IMPLEMENTANDO HPSG EN ALE.

Salvador Climent Xavier Farreres
climent@goliat.upc.es farreres@lsi.upc.es
Departament de Llenguatges i Sistemes Informàtics
Facultat d'Informàtica
Universitat Politècnica de Catalunya

Abstract.
A core of a grammar of Spanish in HPSG including a solution to deal with verbal subcategorisation alternances avoiding to list different senses in the lexicon is built in ALE. Computational efficacy of HPSG and capabilities of ALE are tested by the way.

Compendio
Por el presente trabajo se describe el desarrollo de un núcleo de gramática del español en HPSG implementada en ALE. La gramática incluye una solución destinada a evitar la proliferación de entradas léxicas causada por la polisemia estructural del verbo. A partir de ello, se examinan las capacidades de ALE y de una aproximación ortodoxa a HPSG para el procesamiento del lenguaje natural.

1 Introducción

El principal objetivo de este trabajo ha sido diseñar un núcleo significativo de la gramática del español en HPSG (Head-Driven Phrase Structure Grammar) (Pollard and Sag 1987,1992) e implementarlo en ALE (Attibute Logic Engine) (Carpenter, 1992b), y a partir de ello examinar (i) cuan apropiada y manejable es una aproximación ortodoxa a HPSG para el procesamiento del lenguaje natural (PLN); y (ii) cuan adecuado es ALE para el desarrollo de una aplicación de dicho tipo.

Asimismo, el desarrollo de la gramática incluye el de un sistema destinado a evitar la proliferación de entradas léxicas causada por la polisemia estructural del verbo (atribución de múltiples esquemas de subcategorización a una entrada verbal). La solución habitual a dicho problema consiste en postular una entrada léxica para cada esquema de subcategorización posible -ver, p.e. Briscoe and Copestake (1991)-; en nuestra opinión ello no es completamente satisfactorio ya que por una parte implica una ampliación considerable del lexicón -ya de por sí un objeto muy complejo en teorías de corte lexicalista como HPSG-, y por otra no da cuenta de la capacidad de los hablantes para usar una misma palabra en contextos diferentes modulando así su significado básico.

La solución que se postula consiste en tomar como realización -y por tanto entrada léxica- básica de cada verbo aquella que comporta un esquema de subcategorización mínimo, el cual es definido como obligatorio para la entrada, tratándose como opcionales el resto de argumentos que la palabra puede tomar.

En las § 2 y 3 del presente trabajo presentamos, respectivamente, HPSG y ALE, ; en §4 describimos la gramática desarrollada, incluyendo la propuesta para el tratamiento de la polisemia estructural verbal; en §5 detallamos las conclusiones a las que hemos llegado respecto a las capacidades y adecuación de HPSG y ALE; y finalmente en el apéndice se incluye la implementación o *código* del sistema.

2 HPSG

HPSG es un formalismo suficientemente conocido, ya que se ha convertido en un punto de referencia obligatorio en los más recientes desarrollos de PLN, por lo que no entraremos a describirlo detalladamente, limitándonos a señalar sus rasgos más significativos.

HPSG es una gramática de unificación basada en estructuras de rasgos (ERs) los cuales se selaccionan de entre un conjunto bien definido de rasgos y valores. Es un formalismo declarativo y basado en la estructura superficial de la oración. Como formalismo de unificación, combina estructuras de rasgos en base a la operación de unificación y la relación de subsunción; en consecuencia, la buena formación de frases depende y es entendida en términos de compatibilidad de ERs. En HPSG las ERs admiten operaciones lógicas, pueden ser agrupadas en listas o conjuntos, y pueden ostentar valores que sean consecuencia de una función. El objetivo final de un análisis en HPSG es aglutinar en una única ER toda la información (morfo/fonológica, sintáctica, semántica, pragmática) contenida en una cadena de palabras.

Los componentes de una gramática HPSG son: signos (léxicos y frasales); reglas de combinación de signos; pricipios (universales y dependientes de la lengua) que afectan a la aplicación de todas las reglas; reglas (dependientes de la lengua) de orden superficial; y reglas léxicas.

A diferencia de otras teorías sintácticas (GB, GPSG) que conceden mayor relevancia al sistema de reglas, en HPSG la mayor parte de la información está codificada en el léxico. Es básico en la filosofía de HPSG el poder dar cuenta de la mayor cantidad posible de frases bien formadas del lenguaje mediante un numero mínimo de reglas o "esquemas de combinación", sobre los cuales actúan un asimismo limitado conjunto de principios.

2.1 Signos. Los signos en HPSG pueden ser léxicos o frasales. Se permite, y de hecho es un mecanismo descriptivo fundamental, el que distintos rasgos compartan un mismo valor. Los signos léxicos y los frasales difieren en cuanto a estructura básica únicamente en el atributo <DTRS>, el cual es privativo de los signos frasales. <DTRS> refleja el proceso de combinación de signos que da lugar al signo frasal expresado y es el equivalente al árbol de análisis en otros formalismos.

El resto de rasgos que describen un signo son <PHON> -forma fonológica[1]- <Q-STORE> -información sobre cuantificadores y su alcance- y <SYNSEM>, el cual a su vez está descrito por medio de los rasgos <CAT> -sintaxis- <CONT> -información lógico-semántica y de concordancia- y <CONTEXT> -pragmática-.

La formulación original de HPSG describía <CAT> por medio de los rasgos <HEAD> -núcleo- y <SUBCAT>, cuyo valor era la lista de signos[2] (simples o complejos) subcategorizados por el signo descrito, ordenada según el criterio tradicional de oblicuidad. Dicha información operaba en la construcción del signo de forma similar al mecanismo de cancelación en Gramáticas Categoriales. Así, por ejemplo, el valor <SUBCAT> de un nombre sería el de un determinante, y el de un verbo la lista de sus argumentos. Sin embargo, problemas teóricos de diversa índole aconsejaron a Pollard y Sag (1993) el desglose de dicho rasgo en otros de tipo más especializado: <SUBJ> -sujetos-, <COMPS> -complementos-, <ADJ> -adjuntos-, <MODS> -restricciones selectivas de elementos no nucleares, como p.e. adjetivos- y <SPEC> -especificadores-.

---

[1] En nuestra implementación, dado que se trata de un analizador de texto escrito, forma ortográfica.

[2] De hecho la lista de rasgos <SYNSEM> de dichos signos.

