TOSCA Y EL OBJETO PREPOSICIONAL

Jos Hallebeek

Universidad de Nimega
Países Bajos

Resumen

Este artículo se compone de dos partes. En la primera hay una introducción a las actividades del TOSCA Research Group for Corpus Linguistics de la universidad de Nimega (Países Bajos). Se explica muy brevemente cuáles son los objetivos, métodos e instrumentos de la disciplina de Corpus Linguistics (la lingüística de corpus). Con este fin se pasa revista a los principales proyectos realizados en los últimos veinte años. Recibe atención especial el proyecto dedicado al español. Los productos generados por este proyecto son: un corpus en soporte magnético, una gramática formal y un léxico computacional. Todos del español peninsular contemporáneo.

La segunda parte está dedicada a un tema de la gramática española: el objeto preposicional. Se explica cómo está descrito e incorporado en la gramática formal del español. Primero se formula una definición de la función llamada objeto preposicional o suplemento (en terminología de Alarcos Llorach). ¿Cómo se reconoce y en qué aspectos se distingue del complemento adverbial (o circunstancial)? Sigue luego una presentación de las reglas formales empleadas para la introducción de esta función en la gramática con un ejemplo de resultado de análisis automático.

1. TOSCA Research Group for Corpus Linguistics

El objetivo de Corpus *Linguistics* (la lingüística de corpus) apenas se difiere de otras muchas disciplinas de la lingüística. Estudiando el uso de lenguas individuales quiere conseguir un conocimiento más profundo del uso de la lengua en general. Y esto se hace analizando textos seguidos del lenguaje escrito o del lenguaje hablado reunidos en un corpus. Como se trata de texto seguido no sólo se estudia la estructura de frases sueltas sino también la estructura de textos. La lingüística de corpus se distingue de otras disciplinas lingüísticas porque quiere ser una disciplina auxiliar para otras. Creando bancos de datos con textos analizados y provistos de información sintáctica y morfológica, eventualmente semántica y fonológica, pone a disposición de los lingüistas una fuente de datos con una gran variedad de materiales. El manejo de datos históricos y auténticos se hace así tan fácil como la introspección, incluso para el lingüista teórico. La lingüística de corpus en la última década ha tomado un alto vuelo en los Países Bajos concretamente en la investigación del inglés, que tiene una gran tradición con nombres de gramáticos como Poutsma y Kruisinga. Sería impensable esta disciplina sin el apoyo y la ayuda de instrumentos informáticos. El ordenador concede rapidez y da garantías de homogeniedad en la manipulación del material que se analiza. Resumiendo, se puede afirmar que la lingüística de Corpus es la disciplina lingüística que se ocupa del análisis automático o semiautomático de corpus que estén en machine readable form. También se dedica a la explotación de los resultados del análisis. El investigador compone una gramática formal del aspecto de la lengua que quiere estudiar. Como el corpus es una colección de datos históricos, se estudia el uso concreto de la lengua, es decir con todas sus variedades del uso: diferentes registros, medios y estilos.

Para dar una idea de los instrumentos y de los datos con que trabajamos sigue un resumen de los principales proyectos realizados en nuestro grupo de investigación de Nimega.

1.1. Computer Corpus Pilot Project (CCPP)

Las actividades del Grupo de Investigación para lingüística de Corpus TOSCA perteneciente a la facultad de letras de la Universidad de Nimega (Países Bajos) se iniciaron a principios de los años 70. Se compuso entonces un corpus del inglés contemporáneo de unas 130.000 palabras y en machine readable form. A raíz de la composición de este corpus se estableció un proyecto de colaboración entre los cinco departamentos de inglés de las universidades holandesas bajo el nombre Computer Corpus Pilot Project (CCPP) (ver Aarts & Van den Heuvel 1980). Su objetivo era llevar a cabo una descripción formal de los datos inluidos en el corpus, concretamente en sus aspectos morfosintácticos. La lematización (tagging) del material se hizo a mano, adjudicando a cada vocablo un signo con indicación de su categoría morfológica. Además de estos se introdujeron otros signos para señalar los límites de los constituyentes sintagmáticos (sintagma nominal, verbal, etc.) Así se llegó al establecimiento de la estructura constitutiva de las oraciones contenidas en el corpus. Una vez terminada la lematización manual se hizo el análisis sintáctico de las oraciones mediante un parser generado a partir de una gramática insensible al contexto. En una fase posterior los resultados obtenidos fueron comprobados y eventualmente corregidos. Tras la selección de los análisis correctos se pasó a su almacenamiento en un banco de datos desarrollado especialmente a este fin: la Linguistic Data Base (LDB) (Base de datos lingüísticos).

1.2. Linguistic Data Base

La Base de datos lingüísticos es un programa de ordenador destinado a almacenar datos disponibles en forma de diagrama arbóreo. Surgió como resultado de un proyecto empezado en 1983. Se trata de un sistema que se conduce con menú y que puede adoptarse a cualquier idioma. Incluso sirve para datos no lingüísticos a condición de que estos datos estén disponibles en forma de estructura arbórea. El sistema permite investigar los datos almacenados buscando por ejemplo ciertas estructuras que se especifiquen. La especificación se hace en términos de funciones (sujeto, complemento, etc.) o en términos de categorías (nombre, verbo, sintagma nominal, sintagma verbal, etc.). El programa puede producir tablas de frecuencias de aparición de los datos investigados. Estos datos pueden constituir un corpus completo. Así se obtienen cómputos de frecuencias con referencia al corpus completo. El sistema también ofrece la opción de definir un subcorpus en el que se investiga la presencia de la estructura especificada. El programa de la LDB junto con el corpus enriquecido y analizado del inglés lo ponemos a disposición de instituciones académicas con fines de investigación bajo unas condiciones muy favorables. Actualmente utilizamos el sistema de la LDB en el curriculum del departamento de inglés para cursos de sintaxis. Van Halteren & Van den Heuvel (1990) es un manual que contiene una descripción muy detallada del funcionamiento y de las posibilidades de uso de la LDB.

1.3. Cómputo de estructuras sintácticas del inglés moderno

El corpus del inglés analizado morfosintácticamente y la LDB se utilizan como fuente de datos y como instrumento de trabajo en la realización de un nuevo proyecto empezado hace dos años. Este proyecto tiene como objetivo reunir datos con respecto a las estructuras sintácticas más corrientes del inglés moderno, de su frecuencia y su distribución. Se trata de estructuras de la oración y también de estructuras sintagmáticas. El programa LDB se usa como medio más apropiado para sacar las estructuras del corpus analizado y para determinar sus frecuencias. Dentro de poco saldrá un estudio de Aarts y otros colaboradores que dará cuenta de los resultados obtenidos en este proyecto.

1.4. Tools for Syntactic Corpus Analysis (TOSCA)

La lematización manual del corpus de inglés de 130.000 palabras fue llevada a cabo por estudiantes postgraduados de los cinco departamentos de inglés implicados en el proyecto CCPP. A pesar de tener un manual especial para la lematización el número de inconsistencias encontradas al comparar los trabajos de los diferentes analizadores resultó ser demasiado elevado. Obligó a intercalar unas etapas de revisión antes de pasar al análisis sintáctico automatizado. Esta experiencia llevó a formular un nuevo proyecto de investigación a principios de los años 80. A este proyecto le dimos el

nombre de Tools for Syntactic Corpus Analysis (TOSCA): 'instrumentos para el análisis sintáctico de corpus'. (véase Aarts & Van den Heuvel 1985). Tales instrumentos debían ofrecer un sistema de análisis completamente automatizado de textos en bruto, es decir evitando al máximo una preelaboración manual del texto. En el proyecto adoptamos un nuevo formalismo para describir las reglas gramaticales. Es el formalismo de la Extended Affix Grammar (EAG): 'gramática de afijo ampliada'. Se trata de un tipo de gramática desarrollada en la informática para definir lenguas artificiales. También se utiliza para la descripción formal de lenguas naturales. Una amplia discusión de su aplicación al inglés y otra al español se encuentran en Oostdijk (1991) y en Hallebeek (1992). Tiene dos niveles, que constituyen dos gramáticas libres de contexto unidas. Las reglas gramaticales son del tipo rescritural. La gramática del primer nivel es una gramática libre de contexto corriente. Tiene símbolos terminales y no terminales. En la descripción de una lengua natural los terminales son los elementos lexicales y los no terminales son categorías como sintagma nominal, sintagma verbal, nombre, verbo, etc. A los símbolos no terminales se añaden 'afijos'. Estos contienen información gramatical que sobre todo concierne los rasgos de subcategorización. Hay que pensar en información referente al género, número, persona gramaticales. Se puede considerar el 'afijo' como un parámetro para transferir información de un lugar de la gramática a otro, pues diferentes categorías pueden tener los mismos 'afijos'. Los 'afijos' sirven también como un instrumento para imponer restricciones a la aplicación de las reglas de la gramática. La EAG tiene una condición inherente que exige que los nombres idénticos de afijos dentro de una regla supongan valores idénticos de estos afijos. Esto quiere decir que, por ejemplo, el afijo 'número' utilizado en varias partes de la misma regla debe tener la misma realización: o singular o plural. De esta forma el segundo nivel de la gramática le concede al conjunto el poder de una gramática sensible al contexto. Disponemos de un parser generator que convierte la gramática formal en un programa de análisis automático.

El proceso de análisis tiene dos fases: la lematización (o sea el análisis morfológico) y el análisis sintáctico. Al concluir cada una de estas fases el analista tiene la oportunidad de intervenir en los resultados. Luego se seleccionan los resultados del análisis que se van a incorporar en la LDB. La gramática del inglés compuesta por una colaboradora de nuestro equipo y que sirve como instrumento de análisis automatizado produce diagramas de árbol muy detallados de la estructura de enunciados. Se describen estas estructuras en términos de funciones sintácticas y de categorías morfológicas y sintagmáticas. El método que seguimos garantiza un alto grado de consistencia en los resultados obtenidos. Prevalece la calidad y la consistencia del análisis sobre la cantidad de texto analizado. El material producido se presta a una exploración lingüística subsiguiente.

Actualmente se está llevando a cabo el análisis morfosintáctico de un nuevo corpus del inglés británico compilado en 1988, como producto del proyecto TOSCA. El tamaño de este corpus es de 1.500.000 palabras. Contiene 75 fragmentos de 20.000 palabras. Cada uno de estos fragmentos se ha tomado de diferente obra de acuerdo con una detallada repartición en clases de texto y temas de la obra. Junto con unos conocimientos relevantes de la persona del autor el corpus se presta al estudio de variación lingüística. Oostdijk (1988) ofrece una justificación de la metodología empleada en la composición de este corpus del inglés.

1.5. International Corpus of English (ICE)

El proyecto ICE se inició en 1988 en Londres (Survey of English Usage). Persigue la composición de diferentes corpus de un millón de palabras cada uno, respectivamente del inglés británico, americano, canadiense, australiano, nuevozelandés, indio, nigeriano y africano del este. Se van a utilizar estos corpus para estudiar las variedades nacionales del inglés. El equipo de investigación TOSCA participa en las actividades de este proyecto. Se pretende que se vaya a encargar de la lematización de los corpus haciendo uso del lematizador (tagger) desarrollado en el proyecto TOSCA.

1.6. Análisis Sintáctico Automatizado de Textos Españoles

El formalismo EAG utilizado en el proyecto TOSCA sirve no sólo para la descripción del inglés sino también para la descripción de cualquier otro idioma. De ahí que se formularan en la segunda mitad

de los años 80 dos proyectos: uno dirigido al análisis automatizado del árabe estándar moderno (Ditters 1986) y otro al análisis del español peninsular (Hallebeek 1987). Como fruto de estos proyectos surgieron dos gramáticas formales y dos corpus de 500.000 palabras cada uno. En lo que sigue discutiremos con más detalle el proyecto del español realizado en los años 1985-1989 que lleva como nombre Análisis Sintáctico Automatizado de Textos Españoles. Distinguimos tres productos salidos de las actividades desarrolladas en relación con este proyecto: el corpus de textos, la gramática formal y el lexicón.

