RESOLUCIÓN DE LA EXTRAPOSICIÓN A IZQUIERDAS CON LAS GRAMÁTICAS DE UNIFICACIÓN DE HUECOS

A. Ferrández; J. Peral; P. Martínez; M. Saiz; R. Romero
{antonio, jperal, patricio, max, romero}@dlsi.ua.es
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
URL: http://www.dlsi.ua.es

Resumen

En este trabajo presentamos aportaciones a la resolución de la extraposición a izquierdas que ocurre en las oraciones de relativo por medio de un formalismo lógico basado en restricciones, que es una extensión de las *Gramáticas de Cláusulas Definidas (Definite Clause Grammar DCG)* [17], al cual llamamos *Gramáticas de Unificación de Huecos (Slot Unification Grammar SUG*), a causa de las *estructuras de huecos* o *slot structures* generados automáticamente por el analizador, donde se incluyen las restricciones necesarias para la resolución de la extraposición. Estas estructuras de huecos incluyen de forma automática toda la información morfológica, sintáctica y semántica necesaria para resolver la extraposición. En este artículo presentamos un trabajo complementario a los trabajos [9][11][16] dentro del proyecto de resolución de fenómenos lingüísticos utilizando formalismos basados en restricciones para textos no restringidos.

1. Introducción

El fenómeno lingüístico de la *extraposición a izquierdas* ocurre cuando un subconstituyente de un constituyente se desplaza a la izquierda de éste y es sustituido por un elemento en la oración. Hasta ahora, muchos trabajos de lingüística computacional han tratado este problema desde distintos puntos de vista, centrándose en el tratamiento sintáctico.

En este artículo proponemos un tratamiento y resolución de la extraposición a izquierdas basado en restricciones; para ello incluimos una serie de restricciones a las SUG, que en un paso posterior al análisis, nediante una estructura llamada de huecos, resuelven la extraposición. En este paso posterior y mediante la estructura de huecos se recoge el conjunto

de información que refleja las restricciones para resolución de distintos fenómenos lingüísticos (extraposición, elipsis, anáfora, etc). Además proponemos mediante esta estructura un mecanismo para la obtención de la fórmula lógica asociada a la frase de entrada. Consideramos relevantes estas aportaciones por poder ser orientada a textos no restringidos.

2. El problema de la extraposición

Según la definición de Pereira [18], la extraposición a izquierdas ocurre en una oración cuando un subconstituyente de un constituyente que forma parte de la oración, se representa por otro a la izquierda del que esté incompleto.

De toda la amplia variedad de expresiones que se engloban dentro de la extraposición nos centraremos en la extraposición en las oraciones de relativo. Particularmente estudiaremos los casos en los que la oración de relativo desempeña la función de un adjetivo (modificar a un sustantivo) y el pronombre relativo que aparece en esta oración es *que*.

Si consideramos el conjunto de reglas gramaticales de la Figura 1, entonces obtendremos el árbol de derivación de la frase *la casa que Juan construyó tiene ventanas*, mostrado en la Figura 2.

O → SN, SV	*O_relativo → Pronombre_rel, O*
O → SNRel, SV	*SV → Verbo, SN*
SN → Nompropio	*SV → Verbo, SNRel*
SN → Nombre	*SV → Verbo, SPrep*
SN → Det, Nombre	*SPrep → Prep, SN*
SNRel → SN, O_relativo	*SPrep → Prep, SNRel*

Figura 1

En la Figura 2, un subconstituyente X que forma parte de un constituyente Y ha sido extrapuesto a la izquierda Z. Existe un pronombre relativo que es la marca que hace referencia al subconstituyente extrapuesto. En este ejemplo se observa que el subconstituyente extrapuesto posicionalmente acompaña al verbo y desempeña la función de objeto directo en la oración de relativo. Hay que destacar que tanto la posición como la función que desempeña el subconstituyente extrapuesto varía de un caso a otro.

Podemos ver otro ejemplo en la Figura 3. Como se puede apreciar en ambas figuras, el subconstituyente extrapuesto ocupa en cada caso posiciones y categorías gramaticales diferentes. Tenemos que abordar, pues, una serie de problemas. En primer lugar, detectar el subconstituyente extrapuesto (Z); después, detectar la posición y función que ocupa en la oración de relativo.