2.2 Principios. En cuanto a los principios, éstos son de aplicación general sobre toda regla de la gramática, actuando en consecuencia como condiciones de buena formación de éstas. Los hay universales y dependientes del lenguaje, siendo los principales principios universales los de núcleo (Head Feature Principle, HFP) y de subcategorización (Subcategorisation Principle, SP)[3].

HFP, siguiendo el espíritu de la Teoría X' en gramática generativa, define que un sintagma es proyección de su núcleo; lo cual en HPSG implica que el rasgo <HEAD> de un signo frasal y el rasgo <HEAD> del signo que funciona como su núcleo sintáctico necesariamente comparten el mismo valor.

SP es responsable de la correcta cancelación de categorías en la construcción de un signo complejo a partir de un núcleo y sus complementos. Establece que el valor de la lista de subcategorización de un nodo padre es igual al de la lista de subcategorización de su hijo núcleo menos aquellos elementos de la lista correspondientes al resto de hijos no nucleares. Como ya hemos apuntado, el mecanismo corresponde a la idea intuitiva de cancelación de categorías en Gramáticas Categoriales. En consonancia con el desdoblamiento al que se ha hecho mención anteriormente del rasgo <SUBCAT> en otros rasgos más especializados, SP ha debido ser redefinido de acuerdo con la nueva situación; sin embargo la idea básica del principio continúa siendo la misma.

2.3 Reglas. Por lo que a las reglas se refiere, HPSG las utiliza de tres tipos: léxicas (RLs), de precedencia lineal (RPLs) y esquemas de dominancia inmediata (EDIs). Las primeras son generalizaciones que actúan sobre el lexicón evitando así redundancias en su desarrollo y mantenimiento; un ejemplo típico de RL es la derivación morfológica , p.e. formación de plurales u otras formas flexivas, a partir de lemas o raíces.

Los EDI son esquemas de buena formación de signos complejos; p.e. la formación de un signo SN a partir de un signo N y un determinante. Es importante tener en cuenta que los EDI, en principio, no se postulan para dar cuenta de la formación de signos específicos, sino de tipos de signos que comparten ciertas características comunes; con ello se persigue que un número reducido de reglas dé cuenta de un amplia abanico de fenómenos de composición. Las descripciones de los EDI son parciales y subespecificadas; la configuración final de un signo surgirá de la interacción de los EDI con otros tipos de reglas, los principios y la estructura de tipos -más la acción sobre todos ellos del mecanismo de unificación-.

Las RPLs, finalmente, son dependientes de la lengua y establecen el orden superficial final de las cadenas formadas mediante los EDI.

3 ALE

ALE (Attibute Logic Engine) es un formalismo para el análisis del lenguage natural cuyos elementos más simples son ERs. Está basado en un sistema de tipos con herencia múltiple monotónica sobre el que actúan ciertas restricciones, siendo las principales las siguientes:

(i) cada ER pertenece a un tipo predefinido

(ii) cada ER debe estar *bien tipificada*, en dos sentidos:

(ii.a) sólo los rasgos *apropiados* para el tipo pueden ser incluidos en la ER

---

[3] No hacemos referencia aquí al resto de principios -que afectan a aspectos de tipo semántico, de cuantificación, etc.- puesto que en nuestro trabajo nos limitamos a la sintaxis.

(ii.b) sólo las ERs que pertenecen a tipos *apropiados* para un rasgo pueden ser valores de dicho rasgo.

Aunque ALE permite herencia múltiple, impone ciertas restricciones a la estructura de tipos a fin de que la herencia múltiple pueda ser controlada. Una primera restricción consiste en que cuando un grupo de tipos *padre* que compartan más de un *hijo*, debe existir un único *hijo* interpuesto que los subsuma a todos ellos. Una segunda restricción establece que cada nombre de rasgo puede ser introducido una única vez y en un único tipo de la jerarquía, y que únicamente los *hijos* de ese tipo tienen acceso a su definición. Dichas restricciones permiten la validación automática de la estructura de tipos y sienta las bases para que la misma pueda compilarse, permitiendo así un considerable incremento de la velocidad de procesamiento.

A dicho respecto es asimismo básica la noción de *apropiabilidad*, según la cual cada tipo tiene un conjunto de atributos que le son adecuados. De esta forma puede saberse de antemano si dos estructuras son unificables atendiendo tan sólo a sus atributos, pudiendo deducirse así automáticamente el tipo que resultará de la unificación.

La *apropiabilidad*, junto con la compilación del sistema de tipos repercute en un gran ahorro de tiempo de análisis, al poder detectarse errores de forma rápida evitando así unificaciones innecesarias.

El sistema de tipos es el componente más importante del sistema, ya que todos los demás deben ser definidos de acuerdo con las restricciones que éste impone. Un tipo en ALE se define del siguiente modo:

```
type    sub [subtype1 ... subtypeN]
        intro [feature1:type1 ... featureM:typeM]
```

'intro' define la estructura del tipo, y 'sub' los subtipos de éste. Un tipo puede ser subtipo de más de un tipo; los rasgos que lo definen deben haber sido introducidos en el mismo tipo o en uno de sus tipos *padre* -en caso contrario la definición es errónea y el sistema no podrá ser compilado-. Los valores (type1 ... typeM) de los rasgos que definen el tipo (feature1 ... featureM) definen a su vez los posibles valores que puede adquirir el rasgo; dichos valores neesariamente deben haber sido definidos previamente en el sistema de tipos. Dichas restricciones aseguran que la unificación de dos tipos será a su vez un tipo del sistema.

En ALE pueden asimismo definirse macros, reglas léxicas, reglas gramaticales, entradas léxicas y predicados lógicos. Los macros son una forma de ahorrar *código* y se expanden cuando alguna regla los activa. Las reglas léxicas se usan, como en HPSG, para generar entradas léxicas derivadas a partir de entradas léxicas base, mientras que las reglas gramaticales son del tipo de reescritura. Los predicados lógicos -cláusulas de programación lógica *tipo Prolog*- pueden ser introducidos y utilizados en cualquier punto de las reglas, pudiendo así realizarse comprobaciones y computaciones de tipo no específicamente previsto por ALE.