1.6.1. El corpus del español

En la visión de la Lingüística de Corpus le corresponde al corpus por lo menos una función doble:

- por una parte constituye una base de datos en forma de oraciones, o unidades de mayor extensión, para ser analizadas;
- pero por la otra sirve de material para probar las gramáticas formales parciales o completas desarrolladas antes de tener su forma definitiva.

Para el Corpus de Nimega quedó fijada una extensión total de 500.000 palabras (tokens) en fragmentos de 20.000 palabras de texto seguido cada uno tomados de 25 obras publicadas en España entre 1975 y 1984. La variación de temas y estilos es la siguiente:

1. FICCIóN	2. NO-FICCIóN
1.1. novela	2.1. instructivo-descriptivo: manuales y ensayos
1.1.1. general	2.1.1. pedagogía
1.1.2. policíaca	2.1.2. artes
1.1.3. de ciencia ficción	2.1.3. sociología
1.1.4. de aventuras	2.1.4. antropología
1.1.5. de humor	2.1.5. política
	2.1.6. armamento
1.2. teatro	2.1.7. historia
	2.1.8. feminismo
	2.2. narrativo: biografías, cartas y libros de viaje

Decidimos incluir en el Corpus sólo fragmentos de lenguaje escrito. El lenguaje hablado ofrece particularidades propias - piénsese en fenómenos de elipsis, de errores de concordancia, en la abundancia de giros y anacolutos - y representa también una forma de lengua menos completa en estructuras sintácticas complejas. Nuestro propósito fue componer una gramática formal lo más completa posible. Por esto, optamos por un corpus que contiene textos representativos de más de un registro del lenguaje escrito en el sociolecto más prestigioso del español: el lenguaje culto. Además establecimos que los textos serían:

- sólo de autores españoles y no hispanoamericanos;
- únicamente en prosa;
- de ficción y no-ficción en la proporción de 1 : 2;
- escritos por hombres y mujeres en la proporción de 2 : 1.

Dejamos fuera de consideración la inclusión de material periodístico puesto que posee un estilo y un registro muy propios que en nuestra opinión se prestan más a estudios lexicológicos y

morfosintácticos parciales: la introducción de palabras nuevas, el uso de estructuras sintácticas no corrientes.

Con vistas al posterior análisis gramatical se agregó al pasar los fragmentos de textos al ordenador en el cuerpo de los textos una codificación referente a la presencia de palabras y expresiones extranjeras, a la supresión de elementos del texto como fotos, gráficos, citas, acotaciones, fragmentos en poesía, etc. También añadimos signos que dan información tipográfica sobre el tipo de caracteres empleados (mayúsculas, minúsculas, cursiva, cifras árabes y romanas), la presencia de renglones en blanco y la división del texto en párrafos. Usamos un signo especial para marcar el principio de cada oración nueva.

1.6.2. La gramática formal

La gramática formal compuesta por nosotros reúne los siguientes requisitos:

- describe el uso de la lengua, concretamente el español escrito contemporáneo;
- contiene módulos para los sistemas de reglas morfológico y sintáctico; los aspectos semánticos y pragmáticos quedan fuera de consideración;
- ofrece la descripción más consistente y completa posible dentro de los marcos que se acaban de indicar;
- es capaz de analizar - previa conversión en parser - un fragmento de cualquier texto con un mínimo de intervenciones.

El formalismo aplicado en la descripción de las reglas gramaticales es el de las Extended Affix Grammars (Gramáticas de Afijo Ampliadas). Ya hemos explicado en 1.4. que proviene de la informática. Es un instrumento que se presta a la producción y al análisis tanto de lenguas artificiales como de lenguas naturales. De hecho, su campo de aplicación en la Informática es tan amplio que se utiliza no sólo para la definición de lenguas de programación sino también como una lengua de programación. Se distinguen dos campos principales en las reglas de la gramática: las morfológicas y las sintácticas. Las morfológicas son las que quedan bajo el nivel de la categoría morfológica. Explican la estructura de la palabra reconociendo procesos de derivación, la categoría morfológica del vocablo y el valor de los sufijos flexivos. La configuración de la gramática es modular. Sus partes se corresponden con las unidades gramaticales conocidas, tales como Noun Phrase (Sintagma Nominal), Adjective Phrase (Sintagma Adjetivo), Verb Phrase (Sintagma Verbal), etc. Los fenómenos sintácticos que rebasan los límites de una unidad sintáctica (como el de la coordinación y del comparativo) tienen sistemas de reglas propios incorporados en los módulos de la gramática con que se relacionan. El análisis automatizado se lleva a cabo en dos pasos: primero el análisis léxicomorfológico y luego el análisis sintáctico. Una descripción detallada de la gramática así como una evaluación de su funcionamiento se halla en Hallebeek (1992).

1.6.3. El lexicón

El lexicón es un componente indispensable del sistema de análisis automatizado. Junto con las reglas morfológicas de la gramática se encarga, tras conversión en parser, del análisis morfológico y lexical de textos. Identifica las palabras sueltas indicando su categoría morfológica y eventuales subcategorías (modo, tiempo, persona, número, género, etc.). El lexicón constituye un conjunto de raíces, provistas de indicación de la categoría y eventuales subcategorías presentes (género de nombre, clase de verbo, presencia opcional de preposición fija, etc.). Las raíces son las partes únicas de palabras que forman morfemas libres. Las palabras derivadas se reconocen por las reglas morfológicas de la gramática. Esto nos permite limitar el número de entradas lexicales hasta el mínimo. El lexicón contiene ahora aproximadamente 5000 raíces. Al probar el análisis lexicomorfológico sobre textos del corpus resultó que sólo un 4 por ciento de las raíces encontradas en estos textos faltaba en el lexicón.

Actualmente estamos considerando las posibilidades de realizar el análisis morfosintáctico del corpus completo del español. Lo que falta es suficiente tiempo de investigación, o mejor dicho suficientes fondos para crear este tiempo. En lo que precede hemos señalado que disponemos de los elementos necesarios para llevar a cabo esta tarea. Tales elementos son los que son propios de la lingüística de corpus:

- un corpus en machine-readable form;
- una gramática formal que describe la estructura del español;
- un parser generator que convierte la gramática en parser;
- un lexicón computacional del español;
- un sistema que une corpus, parser y lexicón, y que dirige el proceso de análisis dándole al analista la posibilidad de intervenir;
- un sistema de base de datos que hace accesibles los resultados del análisis a otros investigadores.

2. Análisis formal del objeto preposicional en español

En los manuales de gramática al uso la función sintáctica llamada aquí objeto preposicional recibe poca o ninguna atención. Caso que se menciona, suele dársele el nombre de suplemento, siguiendo el término introducido por Alarcos Llorach en su artículo de 1968 «Verbo transitivo, verbo intransitivo y estructura del predicado» (incluido en Alarcos Llorach 1978). Martínez García (1986) publicó una monografía dedicada al suplemento siguiendo la metodologóa de Alarcos. En lo que sigue formulamos primero una definición de la función llamada objeto preposicional y luego explicamos la manera en que está incorporada esta función en nuestra gramática formal del español.

2.1. El objeto preposicional

2.1.1. Introducción

El objeto preposicional (en forma abreviada OP) es un modificador o complemento del verbo que se une al verbo por medio de una preposición fija. El que sea fija esta preposición significa que por regla general no se puede sustituirla por otra preposición sin modificar la significación del verbo. Comparemos:

(1a) quejarse de una persona,
(1b) quejarse con razón.

Es diferente la significación de 'quejarse' tal como se emplea en (1a) y en (1b). El verbo en (1a) va acompañado de un objeto que tiene la forma de un sintagma preposicional (SPREP): 'de una persona'. El verbo de (1b), en cambio, no necesita un objeto expreso. Otro ejemplo más claro aún de la diferencia de significado que obtiene un verbo cuando va con un SPREP como objeto o sin tal objeto es el siguiente:

(2a) contar con una persona,
(2b) contar con una máquina.

---

[1] Este procedimiento fue sugerido por Martínez García (1986: 109) a fin de distinguir complementos de sujeto o de objeto y suplementos.

[2] Se trata de un método que Kraak (1968: 81) aplica para reconocer el objeto preposicional en holandés. Resulta que también es válido para el español.

Aparentemente se trata en (2a) y en (2b) del mismo verbo 'contar' acompañado de un SPREP introducido por la preposición 'con'. Sin embargo, el verbo tiene sentidos diferentes en los dos casos. Tanto en (1b) como en (2b) es posible sustituir la preposición 'con' por 'sin' sin cambiar la significación del verbo: 'quejarse sin razón, contar sin una máquina'. En (1a) y en (2a) la sustitución de las preposiciones 'de' y 'con' por otra traería consigo un cambio del significado del verbo: 'quejarse con una persona', 'contar para una persona'. Estos verbos tienen ahora los mismos significados que en (1b) y (2b). Existe una relación directa entre verbo y preposición en (1a) ('quejarse de') y (2a) ('contar con'). De hecho los dos son inseparables. Se supone que en estos casos la preposición constituye una extensión del verbo y que no tiene una significación propia que añada a la relación verbo y sintagma nominal (SN). Preposición y SN cumplen la función sintáctica de un OP (objeto preposicional). Una relación directa entre verbo y preposición no existe en (1b) y (2b). Los SPREP 'con razón' y 'con una máquina' cumplen la función de un complemento adverbial (o circunstancial). En estos ejemplos la preposición 'con' añade su significación propia a la relación verbo y SN.

2.1.2. Maneras de reconocer el OP

El OP siempre tiene la forma de un SPREP. Sin embargo, el complemento circunstancial (o adverbial) con frecuencia se realiza bajo la misma forma. ¿Cómo distinguir sintácticamente el SPREP en función de complemento circunstancial del mismo sintagma en función de objeto preposicional? Existen tres pruebas sintácticas que ayudan a identificar el objeto preposicional.

Prueba de la sustitución del SPREP por un pronombre adverbial [1]

(3a) soñar con los ojos abiertos,
(3b) soñar así,
(3c) *soñar con ellos.
(4a) soñar con las vacaciones,
(4b) *soñar así,
(4c) soñar con ellas.

(El asterisco * señala las construcciones con desvío sintáctico o semántico.)

El verbo 'soñar' de (3a) y (3b) contiene un complemento adverbial. En (4a) y (4c), en cambio, hay un OP. Esto resulta de la relación diferente entre verbo y preposición. En la primera serie esta relación es tal que es posible sustituir el SPREP por un adverbio 'así' suprimiendo la preposición. Incluso es así que la conservación de la preposición en (3c) produce una estructura en la que la significación del verbo no coincide con la de (3a) y (3b). (Como estructura sintáctica sí que es gramatical.) La relación entre el verbo y la preposición es libre en (3a). En la serie (4), al contrario, la relación entre verbo y preposición es tan estrecha que no se permite la supresión de la preposición si se quiere conservar la significación del verbo. Por esto (4b) no es aceptable y (4c) demuestra que con tal que se conserve la preposición la sustitución del SN que forma parte del SPREP está permitida.

Prueba de la dislocación del SPREP[2]

(5a) No se permite hablar de política,
(5b) *Hablar no se permite de política.

---

[3] La conversión de una oración declarativa corriente en una pseudo cleft sentence es una prueba desarrollada por Van den Toorn (1982: 35-36) para el holandés. No emplea el término de pseudo cleft sino el de cleft. El primero es más correcto.

(6a) No se permite hablar de noche,
(6b) Hablar no se permite de noche.

Las oraciones (5a) y (6a) contienen dos verbos independientes: 'permitir' y 'hablar'. Resulta en (5b) que el SPREP 'de política' y el verbo 'hablar' no se pueden separar. Esta separación sí que es posible en (6b) al tratarse de 'hablar' y 'de noche'. El comportamiento diferente de los dos SPREP otra vez se debe al hecho de que la relación entre verbo y preposición es diferente. Sintácticamente tenemos que ver con dos funciones distintas: un OP en (5a) y un complemento adverbial en (6a).

