Episteme

J. Gabriel Amores
Departamento de Lengua Inglesa
Universidad de Sevilla
Tel: +34-5-455.1549
Fax: +34-5-455.1516
gaby@fing.us.es

José F. Quesada
CICA
Sevilla
Tel: +34-5-462.3811
Fax: +34-5-462.4506
josefran@cica.es

Abstract

This paper describes the overall architecture of Episteme, a tool for the development of efficient LFG–based MT systems following the classical transfer approach. The system incorporates a series of novel computational techniques which enhance the overall performance significantly: representation, storage and retrieval of very large feature structure–based knowledge bases, bidirectional event–driven bottom up parsing with top–down predictions and constructive unification with post–copy.

Keywords: Machine Translation, Language Engineering.

1 Introduction

This paper describes the overall architecture of Episteme.

Episteme may be defined as the core of a tool for the development of Machine Translation (MT) systems, similar to Lekta [4, 3], but superior to it in computational efficiency and linguistic coverage.

It has taken approximately two years to develop the tool, until a satisfactory level of robustness and efficiency has been achieved. Our previous experience in the development of JULIETTA [2] and Lekta laid the foundations upon which Episteme has been developed. Our goal has been to obtain a system inspired in unification grammars while achieving the best performance in analysis times. The system has been written in C programming language, comprising more than 20.000 lines of code, and its efficiency results from the implementation of the following techniques:

1. **Representation, Storage and Retrieval of Very Large Feature Structure–based Knowledge Bases.** The lexical module is based on Improved Binary Trees with Vertical Cut [18]. The computational complexity of this technique is $O(log(log(N)))$, where N is the length of the lexicon. We have tested it with artificial dictionaries obtaining that the time necessary to analyze a single lexical item varies from 1 millisecond (with dictionaries of up to 5,000 entries) to 3 milliseconds (with dictionaries of up to 134,000,000 lexical entries).
2. **Bidirectional Event–driven Bottom Up Parsing with Top–Down Predictions.** From a theoretical point of view, this parsing technique [19] reduces between 80 to 95% the amount of arcs and nodes in a conventional chart parser. That is, Episteme only generates 20% of structures in the worst case. The practical consequence of this is that we can parse between 2,000 and 5,000 words per

2.1 Machine Translation Strategy

As regards MT strategy, Episteme follows a transfer approach, which fits perfectly with the double representation which LFG assigns to every sentence. Intuitively, the f–structure serves ideally as the input to transfer in a conventional MT system. While the c–structure conveys language–dependent information which is discarded during transfer, grammatical relations render language–independent information of the sort needed during transfer.

Episteme generates a c– and an f–structure for each sentence in the source language during the analysis phase. The transfer module goes from source f–structure to target f–structure, and generation goes from target f–structure to target c–structure.

The figure below shows the overall architecture of the system, including the main components from a structural standpoint, the functional relations among them, and the flow of information during the translation process.

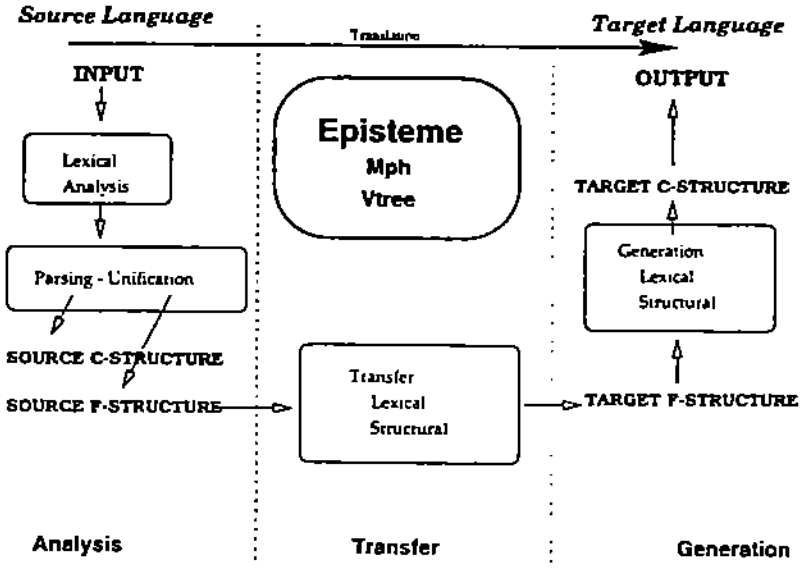


Fig. 1 Overall architecture of Episteme.

3 Specifying Linguistic Knowledge in Episteme

Episteme may be described as the core of a tool for the development of MT systems. That is, Episteme is not a MT system itself, but a *shell* to develop MT systems. Accordingly, it is equipped with a series of specification languages whereby the necessary linguistic knowledge is incorporated into the system, and a series of control commands to configure the functioning of the system.