4 La gramática

La gramática que aquí se presenta está restringida a la sintaxis y dirigida al análisis de frases declarativas. El sistema de tipos, sin embargo, cubre la práctica totalidad de la tipología declarada por Pollard y Sag (1987,1993) -ver *type hyerarchy* en el apéndice- en los capítulos de fundamentación de su formalismo; ello significa que el sistema de tipos que se ha implementado permite la extensión de la gramática hacia la cobertura de un amplio espectro de fenómenos gramaticales, como estructuras

interrogativas, subordinación, dependencias a larga distancia, semántica, etc. Con ello se ha pretendido que el sistema de tipos constituya no simplemente la base para el tratamiento de ciertos fenómenos sintácticos, sino un auténtico núcleo formal de HPSG -siempre dentro del marco *ortodoxo* establecido por Pollard y Sag (1987,1993)-.

La tipología de objetos <SYNSEM> y derivados, en cambio, ha sido desarrollada según nuestros criterios particulares, con el fin de procurar la cobertura de ciertos fenómenos de polisemia verbal de una forma que evite la sobrecarga del componente léxico.

4.1 Sistema de Tipos y Subcategorización Verbal. El sistema de tipos básico de HPSG ha sido extendido en consonancia con los tipos de verbos que se postulan, lo que ha implicado el consiguiente despliegue de tipos <SYNSEM>, <LOC> y <CAT>. Dicha información pudiera haber sido codificada de forma equivalente en macros de ALE, pero no se ha elegido dicha alternativa, por una parte por considerar que se apartaba del espíritu de HPSG -la cual no prevé la utilización de ese tipo de recursos-, y por otra a causa de las múltiples ventajas procedurales derivadas de la codificación de la información en ALE en el sistema de tipos en lugar de en otros módulos, siendo las principales el incremento de velocidad de procesamiento y la detección previa de inconsistencias.

La gramática se ha estructurado alrededor de seis tipos de verbos clasificados de acuerdo con sus posibilidades de subcategorización. La idea básica que subyace a dicha clasificación es la de que, en muchos casos, a pesar de que un verbo presente diferentes alternativas de diátesis, su significado básico es único, por lo que sería deseable que fuera codificado en una única entrada léxica.

Dicha hipótesis está avalada, por ejemplo, por experimentos psicolingüísticos realizados por Gleitman y Gillette (1994) en el ámbito del aprendizaje de primeras lenguas, los cuales parecen demostrar que los hablantes incorporan por una parte *palabras* a las que atribuyen significados (independientes de realización estructural), y por otra *tipos de estructuras*, en los que es posible -bajo ciertas condiciones- enmarcar las *palabras*, las cuales verán modulado su significado de acuerdo con la función que realicen en cada marco estructural.

De acuerdo con esa concepción, no se diría, por ejemplo, que existen verbos transitivos y verbos intransitivos, sino usos transitivos y usos intransitivos de un mismo verbo. Tomando en consideración los ejemplos (1), en (1a,b), "comer" presenta una alternancia transitiva/intransitiva; y en (1c,d), "comprar" se realiza en (1c) en una estructura con objeto directo y en (1d) en otra con objetos directo e indirecto.

(1) a. Juan comía.
b. Juan comía patatas.
c. Juan compró patatas.
d. Juan compró patatas para su hijo.

Ello implica habitualmente que el lexicón deberá contener (como mínimo) las correspondientes entradas independientes representadas en (2):

(2) a. COMER_1 <FN_{sujeto}>
b. COMER_2 <FN_{sujeto} , $FN_{obj.directo}$>
c. COMPRAR_1 <FN_{sujeto} , $FN_{obj.directo}$>
d. COMPRAR_2 <FN_{sujeto} , $FN_{obj.directo}$, $FP_{obj.indirecto}$>

Una variante a la simple codificación de entradas alternativas es la planteada por Briscoe y Copestake (1991) quienes postulan generarlas mediante reglas léxicas aplicadas a entradas *base*. Con ello se simplifica el proceso de escritura de entradas

léxicas y se da cuenta del hecho de que diferentes realizaciones superficiales de un mismo verbo corresponden a un único significado subyacente. Sin embargo, en términos de recursos computacionales, ello no repercutiría en un ahorro de espacio de programa en ALE, ya que en dicho sistema las reglas léxicas se aplican -y por ende se expande el lexicón- en una fase previa a la de ejecución; en consecuencia el lexicon al completo, expandido, ocupará en el momento de la ejecución como mínimo el mismo espacio que en el caso del simple listado de entradas alternativas[4].

En nuestra aproximación, sin embargo, cada verbo mantiene en el lexicón una única estructura que engloba la entrada *básica* y las expandidas. Se considera como estructura argumental básica del verbo su realización superficial mínima, la cual se declara como *obligatoria*, mientras que ulteriores argumentos que darán lugar a una estructura argumental más compleja se declararán como *opcionales* para la entrada verbal. Así, en un verbo como "comer" se entenderá como básico su uso intransitivo y como extendido el transitivo (3)[5]; por consiguiente, mientras su argumento sujeto será de realización obligatoria, el objeto directo se considerará opcional. Ello corresponde a la idea intuitiva de que tanto en el uso transitivo como en el intransitivo de "comer" se está expresando el concepto de que alguien está ingiriendo algo, con la diferencia de que, mientras en el uso intransitivo (2a) la cosa ingerida no está expresada, en el transitivo (2b) sí lo está[6].

(3) COMER <FN_{sujeto} , ($FN_{obj.directo}$)>

En el sistema de tipos la obligatoriedad u opcionalidad de complementos se ha implementado declarando los complementos obligatorios como valores del rasgo <COMPS> y los opcionales como valores de <ADJS> (véase en (4) la especificación esquemática del rasgo <SYNSEM> de un verbo tipo v_o_d, como "comer")[7].

(4)

synsem_vod
LOC: **loc_vod**
CAT: **cat_v_o_d**
HEAD: **verb**
SUBJ: **[NP_nominative]**
COMPS: **[empty_list]**

---

[4] Otro tratamiento sumamente interesante del fenómeno puede hallarse en Sanfilippo *et al.* (1994). En su aproximación las entradas verbales se asocian a tipos subespecificados cada uno de los cuales ramifica en subtipos de rango progresivamente más específico. La solución de Sanfilippo *et al.* (1994) produce un sistema sólido y eficaz pero tiene el inconveniente de que introduce un muy alto grado de complejidad en el sistema de tipos -ya de por sí complejo- de forma que en cada sucesiva ampliación o corrección de la gramática debe producirse una delicada reformulación del sistema de tipos, lo que repercute negativamente en la flexibilidad y facilidad de mantenimiento del sistema.

[5] En la notación utilizada en (3) los constituyentes opcionales se escriben entre paréntesis.

