Ingeniería del Conocimiento

Procesamiento semántico en la arquitectura ARIES

Carlos A. Iglesias, José C. González, Amalio F. Nieto;
José M. Goñi Menoyo, Jesús López López
Deptos. de Ingeniería de Sistemas Telemáticos y de Matemática Aplicada a las Tecnologías de la Información, E.T.S. Ingenieros de Telecomunicación, Universidad Politécnica de Madrid.

Palabras clave: interpretación semántica, interpretación morfosintáctica, procesamiento de lenguaje natural

Resumen: *Este artículo afronta el problema de la conversión de estructuras morfosintácticas a una representación semántica basada en marcos y viceversa. Los algoritmos de interpretación presentados emplean el mismo conjunto de reglas para realizar dicha conversión en ambos sentidos (para análisis y generación) y se centran en recorrer el árbol morfosintáctico/semántico de forma descendente, aplicando las reglas de interpretación recursivamente a sus constituyentes.*

Abstract: *This article deals with the problem of converting morphosyntactic structures into semantic representations and vice versa. The algorithms implemented for interpretation use the same set of mapping rules to perform the conversion in both directions (for analysis and generation) and are based on applying the mapping rules recurrently while traversing the morphosyntactic/semantic tree in a top-down way.*

1. Introducción

En este artículo se presentan los módulos de procesamiento semántico de la arquitectura *ARIES*[1], **A**Rquitectura para **I**nterfaces en l**E**nguaje natural con modelado de u**S**uario. Dicha arquitectura se ha desarrollado por nuestro grupo en los últimos tres años, y está implementada en COMMON LISP. Un diagrama de bloques simplificado de esta arquitectura se muestra en la **figura 1**. En esta figura se pueden apreciar tres fases: análisis de la entrada del usuario, ejecución de alguna acción (sistema experto, BD,...) acorde con dicha entrada, y generación de una respuesta de este sistema al usuario. Como se puede observar, la columna de la izquierda representa la fase de análisis, la columna de la derecha la fase de generación, y en el centro, el sistema que procesa la salida de la fase de análisis y genera alguna respuesta al usuario.

En lo que sigue, se hace una sucinta descripción del sistema global y, más en particular, de los módulos de interpretación semántica y morfosintáctica.

Tanto la construcción de una representación del significado de una frase dada su estructura morfosintáctica en la fase de análisis (*interpretación semántica*), así como el proceso inverso en la fase de generación (*interpretación morfosintáctica*), constituyen fases importantes en el procesamiento del lenguaje natural (LN), siendo a menudo su cuello de botella. Los algoritmos de interpretación presentados procurarán simplificar al máximo el formato de las reglas y minimizar su número. Para ello se exigirá que la ontología sea robusta y se impondrán algunas restricciones a las estructuras morfosintácticas definidas por la gramática. Desde el punto de vista gramatical,se emplea la terminología clásica de la Gramática Léxico-Funcional [Kaplan y Bresnan, 82]. Para ilustrar el funcionamiento del sistema se em-plea a lo largo del texto una aplicación de 'juguete': un interfaz sencillo en lenguaje natural para el sistema operativo UNIX.

2. Descripción de la arquitectura

2.1. Subsistema de análisis

Analizador morfosintáctico: Un analizador recibe una entrada en LN, y como salida proporciona todas las estructuras morfosintácticas, denominadas *estructuras funcionales* o *f-estructuras*, que reflejan las diferentes ambigüedades morfosintácticas de la entrada.

Las fuentes de conocimiento con que se cuenta en este análisis son la *Gramática de Análisis*, que muestra qué construcciones son correctas morfosintácticamente, y *el Lexicón*, que permite identificar los rasgos morfológicos y sintácticos de las cadenas.

En este módulo hemos desarrollado una gramática de análisis en español [Nieto, 92] para el analizador universal de Tomita [Tomita, 88]. El tratamiento de la morfología del español, especialmente de la morfología verbal, se ha basado en el excelente trabajo realizado por Antonio Moreno [Moreno, 92].

Analizador semántico: Su misión principal será determinar las estructuras que, además de ser correctas sintácticamente, son coherentes.