The main concept in Episteme is that of language. The system permits the definition of more than one language, with the only limitation of hardware features. For each language, we may build an analysis grammar, an analysis lexicon, a set of transfer modules to other languages, a generation grammar, and a generation lexicon. All five components are optional for each language.

A configuration command indicates which source and target languages will be active during the translation process. Thus, if we indicate that we wish to translate from language *English* into language *Spanish*, we must have previously defined an analysis grammar and lexicon for *English*, a transfer module into *Spanish*, and a generation grammar and lexicon for *Spanish* as follows:

second. These results have been tested with grammars including recursion, and local and non-local dependencies.

3. **Constructive Unification with Post-Copy.** This algorithm incorporates the following strategies: structure-sharing, reversible unification, constructive unification, disunification and post-copying [20]. Constructive unification by itself eliminates completely the problems of pre-copying, over-copying and redundant copying. This set of techniques reduces the computational load (memory and time) up to a 98% when compared with basic unification algorithms such as the naïve algorithm or the default use of unification in Prolog.

2 Machine Translation

Machine Translation (MT) has become a major area of research and discussion over the past four decades. The task of building systems to mechanize the translation process has been pursued from widely varied and even diametrically opposed approaches. Thus, we may find research groups trying to develop systems based on statistical approaches with (almost) no linguistic information [7] or, at the other extreme, systems which try to incorporate as much world knowledge as possible in the translation process [12]. It is possible, however, to classify MT systems into two categories, depending on the motivations that led to their creation.

1. On the one hand, we find systems created with the purpose of satisfying specific translation needs. Experience shows that when the user environment and the design issues that result from it are clearly defined, MT can be very successful. Examples would be the ENGSPAN/SPANAM system [23] at the Pan American Health Organization, Washington, D.C., Systran at the European Commission, or the often cited Météo [9].

   This approach shows that MT is practical, that it may be cost-effective, and that it can produce more-than-acceptable results that satisfy MT customers at large. We may also include in this group PC-based Machine-Assisted Translation tools such as Globalink, PC Translator, etc.

   Despite of some outstanding achievements, systems belonging to this category are open to criticism on two levels: they are not linguistically motivated and they tend to exhibit a lack of modularity in the separation of the algorithms and the linguistic data.

2. On the other hand, we find linguistically-motivated systems, aimed at solving difficult linguistic and/or translation problems. Eurotra [1] and its derivatives CAT and MiMo [21, 17], and Unitran [11] could be cited as examples of systems belonging to this category. These systems provide effective solutions for many of the difficult translation problems, but it is our belief that they cannot be made operational in a real environment, i.e., with large grammars and large lexicons.

The prototype we will describe below tries to demonstrate that the adoption of a sound and computationally-oriented linguistic theory, specifically Lexical-Functional Grammar (LFG) [13] offers an excellent framework for the development of MT systems that are both operational and linguistically-motivated.

This work builds on earlier versions which studied the feasibility of using LFG to translate medical abstracts from English into Spanish using JULIETTA, a Prolog-based MT, and Lekta, a similar version in C with some limitations in the treatment of structural ambiguity.

```
/***
 ***   SampleTrans.epstm
 ***/

BeginningOfLanguage English
       AnalysisGrammar FromFile E_grammar
       AnalysisLexicon FromFile E_lexicon UsingPred Elex
       TransferToLanguage Spanish FromFile ES_transfer
EndOfLanguage

BeginningOfLanguage Spanish
       GenerationGrammar FromFile S_grammar
       GenerationLexicon FromFile S_lexicon UsingPred Slex
EndOfLanguage

ConfTranslation: English => Spanish

InputSentence (John gave a present to Mary)
InputSentence (John gave Mary a present)
```

In addition, Episteme is equipped with a tool to measure the statistical performance of the system, and a 'trace' capability which generates detailed information about all the operations involved in the translation process.

In the following sections we will offer a detailed study of each of the processes which take place since the system receives a string of words as input until it generates the corresponding translation.

4 Lexical and Morphological Analysis

The phase of lexical analysis receives a string of words as input and its goal is to output a list of the syntactic category/ies and functional structure/s associated with each word.

4.1 Simulating Morphological Analysis through Morphological Generation and Efficient Knowledge-Base Retrieval

The lexicon is built following Mph-Vtree syntax [18]. Mph defines a sophisticated language for the specification of lexicons for unification grammars. Its output may be linked to Vtree, a powerful system for the efficient storage and retrieval of large feature structure-based knowledge bases. The joint use of both systems produces a specification environment close to the linguist and a very efficient module for lexical and morphological analysis.