[6] Una solución en esta línea fue presentada recientemente por Pustejovsky (*en pr.*) en la conferencia inaugural del XI Congreso de la SEPLN, *The Role of Underspedification in Semantic Theory*. En su propuesta de representación léxica (independiente de formalismo gramatical) en la estructura argumental del verbo se definen *Arguments*, de realización obligatoria, y *Default Arguments* (equivalentes a nuestros optativos, de realización no obligatoria).

[7] No es ésta una solución definitiva ya que no es congruente con el espíritu de HPSG, que reserva el rasgo <ADJS> para adjuntos de un núcleo. Aunque ciertamente un adjunto -p.e. un adjetivo en un grupo nominal- es un elemento opcional, no todo elemento opcional es un adjunto -como en el caso que nos ocupa-. Sin embargo ALE no permite implementar la opcionalidad, por lo que hemos optado provisionalmente por la utilización del rasgo <ADJS> como repositorio heterogéneo de elementos opcionales, los cuales, de aparecer en la oración, serán recuperados e integrados en el análisis por la regla **adjs_head rule**. En ulteriores implementaciones, esta solución deberá ser sustituída por un tratamiento que permita distinguir la noción procedural de opcionalidad de la noción gramatical de constituyente adjunto.

ADJS: [NP_accusative]

Nuestra aproximación a la subcategorización resulta, como se ha dicho, en una clasificación de los verbos tomados en consideración en seis tipos, los cuales se muestran a continuación junto a un ejemplo prototípico de cada uno de ellos:

v_sa ("ser")	**v_o_o** ("morir")	**v_o_d** ("comer")
v_d_o ("golpear")	**v_d_o** ("comprar")	**v_di_o** ("dar")

La nomenclatura adoptada sigue la siguiente pauta mnemotécnica: en la cadena **v_X_Y** la posición X indica complementos obligatorios y la posición Y complementos opcionales; por otra parte **d** significa objeto directo, **i** objeto indirecto, **sa** sintagma adjetival, y **o** la lista vacía. En tipos intermedios del sistema se utiliza asimismo **x** para indicar -por oposición a **o**- cualquier tipo de complemento.

Así, **v_d_i** significa "un verbo que toma obligatoriamente un objeto directo, y opcionalmente un objeto indirecto"; y **v_o_d** "un verbo que puede opcionalmente tomar un objeto directo".

Cada tipo de verbo puede además, obviamente, tomar frases preposicionales (FPs) como complementos opcionales. Esta información no se codifica en la tipología verbal ya que es común a todos ellos y está declarada en un tipo de orden superior.

Los tipos verbales implementados admiten la formación de las oraciones prototípicas que se muestran en (5) -más posibles complementos circunstanciales/frases preposicionales-.

(5)
v_sa:	["ser"].	José es tonto.
v_o_o:	["morir"].	José murió.
v_o_d:	["comer"].	Juan comía.
		Juan comía patatas.
v_d_o:	["golpear"].	José golpeó a Juan.
v_d_i:	["comprar"].	Juan compró arsénico.
		Juan compró arsénico para José.
v_di_o:	["dar"].	Juan dió patatas a José.

4.2 Reglas y Principios. Las tres reglas sintagmáticas que operan en la gramática son **subj_head rule, comps_head rule** y **adjs_head rule**, las cuales, como sus nombres indican, combinan núcleos con, respectivamente, sus *sujetos*, *complementos* y *adjuntos*. Dichos términos deben ser tomados con precaución a fin de no ser maliterpretados en el contexto en el que nos movemos. Debe tomarse en consideración que:

(i).- como se ha apuntado en §4.1, los elementos opcionales de los verbos se incluyen en <ADJS>, aunque no sean adjuntos en su estricto sentido gramatical;

(ii).- el término *subject* no debe tomarse en el sentido habitual de la función gramatical "sujeto"; en HPSG no tan sólo las FNs-sujeto son *subjects* de la correspondiente FV, sino que también los determinantes son *subjects* de los Grupos Nominales (GNs) que determinan[8]; por tanto *subject* debe entenderse en HPSG aproximadamente como un complemento no subcategorizado de un núcleo.

Los principios que se han utilizado, son HFP y SP. El Principio Semántico (Semantics Principle), el cual pasa valores <CONT> de un hijo-núcleo a su nodo superior, se ha implementado formalmente, pero su utilización es limitada, ya que el

[8] En ulteriores desarrollos de HPSG se incluye el rasgo <SPEC> para el tratamiento de determinantes. En nuestra aproximación, sin embargo, mantenemos la configuración original por considerarla más simple y no problemática para el español.

único rasgo de <CONT> que se utiliza es <PARA:INDEX>, que establece concordancias de género, número y persona.

Los principios son precondiciones declarativas cuya acción se extiende sobre todos los aspectos de la gramática, en especial sobre las reglas. Desafortunadamente, no es posible implementar en ALE dicho tipo de procedimientos de alto nivel, por lo que se ha debido suplir dicha incapacidad del sistema por el método de incluir en cada una de las reglas los requisitos establecidos por los principios[9]. En estadios preliminares del desarrollo de la gramática exploramos la posibilidad de implementar los principios de HPSG utilizando dicho recurso, introduciendo en las reglas llamadas a cláusulas de programación lógica. Sin embargo dicho camino fue finalmente abandonado dado que por una parte no todos los principios podían ser implementados por ese método, y por otra se introducía *backtracking* de forma innecesaria, empeorando la respuesta en ejecución del sistema.

Las reglas implementadas son las siguientes:

- **subj_head rule** (6) , que combina FVs con sus FNs sujeto, y GNs con sus Determinantes. En ambos casos <COMPS> y <ADJS> deben haber sido previamente recuperados. El resultado de la aplicación de la regla es una frase saturada, es decir, en la que todas las listas de subcategorización son listas vacías. Obsérvese que el hecho de que los rasgos <head:HEAD> y <cont:CONT> pasen del hijo-núcleo al nodo superior (o signo resultante) asegura que sean observados los principios HFP y Principio Semántico. Ello, necesariamente, también sucederá en **comps_head rule** y **adjs_head rule**.

