

Un mecanismo de transferencia para LFG en DCG-Prolog

Gabriel Amores Carredano
CCL - UMIST / Universidad de Sevilla

ABSTRACT

Tanto las gramáticas lógicas como de unificación están siendo ampliamente utilizadas en actuales sistemas de Traducción Automática (TA). Se presenta un prototipo que integra ambas corrientes: Lexical Functional Grammar (LFG), implementado en Definite Clause Grammars (DCG) en Prolog. Tras describir el mecanismo de unificación y algunos aspectos del módulo de análisis, nos centramos en el módulo de transferencia inglés-español. Se describen los algoritmos de transferencia léxica y estructural y cómo se resuelven algunos problemas, comparándolo con otras propuestas. Se asume cierta familiaridad con LFG, DCG y Prolog.

1. Introducción.

Desde la aparición de la LFG [BRE82] y una definición de los postulados de las gramáticas basadas en unificación [SHI86], han sido numerosas las implementaciones que usan ambos formalismos para la TA y otras aplicaciones.

Varios de esos sistemas [EIS86], basan dicha implementación en las ventajas que Prolog ofrece para gramáticas libres de contexto por medio del formalismo DCG [PER80, PER87]. Estos sistemas adoptan un mecanismo de unificación por términos basado en la unificación de colas de listas no instanciadas [EIS86, GAZ89 y HIR88].

Dado el carácter libre de contexto de las gramáticas DCG y el uso de variables lógicas en Prolog [FIN87], algunos autores han sugerido la posibilidad de implementar una gramática LFG usando el formalismo DCG. Esta posibilidad se ve reforzada por el hecho de que las reglas gramaticales en LFG presuponen un isomorfismo entre el componente libre de contexto (*c-structure*) y la descripción que se ha de obtener en términos de funciones gramaticales (*f-structure*).

El prototipo que describimos a continuación trata de poner en práctica estas sugerencias en una aplicación de TA inglés-español. El objetivo final del sistema no es la demostración de la teoría lingüística LFG, sino el desarrollo de un prototipo de TA para *abstracts* de medicina. Esta decisión reduce las estructuras sintácticas que la gramática ha de describir, así como el tipo de información que se incluirá en la *f-structure*.

2. Unificación

La mayor parte de los mecanismos de unificación a los que nos hemos referido anteriormente son *destructivos*. Es decir, las dos estructuras que se han de unificar intercambian información hasta que sean idénticas una a la otra [GAZ89:235-9]. Son por tanto predicados de dos argumentos en Prolog.

Nosotros hemos optado por un mecanismo más costoso pero más eficiente a largo plazo, ya que puede ser utilizado tanto en el análisis como en la transferencia. Consiste en un predicado de tres argumentos en el que el resultado de unificar A y B es C, pero sin perder la información contenida en A y B.

Nuestras reglas gramaticales se definen, por tanto, de la siguiente manera (léase *up* como la metavariable (\uparrow), y *down* como (\downarrow) en LFG).

```

% S ->      NP      VP
%      up.subj=down  up=down
%      up.num=down.num  up.tense

s(s(NP,VP),Fs) ->  np(NP,Fnp),
                   vp(VP,Fvp), {unify(Fvp,[Subj:Fnp],Fs)}.

```

Como se puede observar, el sistema difiere notablemente del algoritmo propuesto por Kaplan y Bresnan [BRE82] ya que el valor de las metavariables no es calculado para resolver las ecuaciones, sino que se delega esa tarea en el mecanismo de unificación.

Por así decir, cada regla es responsable de construir su porción de f-structure y su valor se unifica con los resultados adyacentes en el nodo superior, y así sucesivamente.