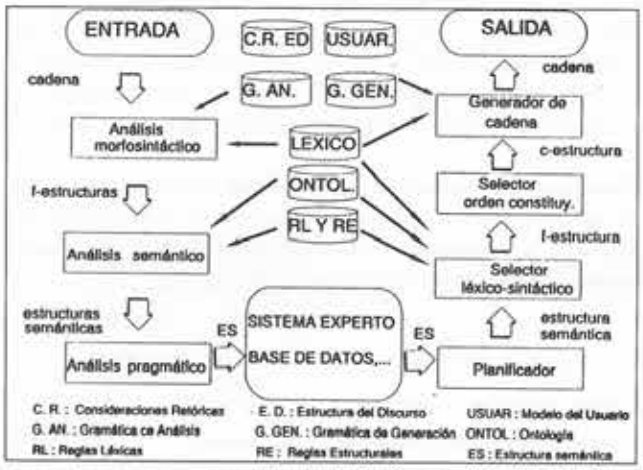

Figura 1: Arquitectura general para interfaces de lenguaje natural

La fuente básica de conocimiento de esta fase de análisis es *la Ontología*, modelo general del mundo que describe los tipos básicos de entidades y relaciones. El lenguaje de representación del conocimiento elegido ha sido el lenguaje basado en marcos *FrameKit* [Nyberg, 88].

Analizador pragmático: El objetivo de esta fase será seleccionar una única estructura semántica de las resultantes del análisis semántico conforme a criterios pragmáticos, así como resolver los referentes lingüísticos. Para ello, tendremos como fuente básica de conocimiento el modelo del usuario, que indicará si esperamos que el usuario responda a una pregunta, haga una afirmación, ... así como el modelo del discurso. En el caso del interfaz UNIX, bastará con realizar esta selección basándose en *expectativas* (según esperamos un mandato, afirmación,...).

2.2. Subsistema de generación

Planificador: Su misión será obtener una estructura del texto que satisfaga los objetivos de comunicación del sistema, y ofrezca un texto fluido. Con este objetivo, aplicará *estrategias y estructuras de discurso* (relaciones RST, esquemas, etc.).

Selector léxico-sintáctico: Su misión será seleccionar la estructura sintáctica óptima que más se adecúe a la estructura del discurso elegida, así como las palabras que más se ajusten a los conceptos, teniendo en cuenta el contexto y al usuario. En esta fase se debe pasar de nuevo del plano semántico al sintáctico. En nuestro sistema, se aplicarán las *reglas sintácticas y léxicas* de forma inversa, recorriendo toda la red semántica. Al proceso de convertir una red de marcos en una *f-estructura* le denominaremos **interpretación morfosintáctica**, por analogía con el proceso de interpretación semántica.

Selector del orden de los constituyentes: En las *f-estructuras* no está determinado el orden de los sintagmas, que se fija atendiendo a consideraciones estilísticas así como a los objetivos del hablante y del oyente. A la estructura funcional con un orden establecido se le denomina *estructura de los constituyentes* o ***c-estructura***. Tanto la selección de la estructura sintáctica como del orden de los componentes gramaticales se realiza en ARIES mediante la gramática de generación. Para seleccionar una estructura u otra con un orden determinado se utilizan rasgos auxiliares en la *f-estructura* que permitan discriminarlas. Por ejemplo, el rasgo *(FOCUS SI)* de un complemento circustancial permite anteponerlo al comienzo de la frase.

Generador de cadena: Su misión es transformar la c-estructura en una cadena en LN. Este generador se ha desarrollado con ayuda de la herramienta *GenKit* [Tomita y Nyberg, 88] basado en una *Gramática de Generación* [Iglesias, 93].

