Britain, D., Clahsen, H., y Spencer, A. (1999). *Linguistics: An introduction.* Cambridge Univ. Press, Cambridge.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002

Roche, E. y Schabes, Y. (1997). Introduction. En Roche, E. y Schabes, Y., editores, *Finite-State Language Processing*, p. 1–65. MIT Press, Cambridge, Mass.

Rosetta, M. T. (1994). *Compositional Translation.* Kluwer Academic Publisher, Dordrecht.

Sager, J. C. (1993). *Language engineering and translation: consequences of automation.* Benjamins, Amsterdam.

Schubert, K. (1988). The architecture of DLT - interlingua or double direct. En Maxwell, D., Schubert, K., y Witkam, T., editores, *New Directions in Machine Translation*, p. 131–144, Dordrecht, Holland. Foris Publication.

Slocum, J. (1987). METAL: the LRC machine translation system. En King, M., editor, *Machine Translation Today: The State of the Art*, p. 319–350, Edinburgh. Edinburgh University Press.

Tellier, I. (2000). Semantic-driven emergence of syntax: the principle of compositionality upside-down. En *Proc. 3rd Conference on the The Evolution of Language*, p. 220–224, Paris.

van de Snepscheut, J. L. A. (1993). *What computing is all about.* Springer-Verlag, New York.

Vandooren, F. (1993). Divergences de traduction et architectures de transfert. En Buillon, P. y Clas, A., editores, *La traductique.* Les Presses de l'Université de Montréal, Montréal.

Vauquois, B. y Boitet, C. (1985). Automated Translation at Grenoble University. *Computational Linguistics*, 11(1):28–36.

Wagner, R. A. y Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173.

Tesis doctoral de la Universidad de Alicante. Tesi doctoral de la Universitat d'Alacant. 2002