Las condiciones de gramaticalidad que postula LFG son efectuadas en el curso del análisis. Claramente, el test de uniqueness queda demostrado al aplicar el algoritmo de unificación. Los tests de coherence y completeness se llevan a cabo cada vez que un predicado requiere la realización de un marco de subcategorización. Así, por ejemplo, en la regla que analiza los verbos trivalentes tipo *give*, se comprueba que el verbo requiere un *obj* y un *obj2* antes de intentar esa regla. Este test se efectúa antes de la posible unificación, con lo que se ahorra bastante tiempo de análisis innecesario. Una vez que la regla se ha satisfecho se efectúa el test de completeness comprobando que cada uno de los argumentos que aparecen en el marco de subcategorización tiene el rasgo *pred* realizado. No obstante, sería mas eficiente indexar las reglas de manera que solo se intentaran aquellas que cubrieran los argumentos requeridos por el marco verbal y no intentando una por una como hace Prolog.

```

% VP -> V      NP1      NP2
%      up.obj2=down  up.obj=down
vp(vp(v(V),NP1,NP2),Fvp) ->  v(V,Fv), {allows(Pred,obj),
                                         allows(Pred,obj2)},
                                np(NP1,Fnp1), {unify(Fv,[obj2:Fnp1],F1)},
                                np(NP2,Fnp2), {unify(F1,[obj:Fnp2,Fvp],
                                                    complete(Fvp,Pred))}.

```

3. Lexical Functional Grammar

Como apuntamos anteriormente, este prototipo no pretende demostrar los postulados de la LFG. Realmente, sería mas apropiado decir que utilizamos un formalismo *muy parecido a LFG* o *de tipo LFG*.

Aumentamos los rasgos estrictamente sintácticos propuestos en el modelo estándar LFG, con casos semánticos (*theme*, *agent*, *goal*), que realizan cada una de las funciones gramaticales del marco verbal, y con rasgos semánticos (*human*, *edible*, *animate*), ambas técnicas muy conocidas en TA.

Esta información sirve de apoyo tanto para el análisis en la determinación de las preferencias semánticas que el verbo impone con respecto a sus argumentos, y para la transferencia estructural, especialmente en el caso de las pasivas.

La gramática cubre una pequeña porción de fenómenos lingüísticos. En el momento de escribir el artículo eran: morfología completa del inglés, presentes y pasados progresivos y perfectos, pasivas, verbos monovalentes, bivalentes y trivalentes, verbos de control por el objeto tipo *persuade*, y oraciones de relativo con el pronombre relativo implícito y explícito.

Por el momento estamos más interesados en desarrollar los módulos de análisis, transferencia y generación en paralelo, para comprobar la flexibilidad y adecuación de los formalismos que utilizamos.

4. Enfoque de Traducción Automática

Se puede observar que hemos adoptado un enfoque de TA tipo transfer, de segunda generación, bastante estándar, con los módulos claramente definidos. El resultado del análisis habrá de ser, por tanto, una representación de la oración en lengua fuente (LF), según el modelo LFG.

!?- try.

! : the aspirin the doctor gave to the boy was poisoned.

----- c-structure -----

```
s(np(det(the),
  n(aspirin),
  rel(relpron(nil),
    np(det(the),
      n(doctor)),
    vpgap(v(gave),
      npgap(trace),
      pp(pre(to),
        np(det(the),
          n(boy)))))),
  vp(v(v(was)),
    ap(adj(adj(poisoned))))))
```

----- s-structure -----

```
pred:be(acomp)
tense:past
num:sing
subj:pred:aspirin
  num:sing
  spec:the
  semfeat:rx:yes
  qual:pred:give(subj,obj,pobj)
    tense:past
    relpron:nil
    subj:pred:doctor
      num:sing
      spec:the
      semfeat:human:yes
      doc:yes
    obj:pred:trace
      antec:pred:aspirin
        num:sing
        semfeat:rx:yes
        semfeat:human:no
      pobj:obj:semfeat:human:yes
        pred:boy
        num:sing
        spec:the
    pcase:to
  acomp:pred:poisoned
  temp:yes
```

5. Modelo de Transferencia

Como se ha podido observar, el nivel de representación desde el que queremos transferir es bastante superficial. No obstante, es el adoptado por la mayoría de los sistemas de segunda generación [KUD86, NAG86, ALO88, LEO86].