From a computational standpoint it is a model based on inflected forms, as oppossed to other paradigms based on two-level morphology [15].

Nevertheless, the system does not require that all inflected entries be coded manually. Instead, Mph permits the definition of regular morphological phenomena. An additional advantage is that the same specification language may be used to express lexical redundancy rules and some cases of derivational morphology.

Finally, using Vtree to store and retrieve lexical items once they have been generated by Mph results in excellent times. Specifically, Vtree obtains a performance rate of milliseconds per word, independently of the morphological complexity of the languages involved. Complexity only affects Mph, and the time consumed by Mph to generate all forms is compilation time, which is performed just once.

In sum, a lexical analysis model based on morphological generation such as the one presented here is preferable to models based on morphological analysis in real-

time applications and with a large lexicon, where a maximum response time is usually required.

4.2 Mph

Mph has been designed for typed feature structures. Namely, it uses the notion of *shape* to refer to complex feature structures permitted in a language. A shape defines the skeleton of a structure, that is, the attributes which may or must be instantiated.

A shape definition is formed by four components: its name or identifier, body, list of indexes and transformational rules of shapes associated

A body definition in a shape is a list of attribute–value pairs.

The last component in a shape definition are transformational rules. One of the goals of transformational rules over shapes is to capture morphological generalizations found in natural languages. They may also be used as lexical redundancy rules to capture transitive alternations in verbs, for example. Each rule contains two components: a pattern and a set of target structures.

Meta–relations allow us to associate a shape with generation models for new shapes. Despite the use of macros and a transformational rule to obtain the plural of regular nouns, each of the entries above requires specifying all new values, which in fact are the same for all entries. This situation may be simplified by using input forms or *iforms*. These constructions permit to associate a flat, Prolog–like predicate with a complex feature structure, incorporating all the expressive power of Mph. That is, *iforms* allow the inclusion of macros, functions, multi–word entries, etc.

Effectively, the simultaneous use of shapes and macros permits the design of a hierarchy of typed feature structures. In addition, Mph incorporates a default inheritance strategy, whereby assigning different values to the same attribute does not result in an error.

The next example illustrates the expressive power of Mph:

```
/* English
 *      Analysis Lexicon */

BeginningOfLexicon
```

Macro Definitions

```
DefMacros
        <MSg> = (agr:(gen:masc,num:sing,per:3))
        <MPl> = (agr:(gen:masc,num:plur,per:3))
```

Shapes Definition

```
DefShapes
```

Elex: Meta–predicate to establish the link with Episteme

```
        Elex (LU)
        Eagr (agr:(gen,num,per))
```

@shape: shapes hierarchy, default inheritance, and exceptions

```
        Enoun (LU,CAT:n,MOR,head,ggf,@Eagr)
```

```
ActShape Everb
        (LU:give, MOR:[V1,LR1], pred:give, ggf:[subj,obj,pobj],
         pobj:(pcase:to), tense:pres)

        (LU:gave, MOR:[LR1], pred:give, ggf:[subj,obj,pobj],
         pobj:(pcase:to), tense:past)

ActShape Edet
        (LU:a,agr:(num:sing),spec:a)
```

Additional Features:
Lexical Ambiguity,
Homonimy, Disjunction,
and Negation in Atomic Values, etc.

```
EndOfLexicon
```

5 Analysis Grammar

From a functional point of view, the parsing and unification modules in Episteme have been implemented following an *interleaving* strategy. That is, the parser interacts with the unifier during the analysis process.

Broadly speaking the parser in Episteme may be described as a *bidirectional bottom-up chart*, incorporating *top-down predictions*.

The efficiency of a bottom-up chart parser may be increased if useless arcs are eliminated in the first stages of the process. Top-down predictions have been incorporated in Episteme with that goal.

We have implemented a set of simple and intuitive mathematical relations between the nodes in a grammar which allow us to determine whether certain arcs have no guarantee of sucess on certain occasions.

The model of top-down predictions requires that the parser knows all the information regarding possible arc applications over current nodes. This information is obtained through a model of bidirectional event generation.

Our parsing strategy approaches the problem of efficiency from an algorithmic point of view. In addition, Episteme incorporates a computational approach to increase the parsing efficiency further. Namely, the grammar is compiled beforehand, obtaining an internal representation of it which reduces the comparison of strings of characters considerably.

An analysis grammar in Episteme consists of 3 components:

- A series of configuration parameters, of which only `RootsOfGrammar` will be used by the parser. `RootsOfGrammar` specifies possible termination symbols in the grammar, thus allowing for grammars with multiple root symbols.
- A set of context-free productions, each of which contains an identifier, a non-terminal symbol in the left-hand side of the rule, and one or more terminal or non-terminal symbols in the right-hand side of the rule.
- Each production may contain a set of functional equations which will be passed on to the unification module.