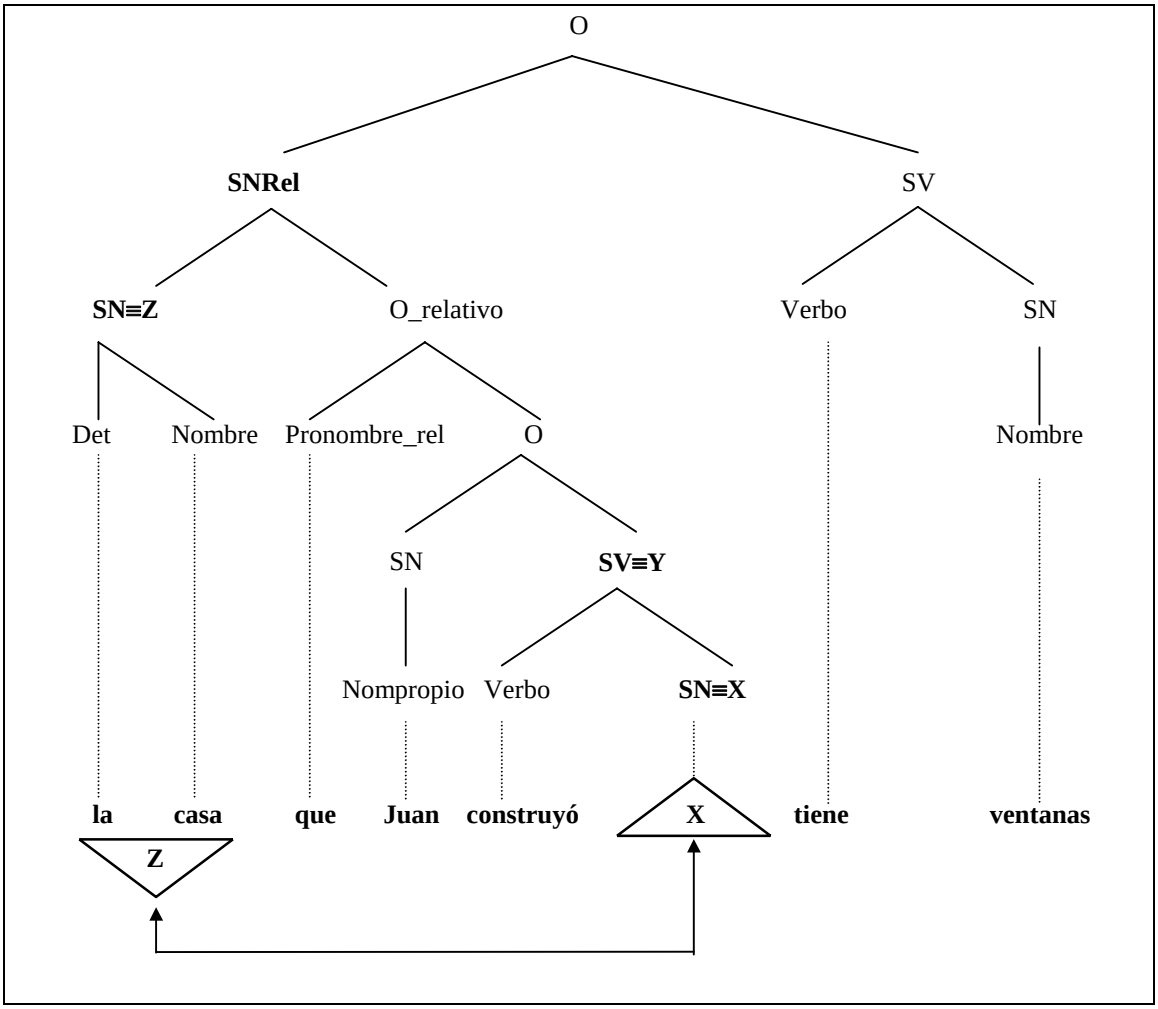

Figura 2

Para resolver el primer problema nos encontramos que en todas las oraciones de relativo que estamos analizando, el pronombre relativo siempre hace referencia al constituyente anterior (Sintagma Nominal) por la propia definición de la gramática. Por lo tanto, el pronombre actúa de marca indicando que el Sintagma Nominal inmediatamente anterior a él es el subconstituyente extrapuesto.

Para detectar la posición que ocupa el subconstituyente extrapuesto en la oración de relativo deberemos analizar la oración de relativo y averiguar si hay algún hueco que se ha de rellenar obligatoriamente y está vacío (por ejemplo: verbos a los que acompañan obligatoriamente un SN, etc.). Si falta únicamente un hueco obligatorio ya hemos encontrado tanto la posición como la categoría gramatical del subconstituyente extrapuesto. Si por el contrario, hay más de un hueco obligatorio que falta por rellenar utilizaremos restricciones de tipo semántico y morfológico para determinar cuál de todos ellos se rellena con el subconstituyente extrapuesto.