(6)

```
subj_head       rule
(phrase,
synsem:loc:(    cat:(   head:HEAD,
                        subj:[],
                        comps:[],
                        mark:[],
                        fill:[],
                        adj:[]),
                        cont:C),
dtrs:(  head_dtr:HEADDTR,
        subj_dtr:SUBJDTR))
===>
cat>
(SUBJDTR,synsem:SUBJ),

cat>
(HEADDTR,
synsem:loc:(cat:(       head:HEAD,
                        subj:[SUBJ],
                        comps:[]),
                        cont:C)).
```

- **comps_head rule** (7) combina verbos con complementos (obligatorios), y preposiciones -entendidas como núcleos de FPs- con FNs. El Principio de Subcategorización actúa eliminando un elemento de la lista de <COMPS> del núcleo al pasar dicha lista al signo resultante.

(7)

---

9 De igual forma, el orden oracional superficial, estipulado en HPSG por principios de ordenación de constituyentes dependientes de la lengua, se realiza en nuestra gramática por el simple método de convertir las reglas en dependientes del orden.

```
comps_head      rule
(phrase,
synsem:loc:(cat:(       head:HEAD,
                        subj:SUBJ,
                        comps:T,
                        mark:MARK,
                        fill:FILL,
                        adj:ADJ),
                cont:C),
dtrs:(  head_dtr:HEADDTR,
        comp_dtr:COMPDTR))
===>
cat>
(HEADDTR,
synsem:loc:(cat:(       head:HEAD,
                        subj:SUBJ,
                        comps:[COMP|T],
                        mark:MARK,
                        fill:FILL,
                        adj:ADJ),
                cont:C)),
cat>
(COMPDTR,synsem:COMP).
```

- **adjs_head rule** (8) combina verbos con *complementos opcionales*, y verbos u otras categorías con adjuntos (p.e. nombres con adjetivos).

(8)

```
adjs_head       rule
(phrase,
synsem:loc:(cat:(       head:HEAD,
                        subj:SUBJ,
                        comps:COMPS,
                        mark:MARK,
                        fill:FILL,
                        adj:T),
                cont:C),
dtrs:(  head_dtr:HEADDTR,
        adj_dtr:ADJDTR))
===>
cat>
(HEADDTR,
synsem:loc:(cat:(       head:HEAD,
                        subj:SUBJ,
                        comps:COMPS,
                        mark:MARK,
                        fill:FILL,
                        adj:[ADJ|T]),
                cont:C)),
cat>
(ADJDTR,synsem:ADJ).
```

Las tres reglas anteriores, aplicadas junto al sistema de tipos y el léxico, permiten la formación de oraciones generables por la siguiente gramática libre de contexto[10]:

(9)

O --> FN FV

[10] Las categorías precedidas por * son terminales; X/Y indica posibilidades alternativas; las categorías entre paréntesis son opcionales; las categorías X_a y X_d están, respectivamente, en caso acusativo y dativo.

FN --> *nombre_propio
FN --> *det *nombre (*adjetivo) (FP)
FP --> *prep FN
FV --> *v_sa *adj
FV --> *v_o_o (FP)
FV --> *v_o_d (FN_a/FP_a) (FP)
FV --> *v_d_o FN_a/FP_a (FP)
FV --> *v_d_i FN_a (FP_d) (FP)
FV --> *v_di_o FN_a/FP_a FP_d (FP)

La concordancia entre nodos viene dada por el mecanismo de unificación, via especificaciones establecidas en las entradas léxicas.

4.3 El Lexicón. En nuestra implementación el léxico es meramente representativo de las clases de vocabulario cubiertas. Un conjunto notable de palabras puede ser representado a partir del mismo simplemente atribuyendo a lemas macros de ALE. Por ejemplo, la entrada del determinante "el" será la de (10), expandiéndose en fase de precompilación según el *macro* **det(X,Y,W,Z) macro** -ver *macro* en el apéndice-.

(10) el ---> @det(def,masc,sing,ter).

No se ha explotado el recurso de las Reglas Léxicas: se ha implementado simplemente una regla de formación de plural como prueba de su funcionamiento en ALE.

5. Conclusiones

El resultado tangible de este trabajo, la gramática para frases declarativas del español nucleadas por seis diferentes tipos de verbo aceptando cada uno de ellos diferentes esquemas de subcategorización, más la estructura de tipos que sirve de base a dicha gramática y a futuras ampliaciones de la misma, puede ser examinado en el apéndice.

En la gramática se ha desarrollado un sistema dirigido a evitar la proliferación de entradas léxicas causada por la polisemia estructural del verbo (atribución de múltiples esquemas de subcategorización a una entrada verbal), el cual se basa en considerar como entrada léxica básica de un verbo aquélla que ostenta el esquema de subcategorización mínimo, y declarando en ella como optativos los ulteriores complementos que pueda tomar.

Por otra parte, de la experiencia obtenida en la implementación en ALE de la noción ortodoxa de HPSG -no de una gramática de unificación similar a HPSG-, extraemos las conclusiones que se describen a continuación.

5.1 Dificultades de implementación de HPSG en ALE. La principal fuente de problemas de compatibilidad entre HPSG y ALE es debida al hecho de que, mientras la primera está basada en principios, el segundo es un formalismo de reescritura.

En primer lugar, como ocurriría con cualquier otro formalismo de reescritura, ALE opera desde el punto de vista de la construcción de estructuras, mientras que el funcionamiento de HPSG está basado en la validación de las mismas. Por consiguiente, en muchos casos no puede existir una relación directa entre reglas de HPSG y reglas de ALE.

Por otra parte, como se ha apuntado anteriormente, varios Principios operan simultáneamente en HPSG sobre cada uno de sus esquemas o reglas. Dado que ALE no permite una codificación independiente de dichos principios, es preciso implementar sus efectos regla a regla, lo cual es una fuente potencial de problemas e

inconsistencias al diseñar, ampliar o modificar la gramática.

5.2 Sobre ALE. En general los resultados ofrecidos por ALE son ampliamente satisfactorios, principalmente por lo que refiere a velocidad de ejecución. Ello es debido a que las fuertes restricciones impuestas sobre la declaración del sistema de tipos permite su precompilación.

También es digna de ser destacada la flexibilidad del formalismo en cuanto permite el libre intercalado de cláusulas Prolog en cualquier punto de cualquiera de sus módulos.

Por otra parte, aunque en ALE las ventajas superan a los inconvenientes, éstos tampoco dejan de existir. La propia fuerte tipificación que posibilita la precompilación del sistema de tipos y consiguiente aumento de la velocidad de ejecución -lo cual significa la principal innovación teórica de Carpenter (1992a) y la más notable capacidad de ALE-, en muchos casos supone un inconveniente, dada la obligatoriedad de declarar como tipos del sistema todos los posibles valores que puedan aparecer. A menudo dichos valores pueden ser irrelevantes o también innumerables, por lo que sería aconsejable poder declarar un único tipo flexible que los subsumiera e hiciera innecesario su detalle exhaustivo; piénsese por ejemplo en la realización ortográfica de los signos/palabras, en dónde no es posible simplemente declarar un tipo *string* de libre realización, o en las listas, en las que es preciso declarar como tipo tantas posibles clases de composiciones de las mismas como puedan darse.