Figure 3 shows a simple English grammar written in a format acceptable for Episteme.

Morphological Rules

```
                RulePattern (MOR:[N1])
                RuleTarget {
                        (MOR:null())
                        (LU:strcat(base->LU:,s),<MP1>,MOR:null())
                }
        Everb (LU,CAT:v,MOR,pred,ggf,@Eagr,tense,pobj:(pcase))
                RulePattern (MOR:[V1])
                RuleTarget {
                 (MOR:extract(base->MOR: [V1]))
                 (MOR:extract(base->MOR.,[V1,VED1]),
                        agr:(per:1|2,num:sing))
                 (MOR:extract(base->MOR:,[V1,VED1]),
                         LU:strcat(base->LU:,s),agr:(per:3,num:sing))
                 (MOR:extract(base->MOR:,[V1,VED1]),agr:(num:plur)) }
                RulePattern (MOR:[VED1])
                RuleTarget {
                 (MOR:extract(base->MOR:,[VED1]),
                  LU:strcat(base->LU:,ed),tense:past) }
```

Lexical Redundancy Rules

```
                Rulepattern (MOR:[LR1])
                RuleTarget {
                 (MOR:extract(base->MOR:,[LR1]))
                 (MOR:extract(base->MOR:,[LR1]),
                        ggf:[subj,obj,obj2],pobj:null()) }

        Edet (LU,CAT:det,@Eagr,spec)
        Eprep (LU,CAT:prep,pcase)
```

Meta–Relations

```
DefMetas
        MetaPattern Enoun()
        MetaTarget Elex(LU:base->LU:)

        MetaPattern Everb()
        MetaTarget Elex(LU:base->LU:)
```

Input Form Definitions

```
    DefIForms
            IFnoun(LU,MOR,ggf)
                    Enoun (LU:base->LU,MOR:base->MOR,head:base->LU,
                            ggf:base->ggf,<MSg>)
```

Regular Entries through
the activation of Input Forms

```
    ActIForm IFnoun
            (present,[N1],[])
            (girl,[N1],[])
            (telescope,[N1],[])

            (John,[],[])

    ActIForm IFverb
            (present,[V1,VED1],[subj,obj,pobj],with)
```