Prueba de la formación de una seudo-hendida [3]

(7a) Los viejos gōzan de una buena salud,
(7b) De lo que gozan los viejos es (de) una buena salud,
(8a) Los viejos gozan de muchas maneras,
(8b) *De lo que gozan los viejos son muchas maneras.

También en esta prueba se investiga la relación entre verbo y preposición. Resulta que en (7a,b) esta relación es más estrecha que en (8a,b). En (7b) la preposición necesita estar en una posición más cercana al verbo mientras que la agramaticalidad de (8b) demuestra que la preposición se une al SN ('muchas maneras') y no al verbo. (7a,b) contienen un OP y (8a,b) un complemento adverbial. En la seudo-hendida el OP se comporta como el objeto directo precedido de la preposición 'a':

(9a) He visto a Juan,
(9b) Al que he visto es Juan.

2.1.3. Clases de verbos que tienen OP

El OP se combina con verbos intransitivos, transitivos y reflexivos. Esto quiere decir que se encuentra el OP en oraciones cuyo verbo principal va solo (sin atributo u objeto directo), cuyo verbo va con un objeto directo o cuyo verbo va con un pronombre personal reflexivo en función de objeto directo o indirecto. Veremos una por una las distintas clases de verbos.

Verbos intransitivos

Ejemplos de verbos intransitivos que pueden ir acompañados de un OP son: 'soñar (con)', 'pensar (en)', 'contar (con)'. Interesante es la siguiente alternancia - señalada por Martínez García (1986: 86-89) - que se ofrece en algunos verbos:

(10a) El libro pertenece a Pedro,
(10b) El libro le pertenece.

(11a) El pueblo pertenece a la provincia de Madrid,
(11b) El pueblo pertenece a ella.

(10a) tiene un objeto indirecto 'a Pedro' según resulta de la sustitución por 'le' en (10b). La oración (11a) que también contiene el verbo 'pertenece', sin embargo, no permite la sustitución de 'a la provincia de Madrid' por 'le'. Este SPREP cumple la función de un OP. Es la presencia del rasgo +/- humano en el SN que forma parte del SPREP la que determina si es OP u objeto indirecto: + humano produce un objeto indirecto y - humano un OP. Sin embargo, no es así que el objeto indirecto y el OP se excluyan mutuamente en la misma estructura:

(12) Este libro no me sirve de nada,
(13) Esto no me entra en la cabeza.

Las dos oraciones contienen un objeto indirecto 'me' y un OP 'de nada' y 'en la cabeza'.

Verbos transitivos

Existen verbos transitivos que además de un objeto directo tienen un OP:

(14) El inspector basa su sospecha en estos datos,
(15) Fijemos la atención en otro argumento,
(16) Nos invitaron al baile.

Algunos verbos transitivos muestran la misma alternancia de objeto indirecto y OP apuntada en los verbos intransitivos:

(17a) Pablo dedica mucha atención a María,
(17b) Pablo le dedica mucha atención.
(18a) Pablo dedica mucha atención a los estudios,
(18b) Pablo dedica mucha atención a ellos.

Al tratarse de un nombre marcado + humano el SN funciona como un objeto indirecto (17a,b). Si es un nombre marcado - humano la función es la de un OP (17a,b).

Verbos reflexivos

(19) Juan se dedica a la música,
(20) Gómez se basa en otro estudio,
(21) Enrique se jacta de sus hazañas.

Los ejemplos (19)-(21) contienen dos tipos diferentes de verbos reflexivos. 'Dedicarse' y 'basarse' son reflexivos ocasionales (no siempre reflexivos) mientras que 'jactarse' es un reflexivo obligatorio, como lo es también 'quejarse'.

2.1.4. Delimitación del OP

En estudios de gramática española y holandesa se excluyen los complementos de lugar y de tiempo de la función de OP (o suplemento). Esto se hace a pesar de que se trate de verbos que se combinan con preposición fija. Veamos los siguientes ejemplos:

(22a) Pablo entra en la casa,
(22b) Pablo entra en detalles.
(23a) El gobierno no llega a la primavera,
(23b) La comitiva llegó al ayuntamiento,
(23c) Los políticos llegaron a un acuerdo.

En (22a), (23a) y (23b) hay un complemento de lugar o de tiempo que tiene la forma de un SPREP. Martínez García (1986: 112-119) no concede a estos SPREP la función de OP. Se basa en la consideración de que resulta posible sustituir 'en la casa', 'al ayuntamiento' por 'aquí', 'ahí', 'allí', suprimiendo las preposiciones 'en' y 'a'. Esta supresión para ella es prueba de que el complemento no

puede ser OP. Sin embargo, vemos que en las siguientes pruebas la relación entre verbo y pronombre adverbial es tan estrecha como la entre el verbo y la preposición tal como vimos en los ejemplos (5)-(6) (prueba de dislocación).

(24a) No se permite entrar en esta casa,
(24b) *Entrar no se permite en esta casa,
(24c) Entrar en esta casa no se permite,
(24d) No se permite entrar aquí,
(24e) *Entrar no se permite aquí,
(24f) Entrar aquí no se permite.

La prueba de dislocación muestra que el pronombre adverbial 'aquí' tiene la misma restricción que el SPREP 'en casa' en combinación con el verbo 'entrar'. Parece ser así que los pronombres adverbiales como 'aquí', 'allí', 'ahí', 'ahora', 'entonces' incluyen en su significación la preposición 'en', 'a'.

En estudios de gramática holandesa (e.o. Toorn 1982: 34) se identifica el OP con un complemento causal. Es decir un complemento que expresa la causa o el motivo de la acción expresada en la oración. Se excluyen por lo tanto también los complementos de lugar y de tiempo. Una interpretación que limita así los SPREP que pueden funcionar como OP deja sin explicar los casos en que un verbo va acompañado de una preposición fija que introduce un complemento de lugar o de tiempo. Pensamos en verbos como: 'proceder de', 'provenir de', 'quedar en', 'ir a', 'volver a'. Consideramos que cualquiera que sea el valor semántico del complemento este tiene la función de OP siempre que se una al verbo por medio de una preposición fija.

2.2. La gramática formal

2.2.1. Características generales

El formalismo en que está escrita la gramática formal del español compuesta por nosotros es del tipo EAG (Extended Affix Grammar). La EAG desarrollada originalmente para lenguajes de programación es una gramática a dos niveles: una base y una extensión. La base es una gramática libre de contexto. Los símbolos no terminales de las reglas de esta gramática son ampliadas con un tipo de parámetros o atributos que se añaden a ellas y que se llaman 'afijos'. Los valores de estos afijos se definen en otra gramática, la del segundo nivel, también libre de contexto. Por ejemplo, el símbolo no terminal NOMBRE de la gramática del primer nivel se aumenta con los afijos género, número y persona obteniendo el símbolo:

NOMBRE (género, número, persona).

En la gramática del segundo nivel se definen género como masculino o femenino, número como singular o plural y persona como primera, segunda o tercera. La EAG tiene la condición inherente de que dos afijos con el mismo nombre presentes en diferentes posiciones de la misma regla deben tener el mismo valor. De modo que en esta regla:

SN : DET (género, número),
NOMBRE (género, número),
SADJ (género, número).

los afijos género y número tienen cada vez el mismo valor: o masculino o femenino, o singular o plural. De esta manera las dos gramáticas libres de contexto se insertan convirtiéndose en una gramática sensible al contexto.

Disponemos de un parser generator para convertir la gramática formal en un analizador automático (parser). Esto significa que la gramática formal no es un programa de ordenador ni contiene indicaciones para el deseado funcionamiento de la máquina. Escribir la gramática formal tampoco exige conocimientos de programar. Unicamente es necesario dominar las convenciones notacionales formales propias de las reglas (empleo de coma, punto, punto doble, etc.). El contenido de las reglas es puramente lingüístico. Nuestra gramática del español no contiene información pragmática ni semántica, se basa sólo en datos morfosintácticos. En la descripción de las estructuras sintácticas seguimos una alternancia sistemática de categoría y función. Por ejemplo, la categoría del SN se analiza en las funciones de DETERMINANTE, PREMODIFICADOR, NUCLEO y POSTMODIFICADOR. El NUCLEO del SN a su vez se realiza en la forma de un nombre o de un pronombre. El POSTMODIFICADOR en la forma de un SADJ (sintagma adjetivo), SPREP, SN o cláusula relativa. En casos en que faltan en la tradición gramatical nombres de categorías o funciones para determinadas unidades sintácticas nos hemos visto obligados a introducir nuevos nombres. Citamos como ejemplo la función de COMPLEMENTO que reúne todos los complementos del verbo y que se realiza en la nueva categoría de un SINTAGMA COMPLEMENTARIO.

La gramática se compone de tres partes: el lexicón, las reglas morfológicas y las reglas sintácticas. Estos tres componentes están escritos en el formalismo de EAG. El lexicón no es más que una serie de definiciones de subcategorías como son RAIZ DE NOMBRE, ADJETIVO, VERBO. Son, pues, los morfemas gramaticales y lexicales independientes. Las formas lexicales (los radicales) son rescrituras alternativas de estas subcategorías. Van acompañadas de información gramatical en forma de afijos. Se trata de afijos de género, número, persona, tipo de verbo, etc. La parte de la gramática que contiene las reglas morfológicas describe e interpreta los fenómenos de derivación y de flexión presentes en verbos, adjetivos, nombres y adverbios. Estas reglas disponen de listas de los prefijos y sufijos (derivativos, flexivos, evaluativos, etc.). Al analizar un texto primero se subdivide este texto en enunciados (utterances) independientes. Luego se dividen los enunciados en palabras sueltas. Las reglas del lexicón y las reglas morfológicas juntas identifican las clase de cada una de estas palabras con sus categorías gramaticales de género, número, persona, tiempo, modo, etc. El proceso de análisis automático es interactivo de forma que después del análisis léxico-morfológico le permite al lingüista intervenir en los resultados corregiéndolos o suprimiéndolos si lo cree necesario.

Después de la fase de interpretación morfológica se pasa al análisis sintáctico del enunciado completo. El componente de las reglas sintácticas comprende una serie de módulos correspondientes a las unidades conocidas con el nombre de SN, SADJ, SADV (sintagma adverbial), SPREP, SV (sintagma verbal), O (oración).

2.2.2. Incorporación del OP a las reglas formales

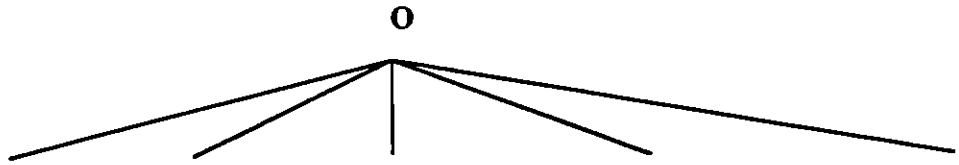
La función del OP se introduce en las reglas de la gramática en diferentes lugares. En primer lugar en el lexicón. Cualquier radical de verbo que permite opcionalmente u obligatoriamente la aparición de un OP va acompañado de información respecto a la preposición o las preposiciones exigidas por este verbo. A modo de ejemplo reproducimos la regla referente a la raíz 'pens' de 'pensar'.

```
(1)  RAIZ VERBO (pr, tipo, A): «pens»,
     coord pr 46 (pr),
     coord tipo 22 verbo (tipo);
(2)  coord pr 46 (VACIO): ;
     coord pr 46 (EN): ;
     coord pr 46 (SOBRE): .
(3)  coord tipo 22 verbo (TR): ;
     coord tipo 22 verbo (INTR): .
```

En la regla (1) se lee que 'pens' es la rescritura del radical de un verbo (RAIZ VERBO). Este verbo va con un afijo llamado 'preposición (pr)'. El valor o los valores de este afijo se definen en una regla especial llamada 'coord pr 46 (pr)'. Esta regla se elabora en (2) donde se ve que los valores de

la preposición pueden ser nulo (vacío), la preposición 'en' o 'sobre'. Como en regla (1) los nombres del afijo 'pr' tras RAIZ VERBO y tras 'coord pr 46' coinciden, sus valores deben ser idénticos. Lo mismo ocurre con el afijo 'tipo', que aparece tras RAIZ VERBO y en la regla 'coord tipo 22 verbo'. En (3) está la regla completa en la que se citan como tipos posibles del verbo 'pensar': TR (transitivo) e INTR (intransitivo). El valor de afijo A que se halla en última posición tras RAIZ VERBO refiere a la clase flexiva del verbo e indica que este verbo pertenece a los verbos en 'ar'. Las reglas introducidas por el elemento 'coord' se llaman 'predicados'. Como se habrá notado, es un tipo de reglas en las que se imponen restricciones a los valores de los afijos. El afijo 'pr' tiene como valores entre otros A, CON, DE, EN, PARA, POR, SIN, SOBRE. El predicado escoge en este caso de entre los valores posibles los pertinentes para el verbo 'pensar'.