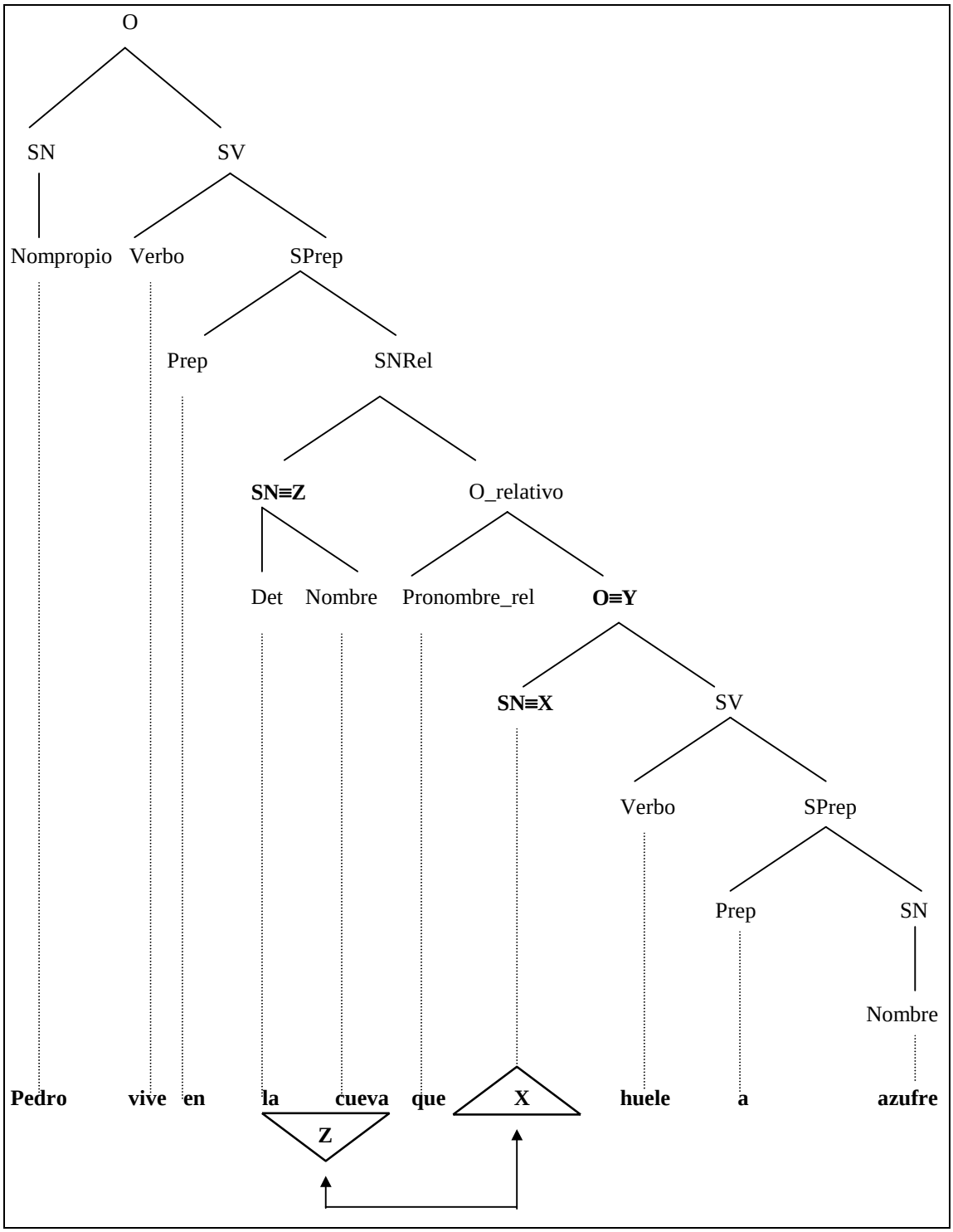

Figura 3. Ejemplo de extraposición: Pedro vive en la cueva que huele a azufre

3. Antecedentes

Las *Gramáticas de Metamorfosis* (*MG*) introducidas por A. Colmerauer [1] son el primer formalismo que permite la descripción de las reglas de la gramática a partir de variaciones sintácticas de Prolog y que hacen transparente al usuario la manipulación de la cadena de entrada durante la entrada, incluyendo nuevas características. Posteriormente, Pereira y Warren

[17] definen las DCG como una restricción de las MG donde la parte izquierda de las reglas tienen un único símbolo lógico no terminal.

Trabajos independientes en programación lógica han empleado las DCG como la fundamentación de muchos formalismos basados en restricciones, tales como las *Gramáticas de Extraposición* (*XG*), *Gramáticas de Cabeceras* (*HG*) y *Gramáticas Discontinuas* (*DG*).

F. Pereira [18] propone las XG con una nueva característica: los saltos entremezclados en la parte izquierda de la regla serán rutinariamente reescritos en orden secuencial en la parte más a la derecha de la regla: *marcador_relativo, salto (x), traza → pronombre_relativo, salto(x).* En la parte izquierda de la regla hay un constituyente vacío, la traza, que ocupa el hueco que ha dejado un constituyente que se ha perdido. El marcador indica que un constituyente a su derecha contiene una traza. Se puede ver como que el constituyente, en cuyo lugar está ahora la traza, ha sido extrapuesto a la izquierda, y su nueva posición se representa con el marcador. Una XG válida podría ser la mostrada en la Figura 4.

```
sentencia → sintagma_nominal, sintagma_verbal.
sintagma_nominal → determinante, nombre, relativo.
sintagma_nominal → traza.
relativo → [ ].
relativo → marcador_relativo, sentencia.
marcador_relativo ... traza → pronombre_relativo.
```

Figura 4.

Todas las reglas, a excepción de la última, son libres de contexto. La última representa la extraposición como una cláusula relativa simple. Se puede decir que las XG describen el fenómeno de la extraposición a izquierdas con potencia por los siguientes motivos:

- Son una extensión de las DCG que pueden ser interpretadas como formalismos gramaticales independientemente de su traducción a cláusulas definidas.
- Proporcionan una descripción simple de la extraposición a izquierdas y de sus restricciones: las restricciones de la extraposición pueden expresarse usando algunas herramientas artificiales como las cadenas de encorchetado, es decir, cadenas de inicio y fin que encierran un subcomponente.
- Se pueden comparar en eficiencia con las DCG cuando se ejecutan en PROLOG.

Por contra, la introducción de restricciones por medio del encorchetado de las frases extrapuestas en las XG de Pereira generan gramáticas y derivaciones complejas y de difícil comprensión. Además, tal y como apunta el mismo Pereira, la conexión entre el formalismo de las XG y las estrategias de análisis tradicionales no es una tarea trivial, lo que complica su implementación.

V. Dahl [3] propone una generalización de las XG, conocidas como DG (llamadas primitivamente Gramáticas de Huecos). Estas gramáticas fueron implementadas por V. Dahl y M. McCord [7] aplicándolas a la resolución del problema del tratamiento de la coordinación en

un compilador conocido como SYNAL. V. Dahl junto a H. Abramson [6], y posteriormente F. Popowich [20], hacen un estudio de conclusiones sobre este compilador. El SYNAL es incrementado por Dahl y Saint-Dizier [8] añadiéndole la característica de los mecanismos de restricción con el propósito de automatizar las restricciones lingüísticas.

V.Dahl [4] generaliza el uso de las DG aplicándolas a otros problemas del procesamiento del lenguaje entre los que se encuentra el problema de la extraposición a izquierdas en las oraciones de relativo.

Las DG son una generalización de las XG donde los símbolos no identificados pueden ser repuestos arbitrariamente (sin necesidad de hacerlo al final de la cadena como ocurre en XG). Una discontinuidad o salto es una subcadena que se separa y se repone sin analizarse, para ser analizada en su nueva localización.