Otra dificultad es la del mantenimiento. Cualquier modificación del sistema de tipos en un componente del sistema repercute inevitablemente en otros múltiples puntos del *código*, los cuáles deben ser editados y modificados a su vez de forma congruente, sin que sea posible automatizar o facilitar dicho proceso. Ello puede no resultar un grave inconveniente en sistemas *de juguete*, pero se convierte en difícilmente tratable en sistemas mínimamente complejos, más teniendo en cuenta que en su desarrollo suelen intervenir, simultánea o diacrónicamente, diferentes investigadores y no es razonable pensar que todos ellos *tendrán en la cabeza* todos los detalles del sistema.

Finalmente, consideramos que, tal como puede hacerse en las reglas y en las entradas léxicas, sería deseable que pudieran establecerse co-indizaciones en el sistema de tipos, ya que aumentaría notablemente la expresividad del formalismo y facilitaría su desarrollo.

5.3 Sobre HPSG. HPSG es un formalismo muy potente para el tratamiento de restricciones lingüísticas. La naturaleza altamente estructurada de sus signos permite codificar en ellos todo tipo de requisitos de concordancia sintáctica o semántica, de forma que tan sólo análisis extremadamente precisos pasarán el *filtro* de la unificación. En la gramática que se ha implementado, a diferencia de lo que suele ocurrir en otro tipo de formalismos, se obtiene un único análisis de cada oración no ambigua.

De igual forma, la complejidad de los signos en combinación con la estricta tipificación del sistema, permite que toda la información recuperable de un análisis sea expresada y aglutinada en una única estructura, evitando así tener que mantener niveles separados de representación para cada tipo de información (p.e. sintaxis y semántica) y ponerlos en correspondencia.

Por contrapartida, la misma la misma complejidad de los signos que es fuente de las ventajas antes mencionadas lo es también de inconvenientes: desarrollar una gramática HPSG supone manejar un lexicón complejo e introducir mucha información

en el mismo. Aunque ésta es una estrategia beneficiosa a largo plazo, no lo es si lo que se pretende es el desarrollo rápido de gramáticas de amplia cobertura. En otras palabras, antes de trabajar con HPSG es preciso plantearse muy seriamente el muy debatido problema del *cuello de botella*[11] de la adquisición léxica.

Como consideraciones finales, debemos exponer nuestro convencimiento de que HPSG es esencialmente un instrumento para la investigación teórica en lingüística, más que un formalismo realmente adecuado para el desarrollo de aplicaciones. Para hacer HPSG realmente manejable creemos que es preciso *relajar* algunos de sus requisitos teóricos, como por ejemplo el de mantener un número muy limitado de reglas a la par que se sobrecarga la estructura de tipos (es más fácil modificar o ampliar una gramática trabajando sobre un conjunto de reglas que sobre una telaraña de tipos); o el de discriminar los signos mediante concatenaciones *profundas* de pares atributo-valor de forma que para recuperar valores es preciso recorrer largos *paths*[12] al tiempo que, para cada valor final, es preciso declarar en el sistema de tipos un nuevo tipo para cada etapa del *path* . A este respecto, sería aconsejable *allanar* los signos, es decir, convertirlos en estructuras atributo-valor con un nivel de indentación menor; quizá ello causara ciertas disfunciones con respecto a la teoría, pero convertiría las aplicaciones basadas en HPSG en más legibles y fáciles de desarrollar.

Referencias

BADIA T. (1993) Inicios de una gramática para el español en ALEP, un formalismo de unificación. Actas del X Congreso de la SEPLN. Santiago de Compostela.

BRISCOE EJ. & COPESTAKE A. (1991) Sense extensions as lexical rules. Acquilex WP num. 22.

CARPENTER B. (1992a) The Logic of Typed Feature Structures. Cambridge University Press. New York.

CARPENTER B. (1992b) ALE: The Attribute Logic Engine User's Guide. Version ß. CMU. Pittsburg.

COPESTAKE A. (1992) The Representation of Lexical Semantic Information. Tesis Doctoral. Cognitive Science Research Papers 280. University of Sussex.

GLEITMAN X. & GILLETTE X. (1994) The Role of Syntax in Verb Learning...

POLLARD C. & SAG I. (1987) Information-Based Syntax and Semantics. CSLI, Stanford.

POLLARD C. & SAG I. (1993) Head-Driven Phrase Structure Grammar. (Draft version). CSLI. Stanford.

PUSTEJOVSKY J. (*en pr.*.) The Generative Lexicon.

SANFILIPPO A., BENKERIMI K. & DWEHUS D. (1994) Virtual Polysemy. COLING 94 Proceedings. Kyoto.

STEFANOVA M., ter STAL W. (1993) A comparison of ALE and PATR; practical experiences. Twente Workshop on Language Technology.

---

11 Traducción literal del término original inglés *bottleneck*,, expresivo de la dificultad de tratar con la necesidad de codificación de un volumen de datos extraordinariamente extenso.