Vamos a ver ahora como se incorpora el OP en las reglas sintácticas de la oración. Esta tiene la siguiente estructura global en términos de funciones:



(SUJETO) (ADVERBIAL) NUCLEO (COMPLEMENTO) (ADVERBIAL)

Las funciones entre paréntesis son opcionales. Resulta que sólo el NUCLEO es obligatorio e indispensable para la oración española. La función de NUCLEO se realiza en forma de SV. La función de COMPLEMENTO toma la forma de una categoría que llamamos SINTAGMA COMPLEMENTARIO. Este reúne todos los complementos verbales; entre ellos el objeto directo e indirecto, el complemento de sujeto y de objeto y también el OP. En la gramática se han formulado reglas especiales para definir qué complementos se combinan con qué clases (tipos) de verbos. El verbo transitivo tiene como complemento obligatorio el objeto directo y como complemento opcional el objeto indirecto. Dentro de los verbos transitivos hay una subclase que además del objeto directo permite o exige la presencia de un complemento de objeto: 'Nombraron a Juan director'. En la primera parte de este informe hemos visto que el OP aparece usado con verbos intransitivos, transitivos y reflexivos. Hemos apuntado también algunas particularidades con respecto a la presencia de otros complementos al lado del OP:

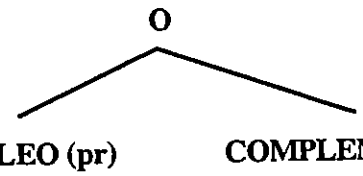
(4) No me sirve de nada,
(5) Pablo dedica mucha atención a los estudios.

La frase (4) contiene un OP y un objeto indirecto y (5) un OP y un objeto directo. Todos estos datos se incorporan a las reglas que definen los complementos que pueden ir con los diferentes tipos de verbos. La unión entre verbo y OP se establece en las reglas de la gramática por medio del afijo 'pr'. Acabamos de indicar que en el lexicón figuran las raíces de verbos provistas de información sobre la posibilidad de ir con un OP y de información sobre la preposición concreta con la que va. Este dato pasa como valor de afijo de la raíz del verbo al SV y por este al NUCLEO de la oración. En forma simplificada:

(6) NUCLEO (pr) : SV (pr).

El afijo 'pr' puede tomar los valores 'con', 'de', 'en', 'por', etc. El OP forma parte del sintagma complementario y se realiza en forma de un SPREP: 'Cada uno piensa en sus cosas'. Se agrega al SPREP el mismo afijo 'pr' que obtiene su valor de la preposición concreta que se encuentra usada en el SPREP. A través del SPREP (pr) el valor del afijo pasa a la función OP. Por esta se traslada al SINTAGMA COMPLEMENTARIO para llegar a la función de COMPLEMENTO que forma parte

de las funciones de la oración. Así conseguimos que a nivel de la estructura de la oración tanto el NUCLEO como el COMPLEMENTO lleven un afijo 'pr':



La condición inherente a las reglas de la EAG exige identidad de valor entre los afijos del mismo nombre. En este caso el valor del afijo 'pr' que acompaña al NUCLEO y al COMPLEMENTO deben ser iguales. Esto significa que sólo en caso de que coincidan la preposición exigida por el verbo - el valor 'pr' del NUCLEO - y la preposición presente en el SPREP - el valor 'pr' del COMPLEMENTO - la gramática reconoce un OP. Sin embargo, como en ella no se incluye información semántica, la gramática identificará también como OP el SPREP de 'hablar de día'. Formalmente no hay diferencia entre 'hablar de dinero' y 'hablar de día'. La secuencia 'hablar de día' producirá un análisis ambiguo porque 'de día' figura en el lexicón como frase adverbial.

Terminamos representando aquí abajo el análisis de la frase 'de María nos acordábamos', tal como lo produce nuestra gramática en la pantalla. En esta oración la función del OP se realiza delante del verbo, NUCLEO de la oración. Esta posición del COMPLEMENTO es una variante de la posición posverbal. Las dos variantes están incorporadas a las reglas gramaticales. El diagrama arbóreo no va de izquierda a derecha sino de arriba hacia abajo. La terminología gramatical usada en nuestras reglas es inglesa. La hemos adaptado al español para este ejemplo. Llamamos la atención a la gran cantidad de afijos que acompañan a algunas categorías y funciones, ver por ejemplo NOMBRE, SV SIMPLE y VERBO.

```
============================================================
 de María nos acordábamos.
============================================================

   ORACION SIMPLE(TR,REF,IND)
    COMPLEMENTO(DE)
       :     SINTAGMA COMPLEMENTO(DE)
       :     OP(DE)
       :        :  SPREP SIMPLE(DE)
       :        :     SUBORD2(DE)
       :        :       P(DE)
       :        :         :  de
       :        :     COMP2(,FEM,SING)
       :        :        :  SN SIMPLE(FEM,SING,DEF)
       :        :        :     NUCLEO(,PROP,DEF,FEM,SING)
       :        :        :       NOMBRE(PROP,UNM,DEF,FEM,SING)
       :        :        :         :  María
    NUCLEO(PLU,PA,IND,DE,TR,REF)
       :  SV SIMPLE(PLU,PA,IND,DE,TR,REF)
       :     COMPLEMENTO(DO,PLU,PA,PER)
       :       SINTAGMA COMPLEMENTO(,)
       :        :OBJETO DIRECTO(PER,AMBIGUO,PLU,PA)
       :        :  NOMBRE(PRO,PER,UNM,DEF,OBJ,AMBIGUO,PLU)
       :        :    nos
       :     NUCLEO(PLU,PA,IND,IMPERF,DE,TR,)
       :       VERBO(PLU,PA,IND,IMPERF,DE,TR,)
       :         :  acordábamos
    MARCADOR FINAL(PUNTO)
      MARCA PUNTUACION(PUNTO)
       :.
```

Referencias

AARTS, J. & W. MEIJS (eds) (1986): Corpus linguistics. Amsterdam, Rodopi.

AARTS, J. & T. VAN DEN HEUVEL (1980): «The Dutch computer corpus pilot project.» ICAME News, 4: 1-8.

AARTS, J. & T. VAN DEN HEUVEL (1985): «Computational Tools for the syntactic analysis of corpora». Linguistics 23: 303-335.

ALARCOS LLORACH, E. (1978a): Estudios de gramática funcional del español. Gredos, Madrid (2e ed.).

ALARCOS LLORACH, E. (1978b): «Verbo transitivo, verbo intransitivo y estructura del predicado.» Alarcos Llorach (1978a: 109-123).

DITTERS, E. (1986): «An extended affix grammar for the noun phrase in modern standard Arabic.» Aarts & Meijs (eds): 47-77.

HALLEBEEK, J. (1987): «Hacia un sistema de análisis sintáctico automatizado: el proyecto ASATE». Martín Vide, C. (Ed) (1987: 303-315): Actas del II Congreso de Lenguajes Naturales y Lenguajes Formales. Universidad de Barcelona.

HALLEBEEK, J. (1992): A Formal Approach to Spanish Syntax. Rodopi, Amsterdam-Atlanta GA.

HALTEREN, H. VAN & TH. VAN DEN HEUVEL (1990): A Linguistic exploitation of syntactic databases. Rodopi, Amsterdam-Atlanta GA.

KRAAK, A. & W.G. KLOOSTER (1968): Syntaxis. Stam-Kemperman, Culemborg-Keulen.

MARTÍNEZ GARCÍA, H. (1986): El suplemento en español. Gredos, Madrid.

OOSTDIJK, N. (1988): «A corpus for studying linguistic variation», ICAME Journal, 12: 3-14.

OOSTDIJK, N. (1991): Corpus linguistics and the automatic analysis of English. Rodopi, Amsterdam-Atlanta GA.

TOORN, M.C. VAN DEN (1982): Nederlandse grammatica. Wolters-Noordhoff, Groningen (8a ed.).

DESDE EL CÁLCULO DE PREDICADOS A LAS GRAMÁTICAS MODULARES

Antonio Menchén / Víctor Díaz

Área de Lenguajes y Sistemas Informáticos
Facultad de Informática y Estadística
Universidad de Sevilla

Resumen

Las Gramáticas Lógicas Modulares (MLGs), se corresponden con las Gramáticas Atribuidas del modelo no-lógico, y permiten haciendo uso de una serie de operadores y símbolos terminales lógicos, obtener a partir del árbol de análisis de una frase su forma lógica.

Hemos observado que puede construirse, desde PROLOG , un sistema de Procesamiento del Lenguaje Natural, modular y sistemático, tratándose en este trabajo de aplicarlo a los problemas propios del castellano, y mostrando a través del acceso a bases de datos relacionales la forma de utilizarlo en otros entornos.

Este trabajo se divide en cuatro partes: CALCULO DE PREDICADOS Y REPRESENTACION SEMANTICA, GRAMATICAS LOGICAS, IMPLEMENTACION DE LAS MLGs y MEJORAS DEL FORMALISMO. En la primera se hace un planteamiento del problema que nos ocupa: representación del lenguaje natural en un lenguaje de ordenador, y constituye la justificación de la segunda y tercera partes. En la segunda se muestra el progreso que trae la utilización de una MLG frente a una Gramática de Clásulas Definidas (DCG), en la tercera se hace una exposición detallada de todos los problemas que plantea la implementación de estos formalismos y en la cuarta se hace una propuesta de futuras investigaciones en este terreno.

1. Cálculo de Predicados y Representación Semántica

Es evidente que sólo una parte de nuestro conocimiento se trasmite a través del lenguaje natural. Siempre han existido un gran número de lenguajes, que llamaremos formales, desprovistos de ambigüedades y que obedecen a reglas estrictas de construcción. Entre estos lenguajes están los de programación en los que habremos de 'convertir' cualquier conocimiento que se desee procesar.

En [1] se apunta que los humanos tenemos una gran capacidad de desambigüación y que esto es debido, simplemente, a que implícitamente poseemos un conocimiento (más o menos exacto, por otra parte), del contexto en que debe de entenderse un párrafo, una frase o una palabra.

Parece pues plausible que tomando un lenguaje formal (conveniente), que represente una frase en LN y añadiéndole la representación del contexto, podremos disponer de programas que: permitan acceder a datos, a bases de conocimiento, a entornos de sistemas operativos, etc, sin que el usuario tenga que conocer SQL, PROLOG o UNIX, pongamos por caso.

Tomando como lenguaje formal el cálculo de predicados, llamaremos estructura lógica a la representación en este lenguaje de la frase sin tener en cuenta el contexto. El lenguaje de programación que utilizaremos para procesar la frase es PROLOG.