Las DG añaden las siguientes características:

- Permiten una reposición libre de las discontinuidades: en el ejemplo de sintagma nominal: “el hombre con cuyo tío Juan partió” se produce una doble extraposición a izquierdas que sería difícil de expresar en XG: “El hombre [Juan partió con [el] tío [del hombre]]”. En las DG una regla simple podría capturarlas: *np(x), salto(y), prep, det, salto(z), prep(de), np(x)* $\rightarrow$ *np(x), prep, [cuyo], salto(z), salto(y).*
- Permiten otras formulaciones discontinuas equivalentes: los saltos pueden reescribirse al final de la parte derecha o en cualquier otro orden cambiando el sentido de la gramática y aceptar los mismos lenguajes obteniendo, en algunos casos, mejores costes.
- Permiten interaccionar entre diferentes reglas de discontinuidad: las XG sólo permitían múltiples saltos si éstos eran independientes o estaban anidados totalmente uno dentro de otro. Las DG permiten cualquier tipo de salto aunque interaccionen entre sí.
- No necesitan los símbolos artificiales de las cadenas encorchetadas para expresar restricciones.
- Obtienen gramáticas más simples sin perder su eficiencia.
- Permiten tratar lenguajes naturales donde las palabras siguen una ordenación libre de palabras (atendiendo más a su declinación que al orden establecido, como el Latín o el Sánscrito).
- Tratan eficientemente el problema de la libre ordenación de constituyentes que pueden o no estar presentes dependiendo de otros constituyentes. Es el caso de los verbos que requieren sólo un objeto directo mientras otros requieren también un indirecto.

Sin embargo, las DG se quedan en el tratamiento sintáctico de la extraposición, sin llegar a tratar de forma clara el problema de las restricciones en este movimiento. V. Dahl tratará posteriormente un subconjunto de las DG, las *Gramáticas Discontinuas Estáticas* (*SDG*) [5], en las que sólo se mueven los constituyentes definidos sobre las discontinuidades mientras los saltos permanecen fijos. Estas SDG proporcionan el filtro necesario para el tratamiento de las restricciones del lenguaje que propone Chomsky en su Teoría de Reacción - Ligamiento [2].

Otros formalismos han tratado el problema de la extraposición basándose en gramáticas no concatenantes, como las HG de Pollard [19] y múltiples derivaciones de las *Gramáticas de Árboles Contiguos* (*TAG*) de Kroch y Joshi [13]. En concreto, las HG se basan en el principio de que la cabecera de un constituyente es el elemento clave de este constituyente. El formalismo de las HG divide un constituyente en tres componentes: un contexto izquierdo, un terminal cabecera y un contexto derecho. En la reescritura de una regla HG, las tres partes de un constituyente pueden colocarse de distinta forma a la de su construcción. Estas gramáticas tienen como inconveniente que para cada tipo de constituyente hay una cabecera única, es decir, un elemento que se puede mover. Por esto no sería posible tratar varios problemas de extraposición dentro de un mismo constituyente de forma simultánea.

Recientemente, Groenink [12] define las *Gramáticas de Movimiento de Literales* (*LMG*) como solución al fenómeno de la extraposición a través de un mecanismo que permite un desplazamiento top - down de información sintáctica, basándose en las gramáticas HG, XG y TAG que ya han usado métodos de desplazamiento sintáctico.

Las LMG separan el tratamiento del lenguaje natural en dos fases: una fase de análisis según el tratamiento tradicional libre de contexto y una segunda fase de eliminación de ambigüedad mediante matching, a diferencia de métodos anteriores que lo hacen en una fase única. Así, las LMG proporcionan una forma clara y sencilla de tratar el fenómeno de la extraposición siendo capaz de modelar varios movimientos de una vez.

Sin embargo, como indica Groenink, no se han estudiado los rendimientos de las LMG en grandes gramáticas. Sería necesario hacer pruebas en gramáticas completas para determinar exactamente la eficiencia del método.

En cualquier caso, los formalismos anteriores tratan, de forma más o menos clara, el problema sintáctico de la extraposición pero en ningún caso se trata el problema del análisis semántico. El método que proponemos en este artículo nos introducirá a una evaluación semántica posterior.

4. Solución propuesta: Slot Unification Grammar (SUG)

4.1 SUG versus DCG

Una DCG [17] puede ser definida como una cuádrupla *(NT,T,P,S)*, donde *NT* es un conjunto finito de símbolos no terminales, *T* es un conjunto finito de símbolos terminales disjunto con *NT*, *P* es un conjunto finito de pares $\alpha \rightarrow \beta$ donde $\alpha \in NT$, $\beta \in (T \cup NT)^* \cup$ *{llamadas a procedimientos}*, y *S* es el conjunto de símbolos iniciales de la gramática. Los elementos de *P* se llaman *reglas de producción*..

SUG añade a la definición previa de las DCG que sus reglas de producción se denotan como α*++>*β, y cada subconstituyente de β puede omitirse de la oración si se escribe entre el operador << y >>. A estos subconstituyentes de β que pueden omitirse les llamaremos *no terminales débiles*, y a los demás que son obligatorios en la oración, *no terminales fuertes*. Además de las reglas de producción SUG tales como α*++>*β, se añaden lo que llamamos *hechos SUG*. Estos hechos tienen solamente el primer miembro de la regla de producción, o sea α, siendo α o *coordinated*, *juxtaposition*, *fusion*, *basicWord* o *isWord*. Estas variaciones se hacen con la intención de eliminar toda la recomputación innecesaria y las reglas de recursividad a izquierdas (por ejemplo: $A \rightarrow A\ B$ o $A \rightarrow B$, $B \rightarrow A\ C$). Se puede encontrar una explicación más detallada de SUG en [9][16][11], donde se desarrolla en mayor profundidad el modo en que se resuelve la coordinación, yuxtaposición y el acceso al diccionario de palabras.