Esto responde también a dos premisas apuntadas anteriormente:

- a) El tipo de textos que pretendemos traducir presenta una estructura muy similar a su equivalente en español.
- b) La distancia entre el par de lenguas que se va a traducir no es muy grande, e incluso en lenguas tan distantes como el inglés y el japonés, se adopta el mismo enfoque [KUD86].

Nuestro módulo de transferencia es comparable al de Kudo y Nomura [KUD86]. La diferencia estriba nuevamente en que donde ellos aplican un algoritmo de resolución de ecuaciones para determinar la *f-structure* de la oración en lengua destino (LD), nosotros aplicamos el algoritmo de unificación tal como hicimos en la fase de análisis.

Hemos de entender, pues, este módulo de transferencia como un nuevo proceso de unificación cuyo resultado será la representación en LD.

El algoritmo de transferencia se compone de varios pasos sucesivos

```
transfer(English,Spanish) :-
  prune(English,Eng1),
  rearrange(Eng1,Eng2),
  lexftrf(Eng2,Span1,Eng2),
  strtrf(Span1,Span2,Span1),
  prune2(Span2,Spanish).
```

En un primer paso se elimina la información irrelevante al proceso de transferencia (*prune*). Posteriormente se ordena la lista que contiene los pares atributo-valor de tal manera que el núcleo de cada función gramatical (*pred*) vaya en primer lugar.

A continuación se procede a la transferencia léxica. Solo algunos rasgos han de ser transferidos, aquellos que son *translatable*. El resto se copian en la lista destino. Esto ahorra un tiempo de consulta innecesaria en el diccionario bilingüe.

```
lextrf([F1:V1|Rest],Result,Model) :-
  translatable(F1),
  ((F1:V1|Cond1) => Rhs,
  unify(Cond,Model,Model),
  lexftrf(Rest,Result1,Model),
  unify(Rhs,Result1,Result).
```

En el caso de que el rasgo sea *translatable*, se busca en el diccionario la regla de transferencia léxica que corresponde a dicho rasgo. El diccionario bilingüe se compone de una parte izquierda y una parte derecha unidas por el operador '=>'. La parte izquierda se compone a su vez de una lista con un primer elemento y su cola. El primer elemento es el rasgo que queremos traducir y la cola actúa como contexto que ha de satisfacerse como condición para transferir ese rasgo como se indica en la parte derecha de la regla.

([pred:be(acomp),acomp:[temp:yes]]) => ([pred:estar(acomp)]).
 ([pred:be(acomp),acomp:[temp:no]]) => ([pred:ser(acomp)]).

Es aquí donde entra en juego la flexibilidad del mecanismo de unificación. Si el resultado de unificar A y B es B, implica que A se contiene en B, es decir, el contexto A se ha satisfecho. Esto nos permite definir una condición a cualquier nivel de la estructura, ya que el mecanismo de unificación puede recorrer la intrincada *f-structure* en busca de dicho valor.

La parte derecha de la regla incluye la traducción y la información pertinente a dicha unidad léxica en cuanto a la LD (género, reflexibilidad, clíticos, etc).

Resultado de transferir *The aspirin the doctor gave to the boy was poisoned*

```

transfer? (y/n)
l: y.
pred:estar(acomp)
tense:past
num:sing
subj:pred:aspirina
  num:sing
  spec:el
  gen:fem
  semfeat:rx:yes
qual:pred:dar(subj,obj,pobj)
  tense:past
  relpron:que
  cl:pred:pro
  case:dat
  subj:pred:doctor
    num:sing
    spec:el
    gen:masc
    semfeat:human:yes
    doc:yes
  obj:pred:trace
    antec:pred:aspirina
      num:sing
      gen:fem
      semfeat:rx:yes
    semfeat:human:no
  pobj:obj:semfeat:human:yes
    num:sing
    spec:el
    pred:ni^no
    gen:masc
  pcase:a
acomp:pred:envenenado
temp:yes
  
```

El resultado del *transfer* léxico es una *f-structure* todavía propia del inglés, pero con los nodos terminales en español, tal como hace TRANSF en ARIANE-78 [HUT86].