CATEGORIAS SINTACTICAS	CATEGORIA SEMANTICA
Verbo, Nombre, Adjetivo, Adverbio	MARCO
Conjunción, Preposición, Determinante	RANURA

Tabla 1

3. Módulo de interpretación semántica

La interpretación semántica consiste en la transformación de una *f-estructura* en una estructura semántica equivalente y coherente. Este proceso se realiza para cada *f-estructura* obtenida por el analizador morfosintáctico, obteniéndose una estructura semántica equivalente, o NIL, en caso de que no verifique alguna restricción semántica[2]. Los principios en que se basa la interpretación semántica en ARIES son:

- Utilización de **las mismas reglas** para las interpretaciones semántica y morfosintáctica. Esta es una de sus principales diferencias frente a otros sistemas como KBMT [Goodman y Nirenburg, 91] que emplea diferentes reglas con distinto formato para análisis y generación; o SIM [Hauptmann, 91] que se centra en la fase de análisis. El empleo de las mismas reglas -léxicas y estructurales- para análisis y generación evita tener que codificarlas dos veces.

- **Fuerte 'tipado' de las categorías sintácticas**. A cada categoría sintáctica se le hace corresponder un *tipo* semántico, según la **tabla 1**. Todas las categorías que se corresponden con un marco representan claramente conceptos, y deberán figurar en las *f-estructuras* como un par (ROOT <palabra>), que es fácil de generar en la gramática de análisis. La conjunción, la preposición y el determinante se asocian a ranuras de los marcos, y en la *f-estructura* estarán representados por un par (<categoría> <valor>): son modificadores de los conceptos anteriores.

- Tal como muestra el algoritmo de conversión, la interpretación semántica se realiza **recorriendo todo el árbol** de la *f-estructura* una sola vez de la raíz a las hojas.

- La frase se hace corresponder con **el verbo principal**. Las frases incomplentas sin verbo (v.g. *Sí, La vecina del primero, ¡vaya!*...) necesitarían un tratamiento especial, según sean respuestas a una pregunta o interjecciones.

- Las **restricciones semánticas** (conjunto de valores o clases de marcos permitidos) se imponen en la ontología, y no en las reglas de interpretación, como hace Allen [Allen, 87].

- **Generalización de las reglas**: podemos escribir una regla para una clase de marcos, y será aplicable a todos los marcos de dicha clase (vg. clase de los verbos transitivos).

El algoritmo de interpretación semántica se muestra en el **cuadro 1**. Como se observa, se aplican dos tipos de reglas: *léxicas y estructurales*.

- Las **reglas léxicas** harán corresponder las palabras que tienen como tipo semántico a un marco, con dicho marco.

- Las **reglas estructurales** harán corresponder rasgos morfosintácticos de las palabras con rasgos semánticos de los marcos.

Ilustraremos el algoritmo con un ejemplo: supongamos que queremos interpretar semánticamente una de las f-estructuras ambigüas de la frase: ***Borra el fichero $pepe.doc$ al mediodía***[3]. La *f-estructura* de la que partimos es:

```
((ROOT BORRAR)  (PERSONA 2) (NUMERO SINGULAR)
                (TIEMPO PRESENTE)
                (MODO IMPERATIVO) (VOZ ACTIVA)
        (SUJETO ((ROOT &ELIDIDO) (PERSONA 2)
                (NUMERO SINGULAR) (GENERO MASCULINO))
        (CD     ((ROOT FICHERO) (NUMERO SINGULAR)
                (GENERO MASCULINO) (DET DEFINIDO)
                (NOMFICH PEPE.DOC) (PERSONA 3)))
        (CI     NIL)
        (CC     ((ROOT MEDIODIA) (PREP A)
                (GENERO MASCULINO) (NUMERO SINGULAR)
                (PERSONA 3) (DET DEFINIDO)))
```