Podemos ver en la Figura 5 un ejemplo de la gramática SUG que emplearemos para resolver los casos de extraposición a izquierdas que trataremos en este artículo, y su gramática DCG equivalente. En este ejemplo se puede apreciar la disminución del número total de reglas de la gramática de SUG respecto a DCG.

Reglas DCG	***Reglas SUG***
o → snRel, sv. *o → sn, sv.* *o → sv.* *snRel → sn, o_relativo.* *sn → det, nombre.* *sn → nombre.* *o_relativo → pronombre_rel, o.* *sv → verbo, sn, sp.* *sv → verbo, snRel, sp.* *sv → verbo, sn.* *sv → verbo, snRel.* *sv → verbo, sp.* *sv → verbo.* *sp → prep, sn.* *sp → prep, snRel.*	*o ++> <<snRel>>, sv.* *snRel ++> sn, <<o_relativo>>.* *sn ++> <<det>>, nombre.* *o_relativo ++> pronombre_rel, o.* *sv ++> verbo, <<snRel>>, <<sp>>.* *sp ++> prep, snRel.*

Figura 5.

4.2 Estructura de huecos devuelta por el analizador SUG

En las secciones anteriores hemos descrito SUG como una extensión de DCG. Hay traductores que convierten las reglas DCG en cláusulas Prolog, añadiendo dos argumentos extras a cada símbolo que no está entre llaves o corchetes. Cada regla DCG por ejemplo $o \rightarrow sn, sv$, es traducida a Prolog del modo siguiente: *o(L1,L3) :- sn(L1,L2), sv(L2,L3),* donde *L1, L2* y *L3* son las listas de palabras de la frase a analizar.

Nosotros hemos desarrollado un traductor de reglas SUG a cláusulas Prolog. Este traductor ha sido ejecutado bajo SICStus Prolog 2.1 y Arity Prolog 5.1. Dada, por ejemplo, la regla SUG *o(Numero) ++> sn(Numero), sv(Numero),* se verá traducida a Prolog del siguiente modo: *o(Numero,EHo,L1,L3) :- sn(Numero,EHsn,L1,L2) sv(Numero,EHsv,L2,L3).* Aquí se puede apreciar que se le ha añadido tres argumentos extras a cada subconstituyente de la regla que no está entre llaves o corchetes. Los dos últimos argumentos corresponden a las mismas listas de palabras mencionadas para las DCG. El primero corresponde a lo que llamamos *estructura de huecos (EH)* o *slot structure*.

Esta *EH* almacena la información sintáctica, morfológica y semántica de cada constituyente. Toda esta información se almacena de manera automática por el traductor SUG tal y como se muestra en la Figura 6. Cada *EH* está formada por una estructura con functor el nombre del constituyente (*sn, sv,*…). Su primer argumento corresponde a otra estructura con nombre *conc* la cual incluye todos los argumentos de la regla gramatical (información morfológica y semántica: *Numero, Genero, TipoSemantico*). El segundo corresponde a la variable de la fórmula lógica final del constituyente. Y los restantes argumentos corresponden a las estructuras de huecos de sus subconstituyentes.

En esta *EH* el traductor dejará como variables Prolog sin instanciar ("_") aquellos huecos que correspondan a componentes opcionales que no han sido analizados en la frase, información de especial interés a la hora de resolver la extraposición. Otra ventaja de estas variables sin instanciar, es la facilidad con que nos permiten recuperar los elementos extrapuestos o elididos de un constituyente. Esta recuperación se hará en un sólo paso por medio de la unficación: *Hueco=ElementoExtrapuesto*.

La información semántica corresponde a restricciones semánticas que han sido incluidas en el análisis sintáctico. Nosotros hemos aplicado el método IRSAS [14][15] para incorporar estas restricciones semánticas en el análisis. Con la intención de simplificarlos, a menos que sea necesario, en los restantes ejemplos de estructuras de huecos, no se incluirán los campos correspondientes a la estructura *conc* ni la variable *X*.

Regla SUG:

sn(Numero, Genero, TipoSemantico) ++> — *% Slot Structure: SN*
<<determinante(Numero,Genero)>>, — *% Slot Structure: DET*
nombre(Numero, Genero, TipoSemantico), — *% Slot Structure: N*
<<sp(TipoSemantico)>>. — *% Slot Structure: SP*

SLOT STRUCTURE: *(sn: "un perro")*

*SN=sn(D,**X**, DET,N,SP), donde:*
D=conc(singular, masculino, noHumano)
X *la variable de la fórmula lógica final del constituyente*
DET=determinante(DD, X, WD), donde:
DD= conc(singular)
*WD es el átomo Prolog '**un**'. Se corresponde con la palabra de la oración.*
N=subst(DN, X, WN), donde:
DN= conc(singular,masculino,noHumano)
*WN es el átomo Prolog '**perro**'. Se corresponde con la palabra de la oración.*
SP es una variable Prolog sin instanciar ya que es un componente opcional que no ha aparecido en SN.

FORMULA LOGICA: *(oración: "un perro ladró")*

*existe(**X**,perro(**X**),ladrar (**X**))*

Figura 6. Estructura de huecos del sintagma nominal "un perro".

A continuación vamos a intentar clarificar el proceso en el que se obtiene la fórmula lógica y cómo se interrelaciona con el módulo de resolución de la extraposición. Este proceso se ha resumido en la Figura 7. En primer lugar se realiza un análisis sintáctico de la frase y obtenemos su estructura de huecos, *EHo*. Después, se aplica el módulo de resolución de la extraposición a izquierdas utilizando *EHo*. La solución consistirá en una nueva estructura de huecos *EHo'* en la que se ha eliminado la extraposición. Este *EHo'* se utilizará para obtener la fórmula lógica final.