Irregular entries through
the activation of Shapes

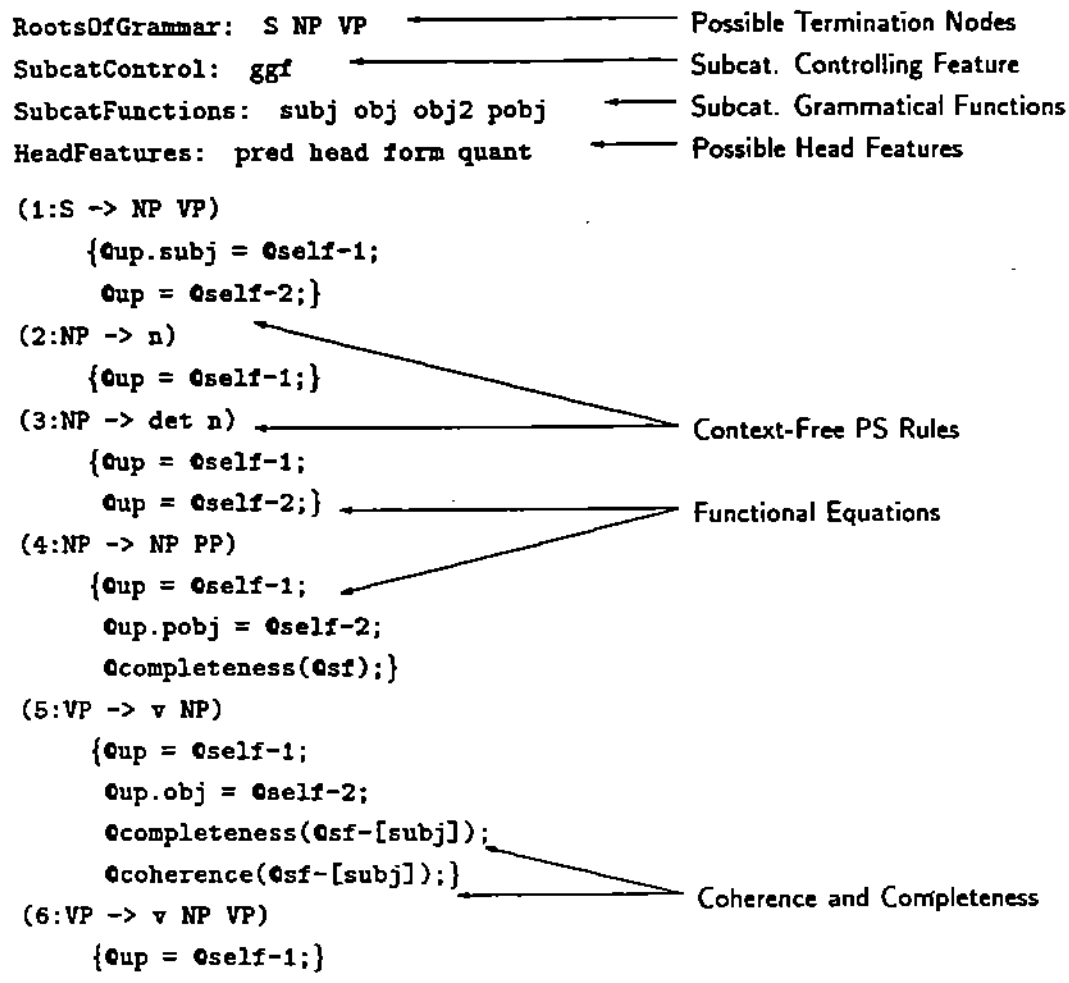


Fig. 2 English Basic Grammar for Episteme.

5.1 Unification

The unification module of any unification–based NLP system usually consumes around 80 or 90% of the total computation time. The relevance of this module justifies that we make a special effort in the design of the algorithms and implementation strategies. In addition, we should take into account the linguistic requirements regarding the expressive power that unification grammars usually demand.

Thus, we could divide the unification module in two distinct components. On the one hand, we have the unification algorithm proper, which is independent of any linguistic formalism. On the other, we would find the specification layer, which tries to capture the strategies and notations found in the particular theory being implemented. In our case, the latter has been designed having LFG in mind, although the algorithm is valid for any unification–based formalism.

The core of the module implements a reversible unification strategy, based on disunification and post–copying [20]. The strategy relies on a sophisticated data organization which obviates most copying processes during unification. If unification fails, the disunification algorithm recovers the original data structures faithfully. If unification succeeds the result is copied (post–copied) and the disunification process recovers the original

input structures.

The current version allows the use of atomic values, atom negation and disjunction (negated or not), and lists.

As regards the LFG notation, the unification algorithm covers the basic equational unification (=) plus: structure assignment (=a), evaluation and conditional execution (`if ...then ...else`), specific functions for the manipulation of character strings and lists (`@concat`, `@member`, `@count`), mathematical operators (+,-,*,/), logical (!,&&,||) and relational operators (==,!=,<,<=,>,>=), and coherence and completeness controls (`@coherence`, `@completeness`). Classical LFG metavariables ↑ and ↓ are called `@up` and `@self-N` in Episteme. We have not implemented =c restrictions [13] nor functional uncertainty [14].

The example in Fig. 2 includes some of these functions.

6 Transfer

After the analysis phase, the parser and the unifier have obtained one or more constituent structures (c–structure in LFG) and one or more functional structures (f–structure) for each grammatical sentence. According to the machine translation approach we are assuming in Episteme, the next step will consist in transferring the source f–structure into an equivalent f–structure in the target language. [1]

The transfer module is divided in two stages, as in most transfer–based MT systems. First, the **lexical transfer** phase applies translation rules triggered by atomic–valued features. Then, during **structural transfer**, translation rules may modify complex–valued features, which results in changing the internal structure of the sentence.

One of the innovations incorporated in the transfer module is that the user may specify the order in which features are to be transferred, according with the following syntax:

```
FeatureTransferOrder: <first-list> ... <last-list>
```

This option has been implemented to allow a wide variety of cases. For instance, it is possible to indicate which feature has the highest priority of all:

```
FeatureTransferOrder: a ...
```

the least priority:

```
FeatureTransferOrder: ... z
```

or a specific sequence:

```
FeatureTransferOrder: a b c ... x y z
```

6.1 Lexical and Structural Transfer

Lexical and Structural Transfer rules are very similar in their syntax. They consist of three main components:

1. A triggering feature:
2. A set of conditions: and

[1] It is possible to set up the number of analysis, transfer and generation outputs we wish to obtain when the input is ambiguous.

3. A set of actions.

Following is a listing of two simple transfer rules. The first will transfer the feature descr as pmod inserting the appropriate preposition as well. This rule covers the generalization of tranferring premodifying nouns as prepositional phrases with *de* into Spanish: *data structure => estructura de datos.*

The second rule changes the internal structure of the verb *give* followed by two objects into a *v NP PP* pattern in Spanish: *to give someone something => dar algo a alguien.*

```
StructuralTransfer descr
    (=> pmod @do {@target.pmod.pcase =a de; })

LexicalTransfer pred
    (give => dar @when (@source.ggf == [subj,obj,obj2])
                @do {@transferas(@source.obj2,@target.obj);
                     @transferas(@source.obj,@target.pobj);
                     @target.pobj.pcase =a a}
            dar )
```

7 Generation

Once the source feature structure has been transferred into an equivalent one in the target language, the next phase involves generating the string of words in the target language.