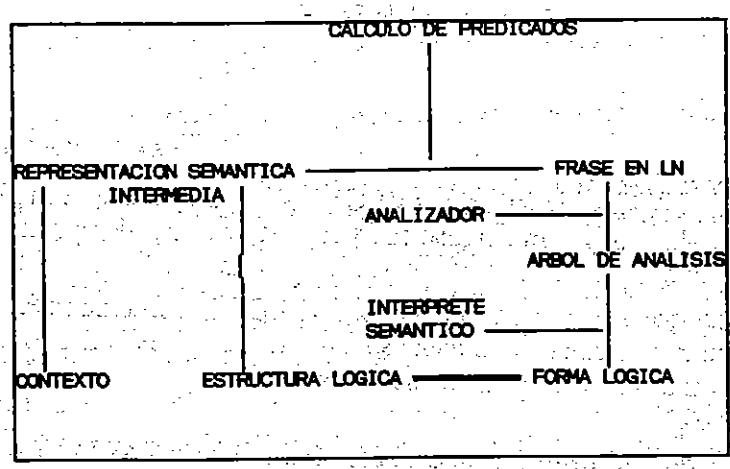
La forma lógica es la representación en PROLOG de la estructura lógica.

Además PROLOG presenta la ventaja de que pueden implementarse fácilmente bases de datos relacionales en él.

Necesitamos, pues, un programa PROLOG que a partir de una sentencia en LN obtenga su forma lógica.

Ese programa se puede obtener de la gramática del lenguaje escrita mediante símbolos no-terminales que son predicados con argumentos, es decir, mediante una gramática lógica, la cual se traduce a un programa PROLOG que analice las sentencias aplicándose otro programa que se denomina «compilador de reglas», obteniéndose por fin el árbol de análisis. Este árbol es una estructura que representa, bien el árbol sintáctico, bien la forma lógica directamente o bien estas dos informaciones aunque de una forma implícita.

Todo lo dicho se resume en el siguiente cuadro:



que se completa con:

1.1. El Lenguaje del Cálculo de Predicados

Es el lenguaje que representa fórmulas lógicas. Esencialmente son nombres de predicados, que representan las relaciones entre otros predicados, variables o constantes, unidos por las llamadas conectivas:

$$\neg, \wedge, \vee, \rightarrow, \equiv$$

Los paréntesis encierran a las expresiones que están relacionadas y separan las distintas fórmulas lógicas, y las comas separan a las expresiones.

Además dispondremos de los cuantificadores:

$$E\ 'A$$

Hay otros elementos que no son de lógica de primer orden pero que se utilizarán para poder dar el sentido correcto a las frases: notación tipada, términos proposicionales y cuantificadores generalizados.

La notación tipada nos permite más poder de expresión del cálculo de predicados de primer orden, para aplicarlo al LN:

De la fórmula:

$$(\forall x)\ (\exists y)\ q(x,y)$$

es tipada como:

$$(\forall x | t1(x))\ (\exists y | t2(y))\ q(x,y)$$

donde se quiere significar que para todos los x de un dominio t1, existe al menos un y de un dominio t2 que verifican la relación q.

Esto hace que sea más precisa la fórmula lógica expresando el significado. Por ejemplo, la frase: Cada hombre ama a una *mujer*, se representa:

$$(\forall x | man(x))\ (\exists y | woman(y))\ loves(x,y)$$

(Como puede observarse a los predicados se les ha dado el nombre en inglés para representar, el significado de la palabra más que la palabra en un idioma determinado.)

Puede verse la equivalencia de las fórmulas que siguen y se aplicarán en el resto del trabajo:

$$(\forall x | t1(X))\ p(x) \equiv (\forall x)\ (t1(x) \rightarrow p(x))$$

$$(\exists x | t1(x))\ p(x) \equiv (\exists x)\ (t1(x) \wedge p(x))$$

Esto permite tratar la fórmula inicial como su equivalente:

$$(\forall x)\ (t1(x) \rightarrow (\exists y)\ (t2(y) \wedge q(x,y)))$$

Así la frase: Cada hombre ama *a una mujer*, se representa:

$$(\forall x)\ (man(x) \rightarrow (\exists y)\ (woman(y) \wedge loves(x,y)))$$

1.2 Representación en Estructuras Lógicas

En [2] se enuncia «un*a frase en lenguaje natural está formada gramaticalmente por un verbo y un cierto número de sintagmas nominales*».

Nos basaremos en el siguiente principio de asociación: A toda estructura gramatical se le puede asociar una estructura lógica estableciendo las dos transformaciones:

a) El verbo por un predicado.

b) Cada sintagma nominal por una cuantificación restringida al contenido de ese sintagma.

La frase: Un p*erro persigue a un gato*, $(\exists x | dog(x))\ (\exists y | cat(y))\ follows(x,y)$

que es equivalente a: $(\exists x)\ (dog(x) \wedge (\exists y)\ (cat(y) \wedge follows(x,y)))$

De esta forma, de una manera más o menos afortunada, podremosrepresentar todo tipo de frases:

Así la frase: García *trabaja en todos los proyectos dirigidos por López*, queda representada:

$$(\forall x)\ (project(x) \wedge leads(LOPEZ_1, x) \rightarrow works_{on}(GARCIA_1, x))$$

Esta representación distingue entre «*proposición*» y «*descripción*». El verbo expresa, principalmente, una proposición que puede ser afirmada, negada, o ser la base de una pregunta o una orden, el sintagma nominal expresa una descripción de un conjunto de objetos.

1.3 Equivalencia entre Estructura Lógica y Forma Lógica

Según hemos visto ya, un determinante que se corresponde perfectamente con el cuantificador universal es cada.

La representación:

$$(\forall x)\ (man(x) \rightarrow (\exists y)\ (woman(y) \wedge loves(x,y))$$

Vamos a ponerla de la forma:

$$(\forall x)\ (P)$$

donde x es una variable sobre la que se cuantifica y P es la proposición que es cierta para todo valor de x. Sabemos que si P tiene la forma:

$$A \rightarrow B$$

su equivalente es:

$$\neg A \vee B$$

que equivale a:

$$\neg (A \wedge \neg B)$$

En [3], McCord propone como forma lógica de la anterior frase

each(man(X), ex(woman(Y),loves(X,Y))).

donde each(P,Q) :- not((P , not(Q))).

Con lo que está perfectamente justificado el decir que la forma lógica anterior representa la estructura lógica de la frase en LN.

De igual manera, para el cuantificador existencial, se obtendría:

ex(P,Q) :- P , Q.

1.4 Formas Lógicas asociadas a distintas Categorías Gramaticales

Tal como se apunta en [4], se llama forma lógica, LF, a la representación del significado de una frase en un lenguaje lógico. Además, se dice algo muy importante: la *estructura de la frase en subfrases, determina la construcción de la forma lógica en subformas lógicas*. Sin darnos cuenta hemos dado un paso de gigante en cuanto a la implementación, ya que a partir de ahora prescindiremos de la representación en el cálculo de predicados (aunque siempre partiremos de él), traduciendo directamente una frase a su forma lógica. Ahora estamos en disposición de escribir la gramática de un LN y obtener por el principio de composición de formas lógicas, la forma lógica de una frase, pero este es un tema que dejaremos para más adelante.

Cuantificadores como muchos, unos pocos o la mayoría pueden tratarse de la misma forma que cada o uno, sin más que definir convenientemente su forma lógica.

La frase: Unos *pocos tigres son mansos*, tendría como forma lógica:

few(tiger(X),tame(X)).

Al primer argumento, del predicado, se le llama base: especifica la base o el rango de items sobre lo que se cuantifica. Al segundo se le llama foco: especifica aquella parte de la sentencia sobre la que se hace énfasis (en los adverbios se verá más claro este segundo concepto).

Además se verifica:

```
few(Base,Foco):-
card(Base,B),
card((Base,Foco),BF),
small(BF,B).
```

donde:

```
card(P,N):-
setof(P,P,S),
length(S,N).
```

El predicado setof(X,P,S) nos devuelve en S el conjunto ordenado (lista) de todos los términos X que hacen que P se cumpla, y length(S,N) devuelve en N el número de elementos de S. El predicado small(M,N) se cumple para valores de M/N menores a uno dado.

Con muchos se realizaría algo análogo. La diferencia estaría en el valor límite de la fracción anterior. Para el cuantificador muchos, es necesario tener datos con que comparar este valor, ya que es imposible determinar en valor absoluto el límite de la fracción M/N.

Otro cuantificador que aparece a menudo es el determinante definido. Su forma lógica será:

the(P,Q):- card(P,1), P, Q.

En cuanto a los adverbios, si tomamos la frase: Juan compró ayer un gato, el adverbio nos dice el tiempo de validez de la sentencia.

En [3] la traducción a forma lógica es:

yesterday(ex(cat(X),buy(juan,X))),

Luego el argumento es una forma lógica. Cuando esto ocurre se dice que la categoría gramatical es intencional (en oposición a extensional). La mayoría de los adverbios tienen dos argumentos, por lo que la forma lógica es:

adverbio(Base,Foco).

En frases adverbiales aparece con frecuencia el fenómeno de la **enfatización**, su importancia es tal, que determina en gran medida la forma lógica.

Así la frase: Ju*an siempre compra libros en el Corte Inglés*, si se enfatiza dónde compra los libros, tiene como forma lógica:

always((book(X),buy(juan,X)):E, at(corte_ingles,E)).

Mientras que, si se enfatiza qué compra, dará la forma lógica:

always(at(corte_ingles,buy(juan,X)), book(X)).

La mayoría de los adjetivos son extensionales, y son semánticamente nombres:
Así la frase: Juan ve una casa ro*ja*, tiene como forma lógica:

ex((house(X),red(X)), see(juan,X)).

Otros adjetivos son intencionales: anterior, futuro, falso, verdadero, etc. No modifican entidades, como en el caso anterior, sino a tipos de entidades (No se puede decir anterior(X) o falso(X), siendo X una entidad)

Como ejemplo, tenemos la frase: Juan *se encuentra con su anterior maestro*, dando como forma lógica:

ex(former(teacher(X,juan)), meet(juan,X)).

1.5 El Conocimiento del Contexto

En [4] se define la representación semántica intermedia como una combinación de elementos que son necesarios para la interpretación semántica. Esta representación semántica intermedia estaría formada por las siguientes partes:

La estructura lógica, que se corresponde con la forma lógica de [3].
El contenido conceptual.
El acto de discurso.
Las anotaciones pragmáticas.

1.5.1. El Contenido Conceptual

Tomemos la frase: J*uan lee un libro*, cuya forma lógica es:

ex(book(X),read(juan,X)).

Es evidente que Juan puede leer un libro perfectamente (a menos que sea analfabeto), pero lo que no está claro, al menos en un contexto normal, es que el libro pueda leer a Juan. Este conocimiento es posible añadirlo a la estructura lógica. Vamos a proponer una forma sistemática y que servirá para la mayoría de los casos.

En [2] se propone que cualquiera que sea el dominio se haga uso de estos dos predicados:

instance(<objeto>,<clase>),
propval(<objeto>,<propiedad>,<valor>).

El segundo predicado nos va a servir para indicar que es Juan quien lee, y el libro el objeto leído:

propval(r,agent,JUAN_1),
propval(r,object,x).

donde se ha realizado la abstracción instance(r,read): r es la instancia de read.

Con esta notación la representación intermedia queda:

$$(\exists x|instance(x, book))(\exists r|instance(r, read)), propval(r, agent, JUAN), propval(r, object, x)$$

Estableciendo un sistema de jerarquía de clases dotadas de propiedades que heredan unas de otras, se podrá deducir que el agente de una instancia de «read» es una persona, porque «read» es una subclase de «process» y una propiedad de «process» es «agent» que tiene como tipo de valor «person».

En [3], McCord implementa esta información conceptual en lo que llama ranura (slot), exigiendo, por ejemplo, que el agente del verbo give sea humano, y que un hombre (como el caso de Juan), tenga como tipo semántico el ser humano.

Por lo tanto allí no se puede hablar de representación intermedia desde el momento en que el contenido conceptual no se interpreta, sino que son restricciones semánticas dentro de la gramática.