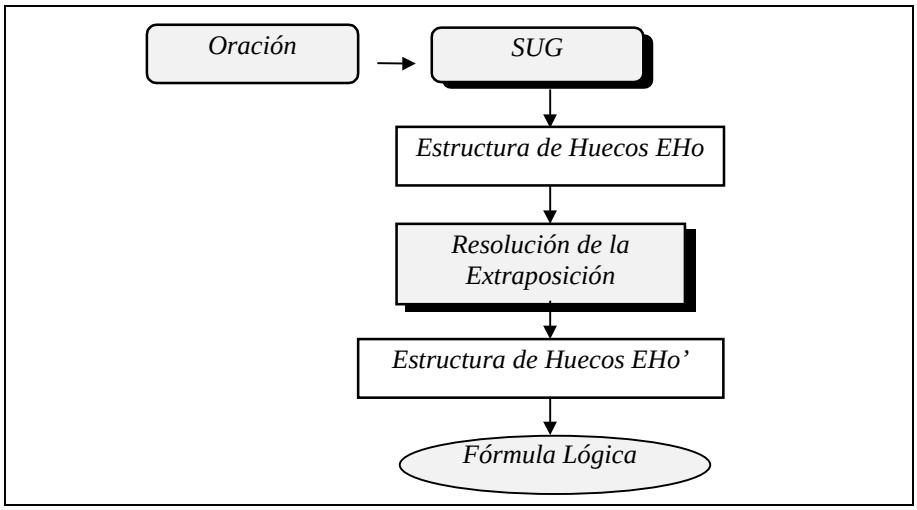

Figura 7.

Es importante el destacar que esta técnica de resolución nos permite obtener sistemas de procesamiento del lenguaje natural muy modulares, en los que las reglas gramaticales y el cálculo de la fórmula lógica son independientes de los módulos de resolución de problemas lingüísticos como son la extraposición de constituyentes o la anáfora

(problema que se resolvería en el mismo punto que la extraposición tal y como se explica en [10]).

4.3 Algoritmo para la resolución de la extraposición a izquierdas

El algoritmo que a continuación proponemos está basado en el proceso de análisis descrito en la Figura 7 y trabajará sobre la estructura de huecos descrita en la Figura 6, la cual se obtendrá al aplicar la gramática SUG de la Figura 5.

Para entender con mayor facilidad este algoritmo, puede resultar útil la Figura 8 en la que se detalla la estructura de huecos correspondiente a un sintagma nominal con extraposición a izquierdas.

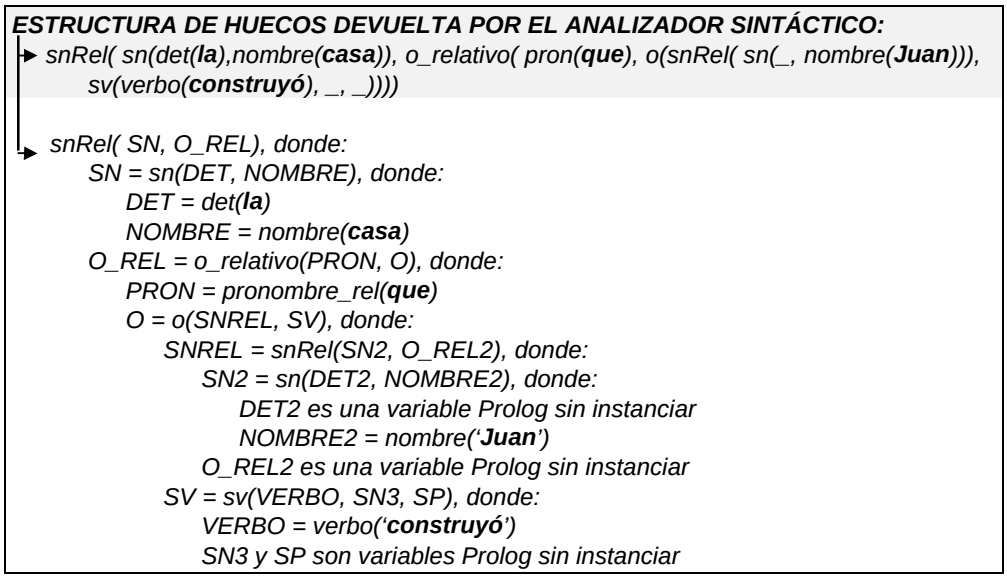

Figura 8. Estructura de Huecos del sintagma nominal: "La casa que Juan construyó".

El algoritmo de la Figura 9 analizará cada constituyente de nombre *snRel*, comprobando que el hueco correspondiente a la oración de relativo haya sido rellenado (situación indicadora de la presencia de una extraposición a izquierdas), en cuyo caso, se activará el proceso de resolución. La recuperación del elemento extrapuesto será sencilla puesto que se corresponderá con el hueco de nombre *sn* y se realizará por medio de la unificación en la variable *SNExtr*. La manera en la que se obtiene el hueco que ha de ocupar *SNExtr* sería mediante la comprobación de los huecos que no han sido rellenados dentro de *SV* con función de sujeto y objeto directo (*SN2* y *SN3*). En caso de que ambos huecos estén llenos, no podríamos resolver la extraposición, por lo que se produciría lo que hemos llamado *Error 1*. En cuanto a la solución final con la extraposición resuelta, sería la variable *EHo'* que almacenaría por unificación *O2*, en la que se ha recuperado el sintagma nominal extrapuesto (*SN_F=SN*), coordinada con el resto de la oración donde se ha sustituido *SNREL* (el sintagma nominal donde aparecía la extraposición) por *SNExtr* (el sintagma nominal que estaba extrapuesto). Es importante destacar que al

hacer esta última sustitución, la variable correspondiente a la fórmula lógica final que almacena esta estructura de huecos será la misma en *SN* que en *O2*, por lo que se consigue que en la fórmula lógica final se haga referencia al mismo sintagma nominal en ambos casos. Respecto al *Error 2*, se puede apreciar cómo se deja para ese momento la comprobación de la presencia de estos huecos obligatorios (objeto directo o indirecto para verbos transitivos), ya que podrían ser ocupados al resolver la extraposición. En la Figura 10 y la Figura 12 podemos ver la aplicación de este algoritmo desarrollada paso a paso.