12 Término inglés por "camino": concatenación de los rasgos hasta llegar al valor final.

6. Apéndice

% type hierarchy %

```
bot sub [sign, synsem, loc, cat, head, list,
nonloc, pform, bool, cont, context, para,
index, psoa, con_struc, pers, num, gen, case,
vform, ontologia, morfo].
sign sub [word,phrase]
 intro [phon:list,synsem:synsem].
word sub []
  intro [].
phrase sub []
  intro [dtrs:con_struc].
con_struc sub [head_struc,conj_struc].
conj_struc sub [].
head_struc sub
[head_subj_struc,head_comp_struc,head_mark
_struc,head_fille r_struc, head_adj_struc]
  intro [head_dtr:sign].
head_subj_struc sub []
  intro [subj_dtr:sign].
head_filler_struc sub []
  intro [filler_dtr:sign].
head_comp_struc sub []
  intro [comp_dtr:sign].
head_mark_struc sub []
  intro [marker_dtr:sign].
head_adj_struc sub []
  intro [adj_dtr:sign].
nonloc sub []
  intro [inher:wh_str,to_bind:wh_str].
cont sub [nom_obj,verb_obj].
context sub []
  intro [backgr:list].
verb_obj sub [].
nom_obj sub [expl_obj,ref_obj]
  intro [para:para].
expl_obj sub []
  intro [para:expl].
ref_obj sub []
  intro [para:ref,restr:list].
psoa sub [].
para sub [expl,ref]
  intro [index:index].
  expl sub [].
  ref sub [npro,pron].
    npro sub [].
pron sub [ppro,ana].
  ppro sub [].
  ana sub [refl,recp].
    refl sub [].
    recp sub [].
index sub []
  intro [pers:pers,num:num,gen:gen].
pers sub [prim,seg,ter].
num sub [sing,plur].
gen sub [masc,fem].
head sub [subst,funct].
subst sub [noun,verb,adj,prep]
  intro [prd:bool].
adj sub []
  intro [mod:synsem].
det sub [].
funct sub [det,mark].
mark sub [].
noun sub [propi,comu]
  intro [case:case].
  propi sub [].
  comu sub [].
case sub [nom,ac,dat,geni,voc,abl].
  nom sub [].
  ac sub [].
  dat sub [].
  geni sub [].
  voc sub [].
  abl sub [].
prep sub []
  intro [pform:pform].
pform sub [].
verb sub [verbsa]
  intro [vform:vform].
verbsa sub []
  intro [prd:true].
vform sub [fin,nonfin].
  fin sub [].
  nonfin sub [].
sn sub [].
prim sub [].
seg sub [].
ter sub [].
sing sub [].
plur sub [].
masc sub [].
fem sub [].
```