La red semántica a la que queremos llegar es:

```
(*BORRAR-FICH1
        (PERSONA (VALUE (COMMON 2)))
        (NUMERO (VALUE (COMMON SINGULAR))
        (TIEMPO (VALUE (COMMON PRESENTE))
        (MODO (VALUE (COMMON IMPERATIVO)))
        (VOZ (VALUE (COMMON ACTIVA)))
        (ACTOR (VALUE (COMMON *ELIDIDO2)))
        (FICHERO (VALUE (COMMON *FICHERO3)))
        (CUANDO (VALUE (COMMON *MEDIODIA4)))
)
(*ELIDIDO2
        (PERSONA (VALUE (COMMON 2)))
        (NUMERO (VALUE (COMMON SINGULAR)))
        (GENERO (VALUE (COMMON MASCULINO)))
)
(*FICHERO3
        (PERSONA (VALUE (COMMON 3)))
        (NUMERO (VALUE (COMMON SINGULAR)))
        (GENERO (VALUE (COMMON MASCULINO)))
        (NOMFICH (VALUE (COMMON PEPE.DOC)))
        (DET (VALUE (COMMON DEFINIDO)))
)
(*MEDIODIA4
        (PERSONA (VALUE (COMMON 3)))
        (NUMERO (VALUE (COMMON SINGULAR)))
        (GENERO (VALUE (COMMON MASCULINO)))
        (DET (VALUE (COMMON DEFINIDO)))
        (PREP (VALUE (COMMON A)))
)
```

1. Aplica reglas léxicas
2. Si no encontrado verbo, error
3. Crea marco del verbo (ejemplo del concepto verbo)
4. Aplica reglas estructurales
5. Repetir para cada regla:
- **Si el valor es múltiple, repetir para cada valor:**
  aplicar reglas estructurales
- **Si la raíz de la categoría es un marco**
  . Introducir, en la ranura de la categoría, la referencia del marco embebido
  . Crear el marco embebido y aplicar el algoritmo desde 4

Cuadro 1. Algoritmo de interpretación semántica

3.1 Reglas léxicas

El primer paso es aplicar las reglas léxicas. Las reglas léxicas sustituirán los pares (ROOT <palabra>) por (ROOT <ejemplo-de-concepto>).

El formato de las reglas léxicas es muy sencillo:
(<palabra> <(conceptos posibles)>).

La lista de conceptos posibles refleja la posible polisemia de la palabra. Como ejemplos de reglas léxicas podemos citar:

```
(FICHERO    (*FICHERO *MUEBLE))
(ARCHIVO    (*FICHERO))
(PRONOMBRE (*PRONOMBRE))
```

3.1.1. Tratamiento de la polisemia (ambigüedad léxica)

En ARIES se crea una f-estructura diferente para cada combinación de ambigüedades léxicas. Éstas se resuelven posteriormente aplicando *restricciones semánticas* en el proceso de interpretación. Para dominios con gran ambigüedad léxica esta solución no sería viable, pero sí lo es en el dominio de aplicación, que aborda la interpretación semántica en el dominio de las aplicaciones informáticas, con un lenguaje restringido y generalmente poco ambiguo[4].

En nuestro ejemplo, tras aplicar las reglas léxicas, obtenemos dos estructuras híbridas entre f-estructuras y estructuras semánticas. Ambas se corresponden con las dos posibles traducciones de la palabra *FICHERO* a los conceptos **FICHERO* y **MUEBLE* . Presentamos a continuación sólo la primera estructura, por brevedad. La segunda estructura es análoga sustituyendo *FICHERO3 por *MUEBLE4.

```
((ROOT *BORRAR-FICH1)
        (PERSONA2) (NUMERO SINGULAR) (TIEMPO PRESENTE)
        (MODO IMPERATIVO)
        (VOZ ACTIVA)
  (SUJETO ((ROOT *ELIDIDO2) (PERSONA 2)
                (NUMERO SINGULAR) (GENERO MASCULINO))
  (CD     ((ROOT *FICHERO3) (NUMERO SINGULAR) (GENERO
                MASCULINO) (DET DEFINIDO)
                (NOMFICH PEPE.DOC) (PERSONA 3)))
    (CI     NIL)
    (CC     ((ROOT *MEDIODIA5) (PREP A) (GENERO
                MASCULINO)
                (NUMERO SINGULAR) (PERSONA 3)
                (DET DEFINIDO)))
```

Las reglas léxicas empleadas han sido:

```
(BORRAR (*BORRAR-FICH))
(&ELIDIDO (*ELIDIDO))
(FICHERO (*FICHERO *MUEBLE))
(MEDIODIA (*MEDIODIA))
```

3.2. Reglas estructurales

Tras aplicar la reglas léxicas se aplican las reglas estructurales a todas las estructuras resultantes. Distinguiremos dos tipos:

generales y *particulares*. Ambos tipos de reglas estructurales harán corresponder rasgos sintácticos de una palabra con rasgos semánticos del concepto correspondiente a dicha palabra; pero mientras que las reglas generales establecen una correspondencia por defecto, las particulares establecen una correspondencia para un marco (o conjunto de marcos) determinado.