1.5.2 Actos de Discurso

Supone el incorporar a la representación intermedia que se obtiene de la estructura lógica y los contenidos conceptuales, la intención del locutor al emitir una frase, de esta forma podemos tratar más adecuadamente preguntas u órdenes. En [3], el acto de discurso se incorpora a la forma lógica. Así una pregunta como: *¿Cada hombre ama a María?*, da la forma lógica:

yesno(each(man(X), E), love(X,mary):E).

1.5.3. Anotaciones Pragmáticas

Permiten resolver ambigüedades que aun quedan para entender el sentidode la frase. Pueden referirse a la cuantificación, a informaciones que completan el contexto o, a las que sitúan el tiempo o el modo.

2. Gramáticas Lógicas

Una característica muy apreciable de la programación lógica es que las gramáticas para los lenguajes (naturales y formales), pueden ser expresadas fácilmente como programas lógicos y que pueden ser ejecutados para el análisis y la síntesis de estos lenguajes. Dentro de esta idea PROLOG resulta ser el lenguaje lógico más indicado.

2.1. Gramáticas Lógicas Modulares (MLGs)

Las gramáticas lógicas más extendidas y que se encuentran implementadas en cualquier versión de PROLOG, son las llamadas Gramáticas de Cláusulas Definidas (DCGs). Consisten en una generalización de las gramáticas de estructura de frase libre de contexto, en la que los símbolos no-terminales pueden contener argumentos, donde, posteriormente, se obtendrán las estructuras de árbol sintáctico o representación semántica.

Las DCGs presentan el problema de la imposibilidad de separar el árbol de análisis de la representación semántica de la frase. Para el LN, donde la acción semántica depende muchas veces del contexto o del énfasis, este formalismo se muestra poco adecuado y potente. El problema que se presenta a menudo es que la conexión entre la estructura lógica y la regla de producción es muy remota y complicada, por lo que diseñar el analizador a partir de una DCG es imposible.

Se hace necesario separar el componente sintáctico del semántico, para poderlos manipular por separado.

En las Gramáticas Lógicas Modulares (MLGs), debidas a Michael McCord, del Watson Research Center, las reglas gramaticales son:

```
S→a : OP-LF
```

que se traduce en la separación entre el componente sintáctico y el semántico.

A OP-LF se le llama terminal lógico, y está formado por el operador OP, que determina como se combinarán las representaciones semánticas parciales, y LF es una forma lógica que establece una relación entre los argumentos que intervienen en la regla gramatical.

En una DCG para cada no-terminal de la forma:

```
nt(...)→B1:B2:...
```

se obtiene un nodo de un árbol:

nt(X1,X2,...).

donde las Xis son los nodos asociados a los Bis.

Para las MLGs el nodo del árbol asociado con cada regla va a ser de la forma:

syn(nt:Arg, Hijos).

donde nt es un no-terminal, Arg es el primer argumento de nt(...) (si no tiene argumentos es []), e Hijos es la lista de nodos producidos por el cuerpo de la regla. Cada no-terminal contribuye con un simple syn, mientras que cada terminal lógico contribuye con un elemento simple, llamado igual que el mismo.

No todos los símbolos no-terminales van a contribuir con un nodo en el árbol de análisis. Esto se hace con el fin de evitar árboles de derivación innecesariamente grandes. A los que contribuyen se les llama no-terminales fuertes, mientras que el resto se llamarán no-terminales débiles.

El intérprete semántico tomará este árbol y obtendrá por diversas manipulaciones la forma lógica.

El componente de interpretación semántica actúa generalmente con los items semánticos, los cuales son de la misma forma que los terminales lógicos, excepto que pueden ser formas lógicas complejas (resultado de la combinación de varios items semánticos).

Ejemplos de terminales lógicas son:

```
l-hombre(X).
Q/P-cada(P,Q).
```

donde l (unificación izquierda) hace que la forma lógica asociada se una a la izquierda (durante la interpretación semántica) con la forma lógica del ítem semántico con el que se esta recombinando. El operador Q/P hace que P sea unificado con la forma lógica del ítem semántico con el que se está recombinando. Por ejemplo: si el ítem «cada» modifica al ítem «hombre» (como en la frase nominal «cada hombre»), el resultado es el ítem: @P-cada(hombre(X),P) donde @P es nuevo operador que hace que P sea unificado con la forma lógica del ítem con el que se está recombinando.

Ejemplo: cuando @P-cada(hombre(X),P) modifica a l-vive(X), el resultado es el ítem semántico l-cada(hombre(X),vive(X)), que en el último paso de la interpretación queda como cada(hombre(X),vive(X)) y puede interpretarse como «cada hombre vive».

Para escribir el programa de traslación (el compilador de reglas), hemos de tener en cuenta que cada no-terminal, además de los dos argumentos diferencia de listas, ha de tener dos argumentos representando a las estructuras de análisis. Si el no-terminal es fuerte habrá que tomar un sólo argumento de la estructura de análisis que represente el syn asociado con la expansión de este no-terminal (el otro argumento será nil). Si es débil los dos argumentos representarán la diferencia de listas de los nodos producidos por la expansión del no-terminal.

3. Implementación de las MLGs

Para poder abordar una aplicación de procesamiento de lenguaje natural es necesario determinar dos aspectos: cuáles son el tipo de oraciones que serán reconocidas y la definición del campo semántico de la palabras que aparecerán en dichas oraciones.

Al determinar estos dos aspectos podremos restringir el número de oraciones entendibles por la aplicación y reducir los problemas de ambigüedad.

El estudio sintáctico nos permitirá definir la gramática que, aunque depende de la aplicación particular, siguen las mismas pautas para su definición independientemente de la aplicación.

Podemos hacer estudios separados de: los sintagmas nominales, los tipos de oraciones compuestas, los sintagmas verbales y los sintagmas preposicionales.

También es importante resolver los problemas de aquellos sintagmas que no estén colocados en su situación natural, que normalmente serán resueltos utilizando variables que recojan información sobre ellos para recolocarlos en la estructura.

El campo semántico de las palabras, en especial de los nombres y verbos, puede ser útil para desechar oraciones sin sentido y para resolver ambigüedades sintácticas, ya que obligan a que además de cumplir la estructura sintáctica coincidan los tipos que pueden combinarse.

Podemos definir una jerarquía de tipos semánticos y asociar a cada palabra del lexicón un tipo, e incluso si esta palabra pudiera tener modificadores, a qué tipo semántico deben de pertenecer. Los tipos los representaremos con dos puntos seguido de un identificador de tipo.

Ej: perro:animal

4. Aplicación a la Consulta de Bases de datos Relacionales

El proceso de trasformar una oración a una forma objetivo entendible por la aplicación particular que se haga, se divide en una serie de fases, según el siguiente cuadro:

Fase	Transformación	
1 División en tokens:	Oración	-> Lista de tokens
2 Análisis sintáctico:	Lista de tokens	-> Arbol sintáctico
3 Análisis semántico:	Arbol sintáctico	-> Forma lógica
4 Análisis objeto:	Forma lógica	-> Forma objetivo

4.1. División de Tokens

En ella se divide la oración en palabras y se le asocia un token que es ella misma, aunque en algunas se refleja en el token algunas características como empezar por mayúsculas o ser un número. También pueden descomponerse las contracciones y eliminar los signos de puntuación. La oración será un tipo cadena de caracteres y se devolverá un lista PROLOG de tokens.

Ej: Oración: «Juan *llegó a las 10.*»

Lista de tokens: [may(juan),llego,a,las,num(10)]

4.2. Análisis sintáctico

Se obtiene la estructura de árbol ya comentada. Este árbol que contiene la estructura sintáctica de la oración es generado automáticamente por la gramática, e incluye los operadores de cuyo funcionamiento hablaremos en la fase tercera. La gramática será definida en función del tipo de oraciones que se pretendan interpretar.

Ej: Queremos obtener una gramática que reconozca los siguientes tipos de oraciones relacionadas con una universidad:
1 - ¿ Qué asignaturas tiene informática ?.
2 - ¿ Cuantos estudiantes de álgebra obtuvieron un notable en física ?.
3 - ¿ Aprobó cada estudiante 3 asignaturas ?.

Vemos que las oraciones son todas interrogativas y en tercera persona.

Simplifica considerar que una oración está formada por un verbo con una serie de modificadores de dicho verbo, en dichos modificadores se incluyen el objeto directo, el objeto indirecto, el sujeto e incluso cualquier sintagma nominal, con o sin preposición, que regularmente aperezca con dicho verbo.

Siguiendo este método tenemos que sintácticamente tendríamos la regla prototipo:

Oración -> Verbo + Modificadores.

De esta manera en la primera oración «asignaturas» e «informática» modifican al verbo «tiene», y en la segunda «física» modifica a «obtuvieron».

La idea es, por tanto, reflejar en el lexicón los verbos con todos los modificadores y sus tipos, quedándonos en el lexicón los siguientes predicados que recogen en sus argumentos: la palabra, su forma lógica, la variable y tipo asociada al sujeto y una lista con las variables, tipos e información sobre qué preposición rige de todos los que consideraremos modificadores verbales, a este último argumento se le denomina ranura. Separamos de la ranura el modificador sujeto por facilidad a la hora de implementar. Con las ranuras permitimos que haya una misma forma verbal pero con distintos tipos de modificadores, es deseable mantener el mismo identificador, a ser posible su forma en infinitivo, para la forma lógica del verbo incorporándose un número para distinguirlos. También usaremos variables que nos unifiquen los núcleos de los modificadores con los argumentos de la forma lógica del verbo.

Ej: verbo(tiene,tener1(X,Y),X:fac,[(Y:asig)]).
verbo(obtuvieron,obtener1(X,Y,Z),X:est,[(Y:not),(Z:prep(en):asig)]).
verbo(aprobo,aprobar1(X,Y),X:est,[(Y:asig)]).

Vemos que asociamos como tipo, el tipo del núcleo del sintagma modificador. La inclusión de los tipos nos permitirá desechar modificadores que no sean coherentes con el verbo, pues al girar sobre el verbo la construcción de la gramática, leído éste sabremos cuántos modificadores tendremos y qué tipo tiene por su predicado asociado en el lexicón.

Un problema sería que puede haber delante del verbo modificadores y ésto según nuestra regla nos obliga a recolocar ese premodificador detrás del verbo. Quizás la manera más sencilla de abordar este problema es obligar a escribir las oraciones según unos patrones, aunque intentando que estos patrones sean naturales. Unas normas podrían ser:

– escribir el sujeto siempre al lado del verbo,
– permitir que sólo haya un modificador como máximo delante del verbo,
– si no hay pronombres interrogativos, colocar primero el verbo y después el sujeto en las oraciones interrogativas.

Aunque parezcan muy rígidas, estas normas se corresponden con las usadas normalmente al hablar. Las ventajas de usarlas son:

– descubrir fácilmente el sujeto asociado al verbo. La relación sintáctica entre el sujeto y verbo es muy fuerte, ya que deben coincidir en número, persona y tipo,
– tener que recolocar como máximo un modificador, esto es fácil usando un argumento Recol que recoja las características de ese modificador y una vez obtenido el verbo, ver que modificador se adapta a él. Si no fuera el sujeto, entonces por las normas anteriores el siguiente modificador debería ser el sujeto.

Queda también estudiar los sintagmas nominales. Distinguiremos los sintagmas nominales según el tipo de palabra que sea su núcleo, nos reduciremos a dos posibles casos: nombre propio y nombre común.

En un sintagma nominal nos interesa saber el número Num, persona Pers y género Gen para resolver problemas sintácticos, como dijimos antes nuestras oraciones son en tercera persona y de ahí podemos obviar la marca de persona. Consideraremos que el tipo de un sintagma nominal Tipo es el tipo semántico de su núcleo y éste nos servirá para resolver las ambigüedades ya expuestas. Puesto que los sintagmas nominales se refieren a entidades, éstas pueden ser cuantificadas mediante presentadores del sintagma, por tanto, también nos interesa saber qué tipo de cuantificación Dtipo tiene. Suponindo que X es la variable asociada al sintagma nominal, estas informaciones pueden ser recogidas mediante una etiqueta de la siguiente forma:

X:DTipo:Num:Gen:Tipo.