6. Otras alternativas al modelo de transferencia

Ultimamente ha habido cierto debate en cuanto a cómo ha de ser el mecanismo de transferencia en un sistema de TA basado en unificación, y, en concreto, en LFG. Frente al modelo típicamente constructivista y derivacional que presentamos, adoptado también por Kudo y Nomura [KUD86], Kaplan y otros [KAP89, ZAJ90] proponen el modelo de *co-description*.

En dicho modelo, a las ecuaciones que han de ser resueltas en el módulo de análisis, se añaden otras que determinan la traducción de dicha unidad léxica.

A pesar de lo atractivo del modelo y de los resultados que obtiene para casos de difícil traducción, Sadler y otros [SAD90] demuestran que la propuesta de Kaplan no se sostiene para los casos en que la traducción de un núcleo esté condicionada de alguna manera por sus nodos dependientes, como es el caso de

(1) The doctor committed a crime => El doctor cometió un delito

(2) The doctor committed suicide => El doctor se suicidó

* El doctor cometió un suicidio

En este caso, la lengua fuente usa una combinación de núcleo + dependiente que se transfiere como la fusión de ambos a la LD, sin constituir una locución idiomática el caso (2), ya que su análisis es idéntico al de (1).

Aunque los criterios de completeness y coherence rechazaran posteriormente la traducción (2*) como no gramatical, necesitaríamos en el diccionario una doble entrada para *suicide*, una que sería traducida y otra que no lo sería, con una disjunción opcional representada aquí por las llaves.

suicide $\{(\tau \uparrow \text{PRED FN}) = \text{suicidio} \quad [\text{regular}]$
 $(\tau \uparrow) = \text{nil} \quad [\text{irregular}]$

Esto no es razonable lingüísticamente. No es razonable afirmar que la traducción de una unidad léxica es cero (nil) sin especificar bajo qué contexto se da esa relación. Pero no hay nada en la segunda ecuación anterior que indique su posible relación con *commit suicide* y no otra cosa.

En definitiva, queda claro que

1. Las dependencias léxicas de este tipo no pueden ni deben ser reconocidas explícitamente como especiales según criterios monolingües. Por tanto la propuesta de Kaplan habría de ser revisada. Esto se debe a que las ecuaciones de traducción se añaden a las unidades léxicas de la LF y no a la representación funcional. De esta forma, cuando queremos decir que la representación que tiene *commit* como núcleo y *suicide* como objeto se traduce como *suicidarse*, no podemos, y hemos de intentar conseguir el mismo efecto diciendo cosas de *commit* y de *suicide* por separado, lo que afectaría gravemente la transparencia de las gramáticas.

2. Es preferible una representación que sea justificable desde el punto de vista monolingüe, y que no afecte a la modularidad del sistema.

7. Control y recursión

Aunque Sadler y otros tienen razón, no aportan una solución a la cuestión. Este tipo de fenómeno presenta dos problemas íntimamente relacionados el uno con el otro: control y recursión. Idealmente, los mecanismos de transferencia deberían ser totalmente composicionales y recursivos, tal como proponen Isabelle y Macklovitch [ISA86] con la siguiente regla.

```
trf(np(Det,Adj,N), np(Det',N',Adj'))
:- trf(Det,Det'),
   trf(Adj,Adj'),
   trf(N,N').
```

Pero, como apuntan Nagao y Tsujii [NAG86] y Alonso [ALO88], es necesario controlar el proceso de alguna manera, y una recursión pura no es posible. Esto es especialmente importante en Prolog, que tiene incorporado su propio mecanismo de reevaluación (*backtracking*). En principio, pues, los mecanismos de transferencia secuenciales suelen ser más eficientes que los puramente declarativos. Quizá esto arroje luz de por qué los sistemas que usan aquél producen resultados de mayor calidad, como en el caso de METAL [ALO88], ENGSPAN [LEO86] y Mu [NAG86], frente a ARIANE-78 [HUT86:239-248], por ejemplo.