Proceso de análisis sintáctico de la oración. Se obtiene su estructura de huecos en ***EHo***
Por cada constituyente con nombre "snRel" de ***EHo***
    SNREL*=snRel(****SN****,****O_REL****)*
    Si el hueco correspondiente a O_REL ha sido rellenado *%nonvar(O_REL)*
        SNExtr*=****SN***
        O_REL*=o_relativo(PRON,****O2****)*
        O2*=o(****SN2****,****SV****)*
        SV*=sv(****VERBO****,****SN3****,SP)*
        Si ***SN2*** *y* ***SN3*** *están rellenos* *% nonvar(SN2), nonvar(SN3)*
            Error 1*: oración incorrecta*
        Si no
            SN_F *= hueco que no haya sido rellenado previamente (SN2 o SN3)*
            SN_F = SNExtr
            Si ***SNExtr*** *respecto a* ***VERBO*** *cumple las restricciones morfológicas (concordancia de número y persona) y semánticas en el hueco* ***SN_F***
                EHo' *=* ***O2*** *&* ***(EHo*** *sustituyendo* ***SNREL*** *con* ***SNExtr)***
Si no existe algún hueco obligatorio en ***EHo'***
    Error 2*: oración incorrecta por ausencia de huecos obligatorios en la frase*

Figura 9. Algoritmo de resolución de la extraposición a izquierdas con SUG.

En caso que ambos huecos (*SN2* y *SN3*) estén vacíos, la manera que tendríamos para seleccionar el hueco adecuado sería mediante las restricciones semánticas y morfológicas (número, persona y género) tal y como se muestra en la Figura 11. Esta comprobación de las restricciones, sería por medio de la estructura *conc* que queda almacenada de forma automática en cada constituyente de la gramática. En esta figura se ha supuesto que las reglas SUG empleadas incluyen los argumentos correspondientes a persona, número y tipo semántico (por ejemplo: *sn(Persona,Numero,Genero,TipoSemantico)* ++> *<<det(Numero,Genero)>>, nombre(Persona,Numero,Genero,TipoSemantico).*).

EHo = o(SNREL, SV)
SNREL = snRel(SN,O_REL)
*SN = sn(**X**, det(**la**), nombre(**casa**))* *% **X** será la variable de la fórmula lógica final*
SNExtr = SN
O_REL = o_relativo(PRON, O2)
*PRON = pron(**que**)*
O2 = o(SN2, SV)
*SN2 = snRel(sn(_,nombre('**Juan**')),O_REL2)* *% '_' indica que no ha aparecido el determinante*
O_REL2 está vacío ('_') *% Esto indica que aquí no hay extraposición*
SV = sv(VERBO,SN3, _) *% Aquí tampoco han aparecido el OD ni el OI*
*VERBO = verbo(**construyó**)*

SN3 está vacío (es una variable Prolog sin instanciar) mientras que SN2 ha sido rellenado:
el hueco que ha de ocupar SNExtr será el de SN3 (además cumple las restricciones semánticas y morfológicas respecto a VERBO):
SN3 = SNExtr

*EHo' = o(snRel(sn(_,nombre(**Juan**)),_),sv(verbo(**construyó**),sn(**X**,det(**la**),nombre(**casa**)),_))*

&

*o(sn(**X**,det(**la**),nombre(**casa**)),sv(verbo(**tiene**),snRel(sn(_,nombre(**ventanas**),_),_))*

Fórmula lógica final:
existe(X, casa(X) & construir(Juan, X), existe(Y, ventanas(Y) & tener(X,Y)))

Figura 10. Resolución de la extraposición: La casa que Juan construyó tiene ventanas

EHo = o(SNREL, SV)
SNREL = snRel(SN,O_REL)
*SN = sn(**conc(terc,sing,noHumano)**, **X**, det(**la**), nombre(**casa**))*
SNExtr = SN
O_REL = o_relativo(PRON, O2)
*PRON = pron(**que**)*
O2 = o(SN2, SV)
SN2 es una variable Prolog sin instanciar
SV = sv(VERBO,SN3, _) *% Aquí tampoco han aparecido el OD ni el OI*
*VERBO = verbo(**conc(prim,sing,acción(humano,noHumano))**, **construí**)*
SN3 es una variable Prolog sin instanciar

el hueco que ha de ocupar SNExtr será el de SN3 ya que:
No podría ser el de SN2 porque no concuerda en persona (terc ≠ prim) y además el verbo 'construir' prefiere en el sujeto un elemento semántico del tipo 'humano'
SN3 = SNExtr

*EHo' = o(_, sv(verbo(**construí**), sn(**X**, det(**la**), nombre(**casa**)), _))*

&

*o(sn(**X**,det(**la**),nombre(**casa**)),sv(verbo(**tiene**),snRel(sn(_,nombre(**ventanas**)), _),_))*