Atendiendo a los distintos tipos de núcleos podemos ver que:

Si el núcleo es un nombre común no estará cuantificado ni normalmente modificado y le asociaremos la etiqueta con sólo la información de su género y tipo. Para definir una entidad no cuantificada usaremos el identificador de tipo def, así la etiqueta sería:

X:def:sg:Genero:TipoN.

Si el núcleo es un nombre propio puede estar cuantificado y puede tener modificadores. Así vemos que en la oración 2 álgebra modifica a estudiantes. Estos modificadores se tratan análogamente a los de los verbos con la particularidad de que no se da el caso de la recolocación. Su etiqueta estará formada por su tipo, número y género al que se le suma el tipo del cuantificador que será unificado mediante otra regla.

Con estas ideas la regla prototipo sería:
Sintagmanominal -> Nombrepropio.
Sintagmanominal -> Determinante + Nombrecomún + Modificadores.

Los determinantes cuantifican a los nombres y tienen un operador especial P/Q para determinar el ámbito de cuantificación que será explicado en la fase 3. Esta cuantificación puede ser de distintos tipos dependiendo de su significado. Una característica es que son una clase cerrada de palabras por lo tanto pueden estudiarse particularmente cada uno. En las oraciones ejemplo vemos que funcionan como determinates: que, 3, cuantos ,cada, y un. La regla prototipo de un determinante será reconocer el determinate y devolver su género, número y tipo.

Los modificadores tienen un tratamiento iterativo consistente en elegir una ranura de las ranuras e intentar reconocer un modificador de esas características, normalmente un sintagma nominal con o sin preposición, hasta que se hayan reconocidas todas. Para su buen funcionamiento es necesaria la correcta definición de las ranuras para los verbos y los nombres, mientras que los modificadores verbales suelen llevar preposiciones a o en, en los modificadores nominales suelen ser precedidos por la preposición de, pero esta regla es quebrantada a menudo. A esta regla compete recolocar si hubiera algún precomplemento, para ello usamos un par de argumentos (Recol,Recol1) cuyo significado es el siguiente:

- si los dos son la lista vacía, o sea ([],[]) entonces o no ha habido precomplemento o este ha sido un precomplemento pero ya ha sido recolocado,
- si son de la forma (recoloca(_),[]) entonces ha habido precomplemento que se ha conseguido recolocar,
- si es de la forma (recoloca(_),recoloca(_)) entonces ha habido un precomplemento pero no ha podido ser recolocado.

Otro aspecto a destacar son los adjuntos que son otro tipo de modificadores, como por ejemplo los adverbios, las complementos circunstanciales, las cláusulas de relativo y otros más que no son recogidos en los argumentos de las formas lógicas de los verbos. Estos son también reconocidos mediante esta regla de tal manera que si existe un modificador que no es coherente con ninguna ranura debe intentarse reconocer como estos otros modificadores. Denominaremos complementos a los modificadores recogidos en las ranuras.

Debido a que las cláusulas de relativo pueden o no tener su propio sujeto es necesario incrementar el número de argumentos de modificadores con XRel, que guarda las características del núcleo nominal ya leído para ver si coincide o no con el verbo subordinado. Los adverbios que semánticamente destacan una parte de la oración o foco, llamados adverbios focalizadores tienen un tratamiento especial que será detallado en la fase 3.

La regla prototipo para los modificadores sería:

Modificadores -> Elegirranura + Complemento + OtrosModificadores.
Modificadores -> Adjunto + OtrosModificadores.
Modificadores -> Noquedanranuras.

Queda el aspecto de definir qué no no-terminales serán fuertes, ya que es fundamental para el buen funcionamiento del análisis semántico. Estos se corresponden con los elementos de la oración más importantes como los sintagmas nominales, sintagmas preposicionales, determinantes, oraciones subordinadas y adverbios. No se definen como tal los núcleos de los sintagmas preposiciones y reglas auxiliares.

La gramática definitiva quedaría:

```
oracion ->
   sujetoverbo(Ranuras,Recol)
   : modificadores(sv,Ranuras,Recol,[],_).
sujetoverbo(Ranuras,Recol) ->
   premodificador(X,Numn,Tipon,Prep)
    : nucleoverbal(Y,Numv,Tipov,Ranuras)
    : coincide(Recol,Prep,(X:Numn:Tipon),(Y:Numv:Tipov)).
sujetoverbo(Ranuras,[]) ->
   (@P)-si_no(P)
   : nucleoverbal(X,Num,Tipo,Ranuras)
   : sintagmanominal((X:Dtipo:Num:Gen:Tipo)).
premodificador(X,Num,Tipo,Prep) ->
   sintagmanominal((X:Dtipo:Num:Gen:Tipo)).
```

```
premodificador(X,Num,Tipo,Prep) ->
   +Prep
   : preposicion(Prep)
   : sintagmanominal((X:pi:Num:Gen:Tipo)).
coincide([],[],Etiqueta,Etiqueta) ->
   [].
coincide(recoloca(X:Tipo,Prep),Prep,(X:N:T),(Y:N1:T1)) ->
   sintagmanominal((Y:Dtipo:N1:Gen:T1)).
nucleoverbal(X,Num,Tipo,Ranuras) ->
   +V
   : verbo(V,Pred,X:Tipo,Num,Ranuras)
   : 1-Pred.

sintagmanominal((X:def:sg:Gen:Tipo)) ->
  +Tnoun
  : nombrepropio(Tnoun,Noun,Gen,Tipo)
  : 1-(X=Noun).
sintagmanominal((X:Dtipo:Num:Gen:Tipo)) ->
   det((X:Dtipo:Num:Gen))
   : +N
   : nombrecomun(N,P,X:Tipo,Num,Gen,Ranuras)
   : 1-P.
   : modificadores(sn,Ranuras,[],[],(X:_:Num:_:Type).

det((X:Dtipo:Num:Gen)) ->
   +D
   : dt(D,B,F,P,X,Num,Gen,Dtipo)
   : F/B-P.

modificadores(Cat,RanurasI,Recol,Recol2,XRel) rightarrow
     elegir((Tiporanura:X:Tipo),RanurasI,RanurasF)
     : complemento(Tiporanura,X:Tipo,Recol,Recol1)
   : modificadores(Cat,RanurasF,Recol1,Recol2,XRel).
modificadores(Cat,Ranuras,Recol,Recol,XRel) ->
   adjunto(Cat,XRel)
   : modificadores(Cat,Ranuras,Recol,Recol,XRel).
modificadores(Cat,[],Recol,Recol,XRel) ->
   [].

adjunto(sv,_) ->
   pp((X:_),(@X)).
adjunto(sv,_) ->
   avp.
adjunto(sn,XRel) ->
   pp(XRel,r).
adjunto(sn,XRel) ->
   relclse(XRel).
pp((X:_),Op) ->
   +Prep
   : prep(Prep,Pred,Y:Tipo,X)
   : Op-Pred
   : sintagmanominal((Y:Dtipo:Num:Gen:Tipo)).
avp ->
   +Adv
   : adv(Adv,Pred,Op)
   : Op-Pred.
relclse(X:_:Num:Gen:Tipo) ->
   +que
   : vhead(X,Num,Tipo,Ranuras)
  : modificadores(vp,Ranuras,[],[],_).
relclse(X:_:Num:Gen:Tipo) ->
```

```
    +que
     : sintagmanominal(Y:_:Numn:Genn:Tipon)
     : vhead(Y,Numn,Tipon,Ranuras)
     : postmods(vp,Ranuras,recoloca(X:Tipo,[]),[],_).

complemento(cd,(X:Tipo),recoloca(X:Tipo,[]),[]) ->
    [].
complemento(cd(a),(X:Tipo),recoloca(X:Tipo,a),[]) ->
    [].
complemento(ci,(X:Tipo),recoloca(X:Tipo,a),[]) ->
    [].
complemento(cp(Prep),(X:Tipo),recoloca(X:Tipo,Prep),[]) ->
    [].
complemento(cd,(X:Tipo),E,E) ->
    sintagmanominal((X:_:_:_:Tipo)).
complemento(cd(a),(X:Tipo),E,E) ->
    +a
    : sintagmanominal((X:_:_:_:Tipo)).
complemento(ci,(X:Tipo),E,E) ->
    +a
    : sintagmanominal((X:_:_:_:Tipo)).
complemento(cp(Prep),(X:Tipo),E,E) ->
    +Prep
    : sintagmanominal((X:_:_:_:Tipo)).
```

4.3. Análisis Semántico

En ella se hace un recorrido tipo preorden en el árbol de la oración de tal manera que se van generando análisis semánticos parciales para cada subárbol mediante las operaciones que son detalladas más adelante, una vez llegada a la raíz del árbol tendremos el análisis semántico de toda la oración, o sea, la forma lógica de toda la oración.

Para realizar el análisis semántico asociamos a cada nodo del árbol sintáctico un ítem semántico aumentado que consta de tres elementos: una etiqueta, un operador y una subforma lógica, estos items semánticos aumentados los representaremos de la siguiente manera:

Etiqueta Operador Subforma lógica.

De una manera automática, a las hojas del árbol, que se les denomina items semánticos, le asociaremos la etiqueta terminal:[], y a los nodos intermedios que son las raíces de los distintos subárboles, que se les denomina items sintácticos, les asociaremos el operador id y la subforma lógica true. El análisis será por tanto obtener la subforma lógica del ítem semántico aumentado de la raíz.

El análisis irá modificando sólo los operadores y subformas lógicas, pues la etiqueta será siempre la correspondiente a la raíz del subárbol que es analizado.

Para obtener esta representación semántica usamos dos tipos de operaciones con el árbol sintáctico: operación de modificación, operación de transformación.

4.3.1. Operación de Modificación

Obtiene un ítem semántico aumentado como la combinación de dos items semánticos aumentados. Con esta operación vamos encajando unas subformas lógicas con otras. Para realizar esta operación nos fijamos sólo en los operadores y subformas lógicas de los items semánticos aumentados.

Ej: Supongamos un subárbol con raíz R y cuyos hijos de izquierda a derecha son H1, H2, y H3, entonces:

R se modifica con H3 dando H3*,
H3* se modifica con H2 dando H2*,
H2* se modifica con H1 dando H1*,
H1* es la representación semántica del subárbol con raíz R.

4.3.2. Operación de Transformación

Transforma la estructura del árbol sintáctico, para ello utiliza información recogida en las etiquetas e incluso en los operadores de los items semánticos aumentados. Esta estructura es transformada de dos maneras:

Operación de reordenación: Cambia el orden de los hijos de un nivel.
Con esta operación ordenamos las subformas lógicas, de tal manera que la operación de modificación se efectúe correctamente. Esta operación es realizada previamente a la de modificación para cada nivel. Para ordenarlas se le asigna un número de orden a los distintos tipos de etiquetas. Con los adverbios focalizadores se debe primero establecer cuál es su foco, una vez colocado el adverbio y su foco al final se establece la ordenación de los demás items.

Ej: Supongamos un subárbol con raíz R y cuyos hijos de izquierda a derecha son H1, H2, y H3, entonces mediante la reordenación podría quedar H2, H1 Y H3.

Operación de elevación: Eleva un nodo hijo al nivel del padre, colocándolo como su hermano izquierdo más cercano. La elevación es necesaria para resolver los problemas de ámbitos de cuantificadores. Antes de modificar un ítem se comprueba si debe ser elevado.

Ej: Supongamos un subárbol con raíz R y cuyos hijos de izquierda a derecha son H1, H2, y R1, y a su vez R1 tiene los hijos S1 y S2, entonces al elevar S2, el resultado sería la raíz R con los hijos H1, H2, S1 y R1*, donde R1* es la representación semántica de R1 sin contar con S1.

Esta operaciones de transformación permiten disociar el orden de escritura sintáctico con el orden en el que aparecen en la forma lógica, lo cual tiene especial aplicación en la búsqueda del ámbito de los determinantes y del foco de los adverbios focalizadores.