El segundo problema es recursión. El problema que plantea el ejemplo (2) en mecanismos de transferencia recursivos es de sobregeneración: cómo evitar que en el caso de *The doctor committed suicide* no se traduzca *suicide*.

Hay dos maneras de solucionar esto. Una sería declarar que *suicide* se traduce como cero en el contexto *commit suicide*. La otra sería borrar *suicide* del resto de la estructura que queda por traducirse. Nosotros hemos adoptado esta segunda posibilidad.

```
lextrf([F1:V1|Rest],Result,Model) :-
  translatable(F1),
  ([F1:V1|Cond] & Y) => Rhs,
  unify(Cond,Model,Model),
  defeat(Y,Rest,Rest1),
  lextrf(Rest1,Result1,Model),
  unify(Rhs,Result1,Result).
```

Sin embargo, cada vez que transferimos una unidad léxica, la estructura original va reduciéndose por efecto de la recursión, con lo que perdemos información de contexto que nos podría ser útil en el futuro. Para prevenir esto, nuestro mecanismo mantiene una copia intacta del resultado del análisis que no será afectada por la recursión (Model en la figura anterior). De esta forma cada vez que la condición de transferencia ha de ser comprobada, tenemos la seguridad de que podrá hacerlo.

Con respecto al control, utilizamos el predicado *rearrange*. Este predicado ordena la *f-structure* de tal manera que los rasgos *pred* vayan al principio de la función que describen. Estos *pred* indican el núcleo de la función gramatical, con lo que controlamos de algún modo el proceso de transferencia comenzando por el verbo principal.

El resultado es el siguiente para el caso que se discutía.

```
!?- try.
!: the doctor committed suicide.
s(np(det(the),
     n(doctor)),
  vp(v(v(committed)),
     np(n(n(suicide))))))
```

pred:commit(subj,obj)
 tense:past
 num:sing
 subj:pred:doctor
 num:sing
 spec:the
 semfeat:human:yes
 doc:yes
 obj:pred:suicide
 num:sing
 semfeat:event:yes

transfer? (y/n)
 l: y.
 pred:suicidar(subj)
 tense:past
 num:sing
 refl:yes
 subj:pred:doctor
 num:sing
 spec:el
 gen:masc
 semfeat:human:yes
 doc:yes

l?- try.
 l: the doctor committed a crime.
 s(np(det(the),
 n(doctor)),
 vp(v(v(committed)),
 np(det(a),
 n(crime))))

pred:commit(subj,obj)
 tense:past
 num:sing
 subj:pred:doctor
 num:sing
 spec:the
 semfeat:human:yes
 doc:yes
 obj:pred:crime
 num:sing
 spec:a
 semfeat:event:yes

```

transfer? (y/n)
l: y.
pred:cometer(subj,obj)
tense:past
num:sing
obj:pred:delito
  num:sing
  spec:un
  gen:masc
  semfeat:event:yes
subj:pred:doctor
  num:sing
  spec:el
  gen:masc
  semfeat:human:yes
  doc:yes

```

8. Transferencia estructural

Una vez efectuada la transferencia léxica se procede a la transferencia estructural en caso de que sea necesaria. El algoritmo es esencialmente el mismo que usamos para la transferencia léxica, con lo que no será reproducido de nuevo aquí.

Un caso típico en el que se debe usar es en el cambio de una estructura pasiva en inglés, ampliamente empleada en lenguaje técnico, a una estructura activa, preferida en español.

```

((subj:X,subj:[role:goal],vcomp:[part:pass,obj2:[role:theme]])) ==>
  ([pobj:[pcase:a,obj:X]]).