3.2.1. Reglas estructurales generales

El formato de las reglas estructurales generales es:

```
(<rasgo morfosintáctico><ranura semántica> [(<restricciones
morfosintácticas>)])
```

Las restricciones morfosintácticas son opcionales e indican qué rasgos morfosintácticos se exigen para que se pueda aplicar la regla. El formato de estas restricciones es:

```
(((rasgo 1 valor 1)      (rasgo 2 valor 2) ...     (rasgo n valor n))
((rasgo n+1 valor n+1)   ...                       (rasgo m valor m)))
```

Este formato es equivalente a:

```
(OR
        ((AND (rasgo 1 valor 1)  (rasgo 2 valor 2)... (rasgo n valor n))
        (AND (rasgo n+1 valor n+1)...                 (rasgo m valor m))))
```

Además se podrán incluir los valores **DEFINED** y **UNDEFINED** para comprobar si un rasgo tiene definido o no algún valor. Algunos ejemplos de estas reglas son:

```
(PERSONA PERSONA)
(NOMFICH NOMFICH)
(CD OBJETO)
(CC PROCEDENCIA (((PREP DE))((PREP DESDE))))
```

3.2.2. Reglas estructurales particulares

El formato de las reglas estructurales particulares es:

```
(<concepto>     ((<rasgo sintáctico 1> <ranura semántica 1>
                        [(<restricciones morfosintácticas rasgo 1>)])
                (<rasgo sintáctico 2> <ranura semántica 2>
                        [(<restricciones morfosintácticas rasgo 2>)])
                ...
                (<rasgo sintáctico n> <ranura semántica n>
                        [(<restricciones morfosintácticas rasgo n>)]))

        [(<restricciones morfosintácticas del verbo>)]
)
```

Estas reglas son aplicables si el marco que queremos transformar está relacionado mediante una relación ***is-a*** o ***instance-of*** con el primer argumento de la regla. Las restricciones morfosintácticas tienen el mismo formato que en las reglas léxicas. A continuación se muestran algunos ejemplos:

```
(*VERBOS-TRANSITIVOS-FICHEROS (CD FICHERO))
(*COPIAR-FICH (CD FICH_ORIGEN) (CC FICH_DESTINO))
```

En el caso de los verbos, se intentan verificar las reglas particulares en primer lugar. Si fallan, bien porque no se cumplan las restricciones semánticas expresadas en la ontología, o bien porque no sean aplicables, se ensayan las generales. Si fallan las reglas estructurales generales y particulares, se muestra un error, pues hay un rasgo sintáctico que no se puede interpretar.

Las reglas estructurales particulares se aplican ordenadas de menor a mayor grado de generalidad. Por ejemplo, primero se aplica la regla para *VERBO-TRANSITIVO-FICHEROS y después la de *VERBO-TRANSITIVO.

Veamos cuál es el resultado de aplicar las reglas estructurales en nuestro ejemplo. Como las dos estructuras son semejantes, desarrollaremos el ejemplo para la primera, y comentaremos qué sucede en la segunda. Tras encontrar el verbo, obtenemos la siguiente estructura:

```
(*BORRAR-FICH1
        (PERSONA 2) (NUMERO SINGULAR) (TIEMPO PRESENTE)
                (MODO IMPERATIVO) (VOZ ACTIVA)
        (SUJETO ((ROOT *ELIDIDO2) (PERSONA 2) (NUMERO
                SINGULAR) (GENERO MASCULINO) )
        (CD     ((ROOT *FICHERO3) (NUMERO SINGULAR)
                (GENERO MASCULINO) (DET DEFINIDO)
                (NOMFICH PEPE.DOC) (PERSONA 3) ) )
        (CI      NIL)
        (CC     ((ROOT *MEDIODIA5)(PREP A)(GENERO
                MASCULINO)(NUMERO SINGULAR)
                (PERSONA 3) (DET DEFINIDO) ) )
```

Ahora debemos aplicar las reglas estructurales a cada ranura sintáctica, para ir construyendo la estructura semántica. En nuestro sistema el valor de una ranura sintáctica puede ser bien atómico, bien una *f-estructura embebida*:

```
(<categoría sintáctica> <valor>)
(<categoría sintáctica> ((ROOT <concepto>)....)))
```

En el segundo tipo de ranuras sintácticas puede haber más de una sublista (ROOT <concepto>) en niveles más internos (*f-estructuras embebidas*).