Fig. 3 shows a simple generation grammar. Each rule contains four elements:

- A context-free portion specifying the subtree which will be generated by this rule, for example:

```
(1:S -> NP VP)
```

- A set of functional equations specifying how to proceed with the generation process for each of the nodes generated by the rule:

```
{@self-1 = @generate(@up.subj);
 @self-2 = @generate(@up);}
```

- In addition, every rule must be preceded by the list of features triggering it:

```
GenerationBlock: pred obj subj ?pobj ?clt_dbl
```

Episteme allows several generation rules to share the same triggering generation features, which leads to the idea of a **generation block**, or set of generation rules. In the example in Fig. 3 generation blocks contained a single rule each. However, real applications demand this capability. For instance, the next example contains a block of more than one rule:

```
GenerationFeatures: pred head spec subj obj eadj
...
GenerationBlock: pred obj
       (24:VP -> v NP manner) @when (@up.manner)
              {@self-1 = @synthesis(@up.pred);
               @self-2 = @generate(@up.obj);
               @self-3 = @synthesis(@up.manner); }

       (25:VP -> v NP)
...
```

The rule above includes the fourth (optional) element in a generation rule:

- A condition of type **@when** which evaluates an expression. Conditions during generation make use of the same specification language and expressions evaluation mechanism designed for unification and transfer.

```
GenerationFeatures:  pred head pcase spec subj obj pobj clt_dbl

GenerationBlock:  pred obj subj ?pobj ?clt_dbl
   (1:  S -> NP VP)
      {@self-1 = @generate(@up.subj);
      @self-2 = @generate(@up);}
GenerationBlock:  head
   (2:  NP -> n)
      {@self-1 = @synthesis(@up.head);}
GenerationBlock:  spec head
   (3:  NP -> det n)
      {@self-1 = @synthesis(@up.spec);
      @self-2 = @synthesis(@up.head);}
GenerationBlock:  pred obj pobj ?clt_dbl
   (4:  VP -> VG NP PP)
      {@self-2 = @generate(@up.obj);
      @self-3 = @generate(@up)pobj);
GenerationBlock:  pred
   (5:  VG -> v)
      {@self-1 = @synthesis(@up.pred);}
GenerationBlock:  pred clt_dbl
   (6:  VG -> clt v)
      {@self-1 = @synthesis(@up.clt_dbl);
      @self-2 = @synthesis(@up.pred);}
```

Generation Triggering Features

May-match (?) Features

Must-match Features

Recursive Generation of Non-Terminal Nodes

Synthesis of Terminal Nodes

Fig. 3 Simple Generation Grammar

Another innovation of Episteme is the inclusion of optional features in the specification of the generation block. This avoids innecesary repetitions in the generation component. For example, rule 1 in Fig. 3 will be used with structures that must contain obligatorily a `pred`, `subj` and `obj` and may contain a `pobj` and/or a `clt_dbl` optionally.

Finally, lexical generation makes use of the same model (Mph–Vtree) described previously for lexical analysis.

8 Episteme at Work

This section illustrates how Episteme may solve a complex translation problem between English and Spanish. Resultative constructions in English include in the predicate the manner in which the action takes place. If the action has a result it is described by means of an adjunct or a secondary predicate. On the contrary, the predicate in Spanish denotes the result and the adjunct denotes the manner.

The blacksmith hammered the metal flat (resultative)

These constructions pose a serious problem to MT systems, since what is a verb in English becomes an adjunct in Spanish and viceversa.

(A) *The blacksmith hammered the metal flat* ->
El herrero aplanó el metal a martillazos

Our analysis is based on the assumption that resultative constructions undergo a process of predicate composition in English [5]. Although our analysis is inspired in LFG, it does not reflect the official trend in the theory.

Resultative constructions are identical in their constituent organisation to so-called *depictive* constructions, in which the adjunct does not describe the result of applying an action over an object, but rather, the state of the object (or subject) when the action took place.

(B) *The waiter served the fish raw* ->
El camarero sirvió el pescado crudo

Therefore, the system must be capable of differentiating each type of construction first, and then apply the corresponding translation rules.