4.3.3. Operadores Semánticos

Para entender mejor la operación de modificación haremos un estudio de los operadores más comunes:

Operador l : Está asociado a los núcleos de los sintagmas nominales y a los verbos. Es un operador de conjunción a la izquierda y simplemente une por la izquierda con el operador & su forma lógica asociada con la forma lógica obtenida hasta el momento.

Operador r: Esta asociado a complementos preposicionales de núcleos nominales. Su comportamiento es similar al operador l, pero es de conjunción a la derecha.

Operador id: Asociado al nodo raíz del subárbol que se está operando. Funciona como operador identidad.

Operador F/B: Es usado para los determinantes y algunos pronombres. Funciona como un predicado con dos argumentos: B llamado base y F llamado foco. Unifica su base B

con la forma lógica obtenida en su mismo nivel a su derecha y sube al nivel superior convertido en el operador @F, para poder unificar su foco F. Este operador aparece con asiduidad como nodo más a la izquierda de un nivel.

Operador @P : Es usado con adverbios no focalizadores y aditamentos. Unifica con P la forma lógica obtenida en ese nivel a su derecha.

Operador B<F: Es usado para los adverbios focalizadores. Funciona como un predicado con dos argumentos: B llamado base y F llamado foco. Unifica su foco F con la forma lógica obtenida en su mismo nivel a su derecha y se modifica convertido en el operador focal(B,Pred,OpB), para poder unificar su base B con la forma lógica obtenida sucesivamente en su nivel pero a la izquierda, por eso utiliza el operador con esos tres argumentos: B en donde unifica al final la base definitiva, OpB que utiliza para ir construyendo progresivamente B y el argumento Pred en donde obtendrá la forma lógica definitiva.

Operador focal(Base,Pred,OpB): Ver operador B<F.

4.4. Ejemplo de construcción de una forma Lógica

Expondremos un ejemplo de la obtención de una forma lógica antes de adentrarnos en su transformación a forma objetivo para una aplicación, supongamos que queremos analizar la siguiente oración: «Di*eron clases todos los profesores de cada departamento*»

El árbol sintáctico generado será el siguiente:

```
s:[]
   @P - sino(P)
   1 - darclase(X)
   np:X:todo:singular:masculino:profesor
      det:X:todo:singular:masculino
         F/B - cada(B,F)
      1 - profesor(X,Y)
      np:Y:todo:singular:masculino:departamento
         det:Y:todo:singular:masculino
            F1/B1 - cada(B1,F1)
         1 - departamento(Y)
```

Las subformas lógicas han sido obtenidas de la oración salvo la subforma sino(P), que se usa para representar oraciones interrogativas cuyo resultado es o verdadero o falso. Tenemos un nombre común profesor que está modificado por otro que es departamento, un verbo dar clases con un solo complemento sujeto y dos determinantes cuantificadores universales, esto es reflejado como de tipo todo en la etiquetas para det y para np. La gramática expuesta debe modificarse en las reglas nucleoverbal y determinante para recoger el verbo dar clase y el determinante todos los.

En el análisis semántico prescindiremos de la información de las etiquetas asociadas al género, número y tipo semántico de los núcleos de los sintagmas nominales.

Se irán analizando sucesivamente y por este orden los subárboles cuya raíz está etiquetada como det:X:todo, det:Y:todo, np:Y:todo, np:X:todo y s:[]. Como mencionamos antes los items semánticos asociados a estos nodos se modifican con sus hijos de derecha a izquierda. Veamos detalladamente el análisis:

Subárbol con etiqueta det:X:todo: Tenemos un hijo, luego no hay operación de transformación, sólo la operación de modificación [mod]. Cualquier operador operado con id es él mismo y su subforma lógica. La etiqueta como dijimos es la del padre [padre].

terminal:[] F/B cada(B,F)
[padre] det:X:todo id true
[mod] det:X:todo F/B cada(B,F)

Subárbol con etiqueta det:Y:todo: Análogo al anterior.

terminal:[] F1/B1 cada(B1,F1)
[padre] det:Y:todo id true
[mod] det:Y:todo F1/B1 cada(B1,F1)

Subárbol con etiqueta np:Y:todo: En líneas generales se pretende que los núcleos nominales o verbales queden mediante la operación de ordenación [ord] lo más a la derecha posible de un nivel, dejando más a la izquierda los cuantificadores, por tanto en este nivel no hay ordenación, al operar el nodo np con terminal obtenemos np:Y:todo l departamento(Y), que debe ser modificado con F1/B1. Como dijimos esta modificación unifica la base B1 y sube al nivel superior convertido en @F1, para unificar su foco F1.

det:Y:todo F1/B1 cada(B1,F1)
terminal:[] l departamento(Y)
[padre] np:Y:todo id true
[mod] np:Y:todo l departamento(Y)
[mod] np:Y:todo @F1 cada(departamento(Y),F1)

Subárbol con etiqueta np:X:todo: Hay reordenación [ord], para que el núcleo profesor sea el primero en ser modificado. Una vez ordenados, los items son modificados, pero al haber dos cuantificadores, nos encontramos con que uno de ellos tiene ya su base unificada pues su operador es @F1, como el padre tiene etiqueta todo, implica que vamos a unificar la base de otro cuantificador, por lo que este ítem np:Y:todo debe ser elevado. Al elevarse el ítem anterior se modifica profesor(X,Y) con la base B de det:X:todo de manera similar a la ya dicha.

det:X:todo F/B cada(B,F)
terminal:[] l profesor(X,Y)
np:Y:todo @F1 cada(departamento(Y),F1)

[ord]

det:X:todo F/B cada(B,F)
np:Y:todo @F1 cada(departamento(Y),F1)
terminal:[] l profesor(X,Y)

[padre] np:X:todo id true
[mod] np:X:todo l profesor(X,Y)
[elev] np:Y:todo @F1 cada(departamento(Y),F1)
[mod] np:X:todo @F cada(profesor(X,Y),F)

Subárbol con etiqueta s:[]: Nuevamente hay una ordenación de items para colocar el núcleo verbal al final, la subforma sino tiene mayor prioridad que los cuantificadores pues se está preguntando por la validez de toda la oración. Vemos como el ítem elevado se convierte en hermano izquierdo del ítem semántico aumentado de su nivel. En este nivel se unifican las variable F y F1 sucesivamente con las subformas parciales obtenidas. El operador @F al ser modificado con cualquier otro operador Op, se convierte en el operador Op.

terminal:[] @P sino(P)
terminal:[] 1 darclase(X)
np:Y:todo @F1 cada(departamento(Y),F1)
np:X:todo @F cada(profesor(X,Y),F)
[ord]
terminal:[] @P sino(P)
np:Y:todo @F1 cada(departamento(Y),F1)
np:X:todo @F cada(profesor(X,Y),F)
terminal:[] 1 darclase(X)
[padre] s:[] id true

[mod] s:[] 1 darclase(X)
[mod] s:[] 1 cada(profesor(X,Y),darclase(X))
[mod] s:[] 1 cada(departamento(Y),cada(profesor(X,Y),darclase(X)))
[mod] s:[] 1 sino(cada(departamento(Y),cada(profesor(X,Y),darclase(X))))

Quedando el análisis de la oración:

sino(cada(departamento(Y),cada(profesor(X,Y),darclase(X))))

4.5. Ejemplo de construcción de una Forma Objetivo

Las formas lógica son generales y puede ser usadas como punto de partida de cualquier aplicación particular. Supongamos que queremos hacer una interfaz en lenguaje natural para consultas a base de datos, para ello una vez obtenida la forma lógica de la oración correspondiente a la pregunta tenemos que transformarla en una sentencia válida que reconozca el sistema gestor de base de datos SGBD. Veremos como transformarla para ser reconocida para un base de datos PROLOG.

En la fase de traducción necesitamos:

- definir los predicados asociados a los cuantificadores, adverbios y predicados no léxicos. En nuestro ejemplo sino y cada.
- descomponer aquellos predicados que contienen a su vez predicados que no sean reconocidos como tales por prolog. Este problema aparece asociado a los complementos circunstanciales o predicados cuyos argumentos puedan ser formas lógicas a su vez.
- tener la traducción a relaciones de la base de datos de todos los posibles predicados reconocibles, especialmente nombres y verbos. En nuestro ejemplo departamento, profesor y darclase.

El predicado sino puede ser definido:

```
sino(P) :-
  P,!,write(«Si, es cierto»).

sino(_) :-
  write(«No, no es cierto»).
```

El predicado cada(P,Q) como se mencionó puede ser definido como:

```
cada(P,Q) :-
  not( (P , not(Q)) ).
```

Si tenémos las siguientes relaciones reflejadas como hechos PROLOG, donde recogemos la información sobre profesores, departamentos y las asignaturas dadas por los profesores de una universidad:

```
departamen(Codigodepartamento,Nombredepartamento)
profesor(Codigoprofesor,Nombreprofesor,Codigodepartamento)
perfilprof(Codigoprofesor,Asignatura)
```

podríamos traducir:

```
departamento(X)  -> departamen(X,_)
profesor(X,Y)    -> profesor(X,_,Y)
darclase(X)      -> perfilprof(X,_).
```

Con estas definiciones y traducciones quedaría la forma objetivo:

```
sino(cada(departamen(X,_),cada(profesor(X,_,Y,_),perfilprof(X,_)))),
```

que sería un predicado directamente ejecutable.

5 Mejoras del Formalismo

En un futuro nos planteamos adecuar todas las técnicas desarrolladas por Michael McCord para implementar las MLGs, a una estructura más sistemática siguiendo el modelo LOQUI, desarrollado en [2].

Creemos que de la gramática se debe obtener la representación intermedia. Pensamos que esto supondría colocar en los sitios adecuados los operadores y terminales lógicos que hiciesen falta.

También pensamos que habrá que profundizar en la representación del conocimiento, y tratar de aplicar este formalismo al tratamiento de textos amplios.

Referencias

[1] PATRICK SAINT-DIZIER AND STAN SZPAKOWICZ, 1190 Logic programming, Logic grammars, Language processing en Logic and logic grammars for language processing Ellis Horwood series in artificial intelligence

[2] JEAN-LOUIS BINOT, 1990 Traitement de la langue naturelle, logique et programmation en Approche logique de líntelligence artificielle, V.3 Ed. DUNOD

[3] MICHAEL MCCORD, 1990 Natural Language ptocessing in Prolog. en Knowledge Systems an Prolog 2ª ed. Ed. Addison-Wesley Publishing Company, Inc.

[4] FERNANDO PEREIRA and STUART M. SHIEBER, 1987. Prolog and natural-language analysis. Ed. CSLI.

También se han usado los siguientes libros y artículos para la elaboración del presente trabajo:

Traitement de la langue naturelle, logique et programmation en Approche logique de líntelligence artificielle, V.1 y V. 2 Ed. DUNOD

MCCORD M. C. (1982). Using slots and modifiers in logic grammars for natural language, Artificial Intelligence, Vol. 18, 327, 367.

DAHL, V. and M. C. MCCORD (1983). Treating coordination in logic grammars, American Journal of Computational Linguistics, Vol 9, 69-91.

MCCORD M. C. (1984). Semantic Interpretation for the EPISTLE system. Proccedings of the Second International Logic Programming Conference, Uppsala, Sweden, 65-76.

MCCORD, M. C. (1985). Modular Logic Grammars, Proccedings 23rd Annual Meeting of the Association for Computational Linguistics, Chicago, 104-117.

MCCORD, M. C. (1985). Desing of a Prolog-based machine translation system, Proccedings of the Third International Logic Programming Conference, London.

MCCORD, M. C. (1985). Semantics of Natural Language: LFL, Presentation at IBM Europe Institute, Oberlech, Austria.

MCCORD, M. C. (1985). LMT: A Prolog-based machine translation system (extended abastract) In: Nirenburg, S. ed., Proceedings of the Conference on Theorical and Methodological Issues in Machine Translation of Natural Languages, Colgate University, Hamilton, 179-182.