La aplicación de las reglas estructurales da lugar a ranuras semánticas en el marco destino o a marcos embebidos, según las ranuras sintácticas sean del primer o segundo tipo, respectivamente.

Así, tras aplicar las reglas estructurales a

```
(PERSONA 2) (NUMERO SINGULAR) (TIEMPO PRESENTE) (MODO
IMPERATIVO) (VOZ ACTIVA)
```

tenemos el siguiente marco:

```
(*BORRAR-FICH6
        (INSTANCE-OF (VALUE (COMMON *BORRAR-FICH)))
        (PERSONA (VALUE (COMMON 2)))
        (NUMERO (VALUE (COMMON SINGULAR)))
        (TIEMPO (VALUE (COMMON PRESENTE)))
        (MODO (VALUE (COMMON IMPERATIVO)))
        (VOZ (VALUE (COMMON ACTIVA)))
)
```

Apliquemos ahora las reglas a la siguiente ranura sintáctica:

(CD ((ROOT *FICHERO3) (NUMERO SINGULAR) (GENERO MASCULINO)(DET DEFINIDO)(NOMFICH PEPE.DOC)(PERSONA 3)))

Como tiene embebido un par (ROOT <concepto>), aplicamos otra vez las reglas estructurales. Si fuera el concepto de tipo **ACCION* (es decir, un verbo), aplicaríamos las generales y las particulares. Al ser un **NOMINAL*, basta con aplicar las particulares. Este nominal podría tener embebido otro concepto (por ejemplo un adjetivo o un verbo) al que se volverían a aplicar las reglas estructurales.

Encontramos el par (ROOT *FICHERO3); se crea un marco, **FICHERO7*, al que se aplican las reglas estructurales.

Tras aplicar la regla (*VERBOS-TRANSITIVOS-FICHEROS (CD FICHERO)), el marco **BORRAR-FICH6* queda:

```
(*BORRAR-FICH6
        ((INSTANCE-OF (VALUE (COMMON *BORRAR-FICH)))
        (PERSONA (VALUE (COMMON 2)))
        (NUMERO (VALUE (COMMON SINGULAR)))
        (TIEMPO (VALUE (COMMON PRESENTE)))
        (MODO (VALUE (COMMON IMPERATIVO)))
        (VOZ (VALUE (COMMON ACTIVA)))
        (FICHERO (VALUE (COMMON *FICHERO7)))
)
```

La ranura sintáctica (CI NIL) no se traduce por tener valor *NIL*. Las ranuras *SUJETO* y *CC* se traducen de forma similar a la *CD*, y resulta la red semántica que nos habíamos trazado como objetivo.

Para la segunda estructura, que contenía **MUEBLE* en la ranura *CD*, la regla estructural (*VERBOS-TRANSITIVOS-FICHEROS (CD FICHERO)) falla al aplicarla a la sublista mostrada a continuación, pues el concepto **BORRAR-FICH* tiene definida una restricción para la ranura *FICHERO*, y sólo puede contener objetos del tipo **FICHERO*.

(CD((ROOT *MUEBLE4)(NUMERO SINGULAR)(GENERO MASCULINO) (DET DEFINIDO)(NOMFICH PEPE.DOC)(PERSONA 3)))

Por tanto, la interpretación semántica da como resultado la estructura semántica obtenida de interpretar la primera f-estructura.