The resultative construction will be identified during analysis taking into account whether the main verb allows this type of construction, signalled by the presence of the (`result:yes`) feature. If so, a new complex predicate will be formed, composed of the main verb and the head of the Adjective phrase. This operation is performed by a special function (`@concat()`), whose result will overwrite (`=a`) the original value of `pred`, assigned by the application of an `@up = self-1` functional equation. If the triggering feature was not found, the Adjective phrase will be analysed as an adjectival adjunct (`aadj`) of the object by default.

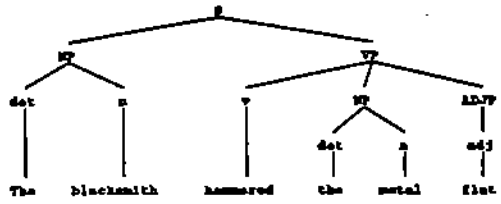
```
(4:VP->v NP ADJP)
       {@up = @self-1;
        @up.obj = @self-2;
        @if (@self-1.result == yes)
        @then {
              @up.pred =n @concat(@self-1.pred,"-",@self-3.pred); }
        @else {
              @up.obj.aadj = @self-3; }
       }
```

The complex predicate *hammer-flat* will be translated as *aplanar* during the transfer phase, and the `manner` feature will hold the manner of action (`manner:"a martillazos"`). The other translation assignments are trivial. The transfer phase inserts gender features as well, which were not present in English.

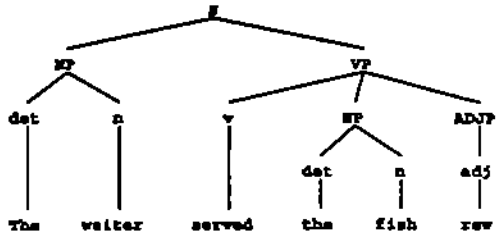
During the generation phase we make sure that the manner adjunct appears right after the verb in Spanish, as the following generation rule shows:

```
GenerationBlock: pred obj
       (4:VP -> v NP manner) @when (@source.manner)
              {@self-1 = @synthesis(@up.pred);
               @self-2 = @generate(@up.obj);
               @self-3 = @synthesis(@up.manner); }
```

The figures below show the c–structures and f–structures resulting from analysing. transferring and generating both examples with Episteme.



(A) C-Structure



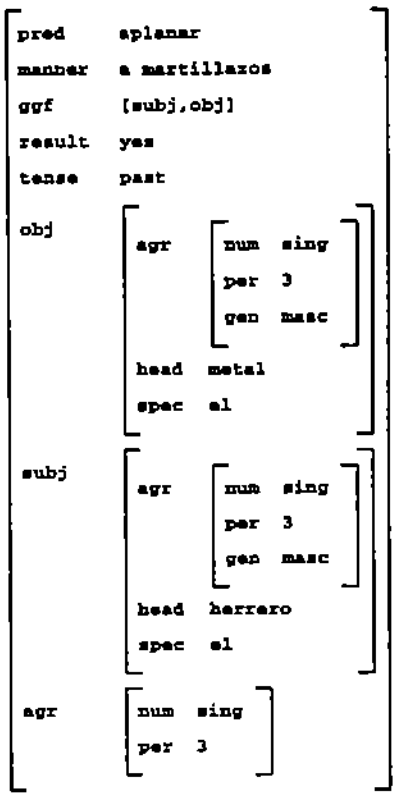
(B) C-Structure

```
ggf      [subj,obj]
pred     hammer-flat
result   yes
tense    past
obj      [ agr   [ num  sing ]
           [ per  3 ]
           head  metal
           spec  the ]
subj     [ agr   [ num  sing ]
           [ per  3 ]
           head  blacksmith
           spec  the ]
agr      [ num  sing
           per  3 ]
```

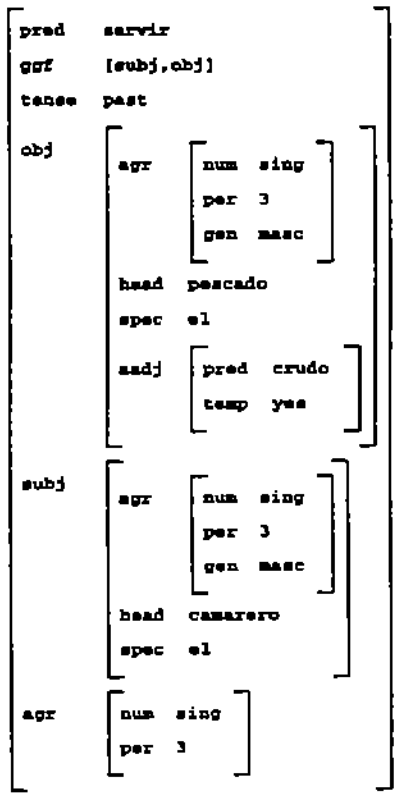
(A) F-Structure

```
ggf      [subj,obj]
pred     serve
tense    past
obj      [ agr   [ num  sing ]
           [ per  3 ]
           head  fish
           spec  the
           aadj  [ pred  raw
                   temp  yes ]
subj     [ agr   [ num  sing ]
           [ per  3 ]
           head  waiter
           spec  the ]
agr      [ num  sing
           per  3 ]
```