```

De nuevo, la información lingüística está separada del algoritmo que la usa, con lo que mantenemos la modularidad requerida. El resultado que se obtiene es el siguiente.

```

!?- try.
l: the boy was given a cake by the girl.
s(np(det(the),
  n(boy)),
  vp(v(v(was)),
    vp(v(v(given)),
      np(det(a),
        n(cake)),
      pp(preposition(by),
        np(det(the),
          n(girl))))))

```

pred:be(vcomp)
 tense:past
 num:sing
 subj:pred:boy
 role:goal
 num:sing
 spec:the
 semfeat:human:yes
 vcomp:pred:give(pobj,obj2,subj)
 part:pass
 subj:role:goal
 num:sing
 semfeat:human:yes
 obj2:pred:cake
 num:sing
 spec:a
 role:theme
 semfeat:edible:yes
 human:no
 pobj:obj:semfeat:human:yes
 edible:no
 pred:girl
 num:sing
 spec:the
 role:ag
 pcase:by

transfer? (y/n)

l: y.

```

pred:dar(subj,obj,pobj)
tense:past
num:sing
obj:pred:pastel
  num:sing
  role:theme
  spec:un
  gen:masc
  semfeat:edible:yes
  human:no
subj:pred:ni^na
  num:sing
  role:ag
  spec:el
  gen:fem
  semfeat:human:yes
  edible:no
ci:pred:pro
  case:dat
pobj:pcase:a
  obj:role:goal
  num:sing
  semfeat:human:yes
  spec:el
  pred:ni^no
  gen:masc

```

9. Referencias

- [ALO88] Alonso, J.A. 'A Model for Transfer Control in the METAL MT-System'. Coling 88, Budapest, 1988. 19-94.
- [BRE82] Bresnan, J. (ed.) *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, Mass., 1982.
- [EIS86] Eisele, A. y Dorre, J. 'A Lexical Functional Grammar in Prolog'. Coling 86, Bonn, 1986. 551-553.
- [FIN87] Finin, T. y Stone Palmer, M. 'Parsing with Logical Variables' en *Natural Language Parsing Systems* (Leonard Bolc, ed), Springer Verlag, Berlin, 1987. 33-45.
- [GAZ89] Gazdar, G. y Mellish, Ch. *Natural Language Processing in Prolog*. Addison Wesley, Workingham, 1989.
- [HIR88] Hirsch, S. 'P-PATR: A Compiler for Unification-Based Grammars' en *Natural Language Understanding and Logic Programming* (Veronica Dahl y Patrick Saint-Dizier, eds), North Holland, Amsterdam, 1988. 63-78.
- [HUT86] Hutchins, W.J. *Machine Translation*. Ellis Horwood, Chichester, 1986.
- [ISA86] Isabelle, P. y Macklovitch, E. 'Transfer and MT Modularity' en Coling 86, Bonn, 1986. 115-117.
- [KAP89] Kaplan, R.M., Netter, K., Wedekind, J. y Zaenen, A. 'Translation by Structural

Correspondences'. 4th Conference of the European Chapter of the ACL, Manchester, 1989. 272-81.

[KUD86] Kudo, I. y Nomura, H. 'Lexical-Functional Transfer: A Transfer Framework in a Machine Translation System based on LFG'. Coling 86, Bonn, 1986. 112-115.

[LEO86] León, M. y Schwartz, L. *Integrated Development of English-Spanish Machine Translation: From Pilot to Full Operational Capability*. Technical Report. Pan American Health Organization, Washington D.C. Octubre 1986.

[NAG86] Nagao, M. y Tsujii, J. 'The Transfer Phase of the Mu Machine Translation System'. Coling 86, Bonn, 1986. 97-103.

[PER80] Pereira, F.C.N. y Warren, D. 'Definite Clause Grammars for Language Analysis – A Survey of the Formalism and a Comparison with Augmented Transition Networks'. *Artificial Intelligence* 13 (1980), 231-78.

[PER87] Pereira, F.C.N. y Shieber, S.M. *Prolog and Natural Language Analysis*. CSLI, Stanford, Ca., 1987.

[SAD90] Sadler, L., Crookston, I., Arnold, D. y Way, A. 'LFG and Translation'. Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language, Austin, Texas, 1990.

[SHI86] Shieber, S.M. *An Introduction to Unification-Based Approaches to Grammar*. CSLI, Stanford, Ca., 1986.

[ZAJ90] Zajac, R. 'A Relational Approach to Translation'. Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language, Austin, Texas, 1990.