En el caso de frases como ***Copia el fichero $pepe.doc$ en el directorio $util$ al mediodía***, habrá dos complementos circunstanciales. Las reglas estructurales en el caso del primero (*en el directorio $util$*) se aplican con éxito, y se hace corresponder *CC* con la ranura *FICH_DESTINO* mediante la regla estructural: (*COPIAR-FICH (CD FICH_ORIGEN) (CC FICH_DESTINO))

Cuando se aplica esta regla al complemento circustancial *al mediodía* falla, pues no es del tipo **FICHERO*. Entonces se aplican las reglas estructurales generales, y se aplica con éxito la regla: (CC CUANDO).

4. Módulo de interpretación morfosintáctica

La interpretación morfosintáctica pretenderá obtener una *f-estructura* a partir de una estructura semántica. En ARIES se realiza aplicando las mismas reglas léxicas y estructurales definidas en la interpretación semántica, pero leyéndolas de forma inversa. Se fundamenta en los mismos principios comentados en la interpretación semántica, variando lógicamente el algoritmo, que se muestra en el **cuadro 2.** Es la operación inversa de la interpretación semántica, de forma que si aplicamos consecutivamente la interpretación semántica y morfosintáctica a una *f-estructura*, obtendremos la misma *f-estructura* (salvo algunos rasgos debido a los fenómenos de sinonimia y polisemia).

La estructura semántica que se va a convertir debe contener todos los rasgos morfosintácticos de la *f-estructura* que se quiere generar. Una vez concluida la selección sintáctica se inicia el proceso de interpretación morfosintáctica, dentro del cual se realiza la elección léxica.

A continuación se muestran un conjunto de reglas léxicas y las reglas léxicas inversas correspondientes:

4.1 Reglas estructurales inversas

Son las mismas reglas estructurales empleadas en el proceso de interpretación semántica.

4.2. Reglas léxicas inversas

Se obtienen a partir de las reglas léxicas empleadas en el módulo de interpretación semántica. Tienen el siguiente formato: (<concepto> <palabras sinónimas>)

;; Ejemplo de reglas léxicas y reglas léxicas inversas correspondientes

```
;;Reglas léxicas                    Reg.léxicas inversas equivalentes
(FICHERO (*FICHERO *MUEBLE))(*FICHERO (FICHERO ARCHIVO))
(ARCHIVO (*FICHERO))               (*MUEBLE (FICHERO))
(ELIDIDO (*ELIDIDO))               (*ELIDIDO (ELIDIDO))
```

4.2.1. Tratamiento de la sinonimia (elección léxica)

Cuando hay varias palabras que son sinóminas es necesario realizar una elección léxica. En ARIES la elección léxica se realiza escogiendo la *palabra empleada menos recientemente*. Así, en el ejemplo, la única vez que hay que realizar una elección léxica es en el caso del concepto **FICHERO*, donde una vez se seleccionará *FICHERO* y la vez siguiente *ARCHIVO*.

1. La f-estructura es NIL
2. Añadir a la f-estructura el par (ROOT<marco>)
3. Aplicar reglas estructurales inversas (r.e.i.) al marco
- Se añaden a la f-estructura las listas(<ranura-según-r.e.i.><valor>)
- Si el valor es una marco, recursivamente se añade como valor la *f-estructura* resultante de aplicar el algoritmo desde 1

4. Aplicar reglas léxicas inversas

Cuadro 2: Algoritmo de interpretación morfosintáctica

5. Conclusiones

El formalismo de interpretación presentado aborda el problema de transitar entre los niveles morfosintáctico y semántico, siendo prometedores los resultados obtenidos en dominios restringidos. Frente a otros enfoques como KBMT, que emplean dos tipos de reglas léxicas, según el marco esté embebido o no, así como reglas estructurales diferentes para análisis y generación, ARIES emplea las mismas reglas léxicas para todos los marcos y las **mismas reglas** estructurales en ambos sentidos.

La filosofía general de diseño trata de simplificar la interpretación semántica y morfosintáctica a costa de robustecer la ontología mediante un mecanismo que mantenga su consistencia, así como restringir el tipo de f-estructuras permitidas en la gramática.

Entre las mejoras previstas en los módulos de interpretación podemos citar:

- El tratamiento de la ambigüedad léxica, manteniendo valores disyuntivos en los marcos (*empaquetamiento local de la ambigüedad*), que se resolverá a medida que se vayan aplicando restricciones semánticas.