% tipología. cat %

```
cat sub [cat_subst,cat_funct]
  intro
[head:head,subj:list,comps:list,mark:list,fill:l
ist,adj:list].
cat_subst sub
[cat_noun,cat_verb,cat_adj,cat_prep].
cat_funct sub [cat_det,cat_mark].
cat_noun sub [cat_propi,cat_comu]
  intro [head:noun].
cat_propi sub []
  intro
[head:propi,subj:e_list,comps:e_list,mark:e_li
st,fill:e_lis t,adj:e_list].
cat_comu sub []
  intro
[head:comu,subj:list_1_det,comps:e_list,adj:li
st_1_adj,mark :e_list,fill:e_list].
cat_verb sub [cat_v_x_x,cat_v_sa]
  intro [head:verb,subj:list_1_noun].
cat_v_x_x sub
[cat_v_o_x,cat_v_d_x,cat_v_di_o].
cat_v_o_x sub [cat_v_o_o,cat_v_o_d]
  intro [comps:e_list].
cat_v_o_o sub [] % morir
  intro [adj:e_list].
```

```
cat_v_o_d  sub [] % comer
  intro [adj:list_1_subst].
cat_v_d_x  sub [cat_v_d_o,cat_v_d_i]
  intro [comps:list_1_subst].
cat_v_d_o  sub [] % golpear
  intro [adj:e_list].
cat_v_d_i  sub [] % comprar
  intro [adj:list_1_prep].
cat_v_di_o  sub [] % dar
  intro [comps:list_2_subst_prep].
cat_v_sa  sub []
  intro [comps:list_1_adj,head:verbsa].
cat_adj  sub []
  intro
[head:adj,subj:e_list,comps:e_list,adj:e_list,m
a rk:e_list,fill:e_list].
cat_prep  sub []
  intro
[head:prep,subj:e_list,adj:e_list,mark:e_list,fi
ll:e_list].
cat_det  sub []
  intro
[head:det,subj:e_list,comps:e_list,adj:e_list,m
ark:e_list,f ill:e_list].
cat_mark  sub []
  intro [head:mark].

% typología synsem  %

synsem  sub [synsemsubst,synsemfunct]
  intro [loc:loc,nonloc:nonloc].
synsemsubst  sub
[synsemnoun,synsemverb,synsemadj,synsempr
ep]
  intro [loc:locsubst].
synsemfunct  sub [synsemdet,synsemmark].
synsemnoun  sub [synsempropi,synsemcomu]
  intro [loc:locnoun].
synsempropi  sub []
   intro [loc:locpropi].
synsemcomu  sub []
  intro [loc:loccomu].
synsemverb  sub [synsemvxx,synsemvsa].
synsemvxx  sub
[synsemvox,synsemvdx,synsemvdio].
synsemvsa  sub []
  intro [loc:locvsa].
synsemvox  sub [synsemvoo,synsemvod].
synsemvoo  sub []
  intro [loc:locvoo].
synsemvod  sub []
  intro [loc:locvod].
synsemvdx  sub [synsemvdo,synsemvdi].
synsemvdo  sub []
  intro [loc:locvdo].
synsemvdi  sub []
  intro [loc:locvdi].
synsemvdio  sub []
  intro [loc:locvdio].
synsemadj  sub []
  intro [loc:locadj].
synsemprep  sub []
  intro [loc:locprep].
synsemdet  sub []
  intro [loc:locdet].
synsemmark  sub []
  intro [loc:locmark].
%
% typología. loc  %
%

loc  sub [locsubst,locfunct]
  intro [cat:cat,cont:cont,context:context].
locsubst  sub
[locnoun,locverb,locadj,locprep]
  intro [cat:cat_subst].
locfunct  sub [locdet,locmark].
locnoun  sub [locpropi,loccomu]
  intro [cat:cat_noun].
locpropi  sub []
  intro [cat:cat_propi].
loccomu  sub []
  intro [cat:cat_comu].
locverb  sub [locvxx,locvsa].
locvxx  sub [locvox,locvdx,locvdio].
locvox  sub [locvoo,locvod].
locvoo  sub []
  intro [cat:cat_v_o_o].
locvod  sub []
  intro [cat:cat_v_o_d].
locvdx  sub [locvdo,locvdi].
locvdo  sub []
  intro [cat:cat_v_d_o].
locvdi  sub []
  intro [cat:cat_v_d_i].
locvdio  sub []
  intro [cat:cat_v_di_o].
locvsa  sub []
   intro [cat:cat_v_sa].
locadj  sub []
  intro [cat:cat_adj].
locprep  sub []
  intro [cat:cat_prep].
locdet  sub []
  intro [cat:cat_det].
locmark  sub []
  intro [cat:cat_mark].

% listas  %

list  sub [e_list,nelist].
e_list  sub [].
nelist  sub[list_1,list_2]
   intro [hd:bot,tl:list].
list_1  sub
[list_1_noun,list_1_prep,list_1_det,list_1_ad
j,list_1_subst ]
   intro [tl:e_list].
list_1_noun  sub []
  intro [hd:synsemnoun].
list_1_prep  sub []
  intro [hd:synsemprep].
list_1_det  sub []
  intro [hd:synsemdet].
list_1_adj  sub []
  intro [hd:synsemadj].
```

```
list_1_subst  sub []
  intro [hd:synsemsubst].
list_2  sub [list_2_subst_prep]
  intro [tl:list_1].
list_2_subst_prep  sub []
  intro [hd:synsemsubst,tl:list_1_prep].
bool  sub [true,false].
true  sub [].
false  sub [].

% ontology        %

ontologia
sub[verbo,jose,def,hombre,tonto,patata,prepos
].
jose  sub [].
hombre  sub [].
def  sub [].
tonto  sub [].
patata  sub [].
verbo  sub [v_x_x,v_sa].
v_sa  sub [ser].
v_x_x  sub [v_o_x,v_d_x].
v_o_x  sub [v_o_o,v_o_d].
v_o_o  sub [morir].
v_o_d  sub [comer].
v_d_x  sub [v_d_o,v_d_i,v_di_o].
v_d_o  sub [golpear].
 v_d_i  sub [comprar].
v_di_o  sub [dar].
morir  sub [].
comer  sub [].
golpear  sub [].
comprar  sub [].
dar  sub [].
ser  sub [].
prepos  sub [a].
a  sub [].

% lexicon        %

el  ---> @ det(def,masc,sing,ter).
la  ---> @ det(def,fem,sing,ter).
las  ---> @ det(def,fem,plur,ter).
los  ---> @ det(def,masc,plur,ter).
hombre  ---> @ comu(hombre,masc,sing).
golpea  ---> @ verb_d_o(golpear,sing,ter).
come  ---> @ verb_o_d(comer,sing,ter).
muere  ---> @ verb_o_o(morir,sing,ter).
compra  ---> @ verb_d_i(comprar,sing,ter).
da  ---> @ verb_di_o(dar,sing,ter).
es  ---> @ verb_sa(ser,sing,ter).
jose  ---> @ propi(jose,masc).
tonto  ---> @ adj(tonto,masc,sing).
tontos  ---> @ adj(tonto,masc,plur).
patata  ---> @ comu(patata,fem,sing).
a  ---> @ preposicion(a).

% macros        %

nom  macro
  (word,
  synsem:synsemnoun).

comu(Ph,Gen,Num)  macro
  (word,
  phon:[Ph],
  synsem:  (synsemcomu,
    loc:(  cat:  (subj:[@
syn_det(Gen,Num,_)],
          adj:[@ syn_adj(Gen,Num)]),
       cont:
       (ref_obj,
       para:  (npro,
          index:(gen:Gen,num:Num))))).
syn_adj(Gen,Num)  macro
  (loc:  (locadj,
    cont:  (ref_obj,
      para:
        (npro,
         index:(gen:Gen,num:Num))))).

adj(Ph,Gen,Num)  macro
  (word,
  phon:[Ph],
  synsem:(synsemadj, @
syn_adj(Gen,Num))).

syn_det(Gen,Num,Pers)  macro
  (loc:  (locdet,
    cont:  (ref_obj,
      para:  (npro,
index:(gen:Gen,num:Num,pers:Pers))))).

det(Ph,Gen,Num,Pers)  macro
  (word,
  phon:[Ph],
   synsem:(synsemdet, @
syn_det(Gen,Num,Pers))).

propi(Ph,Gen)  macro
  (word,
  phon:[Ph],
  synsem:  (synsempropi,
loc:cont:(ref_obj,para:(npro,index:(gen:Gen,n
um:sing))))).

verb_d_o(Ph,Num,Pers)  macro
  (word,
  phon:[Ph],
  synsem:  (synsemvdo,
    loc:  (cat:  (head:vform:fin,
subj:[loc:(locnoun,cont:para:(AG,index:(num:
Num,pers:Pers)) )],
         comps:[loc:cont:para:PAC],
         mark:[],
         fill:[]),
       cont:  (verb_obj,
         agente:AG,
         paciente:PAC)))).

verb_o_d(Ph,Num,Pers)  macro
  (word,
  phon:[Ph],
  synsem:  (synsemvod,
    loc:  (cat:  (head:vform:fin,
```

```
subj:[loc:(locnoun,cont:para:index:(num:
Num,pers:Pers))],
          mark:[],
           fill:[]),
        cont:verb_obj))).

verb_o_o(Ph,Num,Pers)   macro
   (word,
  phon:[Ph],
  synsem:  (synsemvoo,
     loc:  (cat:  (head:vform:fin,
subj:[loc:(locnoun,cont:para:index:(num:Num
,pers:Pers))],
           mark:[],
           fill:[]),
         cont:verb_obj))).

verb_d_i(Ph,Num,Pers)   macro
   (word,
  phon:[Ph],
  synsem:  (synsemvdi,
     loc:  (cat:  (head:vform:fin,
subj:[loc:(locnoun,cont:para:index:(num:Num
,pers:Pers))],
           mark:[],
           fill:[]),
         cont:verb_obj))).

verb_di_o(Ph,Num,Pers)   macro
   (word,
  phon:[Ph],
  synsem:  (synsemvdio,
     loc:  (cat:  (head:vform:fin,
           adj:[],
subj:[loc:(locnoun,cont:para:index:(num:Num
,pers:Pers))],
           mark:[],
           fill:[]),
         cont:verb_obj))).

verb_sa(Ph,Num,Pers)   macro
   (word, .
  phon:[Ph],
  synsem:  (synsemvsa,
     loc:  (cat:  (head:vform:fin,
subj:[loc:(locnoun,cont:para:index:(num:Num
,pers:Pers))],
           mark:[],
           fill:[]),
         cont:verb_obj))).

preposicion(Phon)   macro
   (word,
  phon:[Phon],

synsem:(synsemprep,loc:cat:comps:list_1_nou
n)).

% lexical rules   %

plural_n   lex_rule
@ comu(Phon,Gen,sing)
**>
@ comu(Phon,Gen,plur)
morphs
 (X,V) becomes (X,V,s) when vocal(V),
(X,C) becomes (X,C,es).

% rules          %
```

[ver las reglas en §4.2 del texto principal]

```
% clauses        %

vocal([a]).
vocal([e]).
vocal([i]).
vocal([o]).
vocal([u]).
```