Figura 11. SN2 y SN3 están vacíos: La casa que construí tiene ventanas

EHo = o(SNREL1, SV1)
*SNREL1 = snRel(sn(_, nombre('**Pedro**')),O_REL1)* *% No hay extraposición ya que O_REL1 es una variable sin instanciar* *% O_REL1 está vacío*
SV1 = sv(VERBO1, _, SP)
*VERBO1 = verbo(**vive**)*
*SP = sp(prep(**en**), SNREL)*
SNREL = snRel(SN,O_REL)
*SN = sn(**X**, det(**la**), nombre(**cueva**))* *% **X** será la variable de la fórmula lógica final*
SNExtr = SN
O_REL = o_relativo(PRON, O2)
*PRON = pron(**que**)*
O2 = o(SN2, SV)
SN2 está vacío
SV = sv(VERBO, SN3, SP2)
*VERBO = verbo(**huele**)*
*SP2 = sp(prep(**a**), snRel(sn(_, nombre(**azufre**)), _))* *% Aquí tampoco hay extraposición*

Aquí tanto SN2 como SN3 están vacíos
el hueco que ha de ocupar SNExtr será el de SN2 ya que cumple las restricciones semánticas y morfológicas respecto a VERBO, y además está ocupado uno de los huecos de SV (el correspondiente a SP2)
SN2 = SNExtr

EHo' =
*o(sn(**X**,det(**la**),nombre(**cueva**)),sv(verbo(**huele**),_,*
*sp(prep(**a**),snRel(sn(_,nombre(**azufre**)),_))))*
&
*o(snRel(sn(_,nombre('**Pedro**')),_),sv(verbo(**vive**),_,*
*sp(prep(**en**),sn(**X**,det(**la**),nombre(**cueva**)))))*

Fórmula lógica final:
existe(X, cueva(X), existe(Y, azufre(Y), oler(X,Y)), vivir(Pedro,X)))

Figura 12. Ejemplo de extraposición: Pedro vive en la cueva que huele a azufre

5. Conclusiones

En el presente trabajo, se ha presentado un sistema que permite una solución modular de la extraposición a izquierdas en oraciones de relativo en las que la oración de relativo desempeña la función de adjetivo y el pronombre relativo es *que*. Este sistema incluye un formalismo gramatical, al que hemos llamado *Slot Unification Grammar*, debido a las estructuras de huecos que genera de forma automática el analizador. Este analizador lo hemos implementado en Prolog, y genera dicha estructura de huecos, la cual integra toda la información necesaria para la resolución de la extraposición. Sobre este sistema hemos propuesto un algoritmo y hemos mostrado su resolución en diversos ejemplos de extraposición.

Como trabajos futuros, se nos plantea la generalización del conjunto de reglas SUG empleadas para afrontar problemas de extraposición de diferentes tipos en oraciones coordinadas.

6. Referencias

[1] Colmerauer, A. *"Metamorphosis Grammars"*. Lecture Notes in Computer Science, Springer-Verlag, pp 133-189, 1978

[2] Chomsky, N. *Lectures on Government and Binding*, the Pisa Lectures, 2nd (revised) Edition. Foris Publications, Holland, 1982

[3] Dahl, V. *Discontinuous Grammars*. Computational Intelligence, 5(4): 161-179, 1989

[4] Dahl, V. *More on gapping grammars*. Proceedings International Conference on V Generation Computer Systems, Tokyo, 1984

[5] Dahl, V. *Static Discontinuity Grammars for Government-Binding Theory*. LCCR TR 88-22, Simon Fraser University, 1988

[6] Dahl, V.; Abramson, H. *On gapping grammars*. Proc. Second International Conference on Logic Programming, Uppsala, Sweden, 1984

[7] Dahl, V.; McCord, M. *Treating coordination in logic grammars*. American Journal of Computational Linguistics, 9, 69-91, 1983.

[8] Dahl, V.; Saint-Dizier, P. *Constrained Discontinuous Grammars a linguistically motivated tool for processing language*. TR LCCR 86-11, Simon Fraser University, 1986

[9] Ferrández, A.; Moreno, L.; Palomar, M. *"Un formalismo para el tratamiento gramatical de la coordinación: Gramática de Unificación de Huecos"*. Novatica, 115. 1995

[10] Ferrández, A.; Palomar, M.; Moreno, L. *"Slot Unification Grammar and anaphora resolution"*. Presentado a la ACL'97. Pendiente de aceptación. 1997

[11] Ferrández, A.; Palomar, M.; Moreno, L. *"Slot Unification Grammar"*. Presentado a la SEPLN'97. Pendiente de aceptación. 1997

[12] Groenink, A. *Literal Movement Grammars*. 1995.

[13] Kroch, A.S.; Joshi, A.K. *Analyzing Extraposition in a TAG*. Ojeda Huck, editor, Syntax and Semantics: *Discontinuos Constituents*. Acad Press, New York. 1986

[14] Moreno, L.; Andrés, F.; Palomar, M. *"Incorporar Restricciones Semánticas en el Análisis Sintáctico: IRSAS"*. Procesamiento del Lenguaje Natural n.12. 1992.

[15] Moreno, L.; Palomar, M. *"Semantic Constraints in a Syntactic Parser: Queries-Answering to Database"*. Database and Expert Systems Applications. Springer-Verlag. 1992.

[16] Palomar, M.; Ferrández, A.; Moreno, L. *"Aportaciones a la Resolución de la elipsis en la coordinación"*. Procesamiento del Lenguaje Natural, nº16, 1995.

[17] Pereira, F.; Warren, D. *"Definite Clause Grammars for Language Analysis- A Survey of de Formalism and a Comparison with Augmented Transition Networks"*. Artificial Intelligence, vol. 13. 1980

[18] Pereira, F.C.N. *"Extraposition Grammars"*. Computational Linguistics, 7(4), 1981

[19] Pollard, C.J. *Generalizad Phrase Structure Grammars, Head Grammars, and Natural Language*. Ph. D. thesis, Standford University. 1984

[20] Popowich, F.P. *Unrestricted gapping grammars*. Computational Intelligence Journal, vol 2, pp 28-53, 1986