- El tratamiento de casos particulares que no siguen el *principio de composicionalidad*, como determinadas frases idiomáticas (v.g. *Tiene muchos pájaros en la cabeza*).

- El tratamiento de frases sin verbo.

- Desarrollo de un subsistema de aprendizaje de reglas de interpretación de modo similar al de Hauptmann [Hauptmann, 91] y su posterior integración en la arquitectura ARIES.

El subsistema de interpretación se integrará con otras líneas actuales de investigación de nuestro grupo en el marco de la arquitectura ARIES. Entre éstas, destacamos el desarrollo de un léxico y de una gramática más completos, la definición de una ontología genérica para aplicaciones informáticas, y la realización de una componente de modelado de usuario.

Referencias

[Allen, 87] James Allen, *Natural Language Understanding*. Ed. Benjamin/Cummings Publishing Conpany, Inc, 1987.

[Goodman y Nirenburg, 91] Kenneth Goodman, Sergei Nirenburg, *The KBMT Project: A Case Study in Knowledge-Based Machine Translation*. Ed. Morgan Kaufmann Publishers, 1991.

[Hauptmann, 91] Alexander G. Hauptmann, *Meaning from Structure in Natural Language Processing*. Ph.D. Thesis. Carnegie Mellon University, 1991.

[Hirst, 88] G. Hirst, *Semantic Interpretation and Ambiguity*. Artificial Intelligence, Marzo 1988.

[Iglesias, 93] Carlos Angel Iglesias Fernández, *Prototipo de un Generador de Lenguaje Natural*, Proyecto Fin de Carrera, E.T.S.I. de Telecomunicación de U.P.M., 1993.

[Kaplan y Bresnan, 82] Ronald Kaplan, Joan Bresnan. *Lexical Functional Grammar: A formal system for grammatical representation*. En *The Mental Representation of Grammatical Relations*. J.Bresnan, ed. MIT Press, 1982.

[Moreno, 92] Antonio Moreno Sandoval. *Un modelo computacional basado en la unificación para el análisis y generación de la morfología del español*. Tesis doctoral. Dpto de Lingüística, Lenguas Modernas y Filosofía de la Ciencia. Facultad de Filosofía y Letras. Univ. Autónoma de Madrid.

[Nieto, 92] Amalio Fco. Nieto Serrano. *Reconocedor universal de Tomita: adaptación y desarrollo de una gramática para el español.*, Proyecto Fin de Carrera, E.T.S.I. de Telecomunicación de U.P.M., 1992.

[Nyberg, 88] Eric H. Nyberg. *The FrameKit User's Guide*. Carnegie Mellon University, Mayo 1988.

[Tomita, 88] Masaru Tomita et al. *The Generalized LR Parser Compiler. Version 8.1: User's Guide*. Carnegie Mellon, 1988.

[Tomita y Nyberg, 88] Masaru Tomita, Eric H. Nyberg. *Generation Kit and Transformation Kit. Version 3.2. User's Manual*. Carnegie Mellon, 1988.

Notas

[1] Este trabajo ha sido financiado parcialmente por el Plan Nacional I+D, con cargo al proyecto *Arquitectura para Interfaces en Lenguaje Natural con Modelado de Usuario* (TIC91-0217C02-01).

[2] Debido a que de las estructuras semánticas obtenidas en este proceso, el analizador pragmático selecciona la que satisfaga determinados criterios *pragmáticos* o *expectativas*, no es necesario transformar todas las f-estructuras, si no que puede finalizarse el proceso cuando una estructura semántica cumpla estas expectativas.

[3] En nuestra aplicación, aquellas palabras del *léxico dinámico* -no contenidas en el lexicón-, tales como los nombres de los ficheros, se delimitarán mediante un carácter especial (en los ejemplos *$*).

[4] Estamos estudiando empaquetar localmente la ambigüedad en ver-siones futuras, manteniendo valores disyuntivos en los marcos, que se vayan resolviendo a medida que se aplican las restricciones, de forma semejante al proceso de las *palabras polaroid* de Hirst [Hirst, 87].