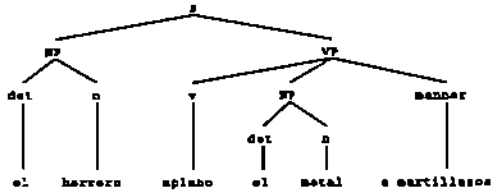
(B) F-Structure



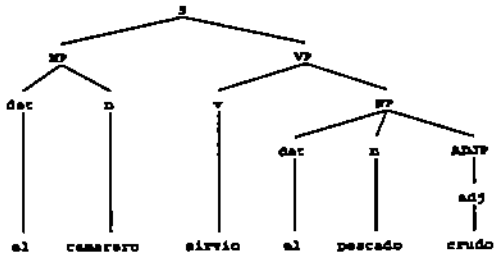
(A) Transfer



(B) Transfer



(A) Generation



(B) Generation

References

[1] Alshawi, H., D.J. Arnold, R. Backofen, D.M. Carter, J. Lindop, K. Netter, J. Tsujii & H. Uszkoreit. 1991. *Eurotra 6/1: Rule formalism and virtual machine design study: Final report*. Technical Report, SRI International, Cambridge.

[2] Amores, J.G. 1992. *A Lexical Functional Grammar-based Machine Translation System for Medical Abstracts*. PhD Thesis. Universidad de Sevilla.

[3] Amores, J.G, J.F. Quesada & D. Tapias. 1994. Traducción Automática basada en el formalismo LFG con entrada y salida por voz. *Comunicaciones de Telefónica I+D*. 5(2). 132–147.

[4] Amores, J.G. 1996. LektaII: Fase de análisis de un prototipo de TA con entrada y salida por voz. En [8]. 199-225.

[5] Amores. G. & G. Alvarez. 1994. Resultative and Depictive Constructions in English: An LFG-Based Approach for Machine Translation. In [16] 335-341.

[6] Bresnan. J. (ed.) 1982. *The Mental Representation of Grammatical Relations.* Cambridge. Mass: MIT Press.

[7] Brown. P.F., J. Cock. S.A. Della Pietra. V.J. Della Pietra. F. Jelinek. J.D. Lafferty. R.L. Mercer y P.S. Roossin. 1990. "A Statistical Approach to Machine Translation." *Computational Linguistics* **16 (2)**. 79- 85.

[8] Instituto Cervantes. 1996. *Estudios computacionales del inglés y del español.* Madrid. Publicaciones del Instituto Cerva: ....

[9] Chandioux. J. MeteoTM: An Operational Machine Translation System. Presentation. RIAO 88 (Manchester Institute of Technology. March 1988).

[10] Dalrymple. M.. R.M. Kaplan. John T. Maxwell III & A. Zaenen (eds.) 1995. *Formal Issues in Lexical-Functional Grammar.* CSLI Lecture Notes No. 47. Stanford: Center for the Study of Language and Information.

[11] Dorr. B. 1990. A Cross-Linguistic Approach to Translation. Austin. TX: *Proceedings of the Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language.* 13-33.

[12] Goodman. K. & S. Nirenburg (eds.) 1991. *The KBMT Project: A Case Study in Knowledge-based Machine Translation.* San Mateo. CA: Morgan Kaufmann.

[13] Kaplan. R. & J. Bresnan 1982. Lexical-Functional Grammar: a formal system for grammatical representation. In [6]. 173-281.

[14] Kaplan. R. M. & A. Zaenen. 1995. Long-distance Dependencies. Constituent Structure. and Functional Uncertainty. In [10]. 137-165.

[15] Koskenniemi. K. 1984. Morphology with Two Level Rules and Negative Rule Features. *Proceedings of COLING-88.* Budapest. Hungary.

[16] Martin Vide. ed. 1994. *Lenguajes Naturales y Lenguajes Formales X.* Barcelona: Ediciones y Promociones Universitarias.

[17] Van Noord. G.. J. Dorrepaal. P. van der Eijk. M. Florenza. H. Ruessink & L. des Tombe. 1991. An Overview of MiMo2. *Machine Translation* 6 **(3)**. 215-228.

[18] Quesada. J.F. & G. Amores 1995. A Computational Model for the Efficient Retrieval of Very Large Structure-Based Knowledge Bases. *Proceedings of KRUSE Symposium.* University of California at Santa Cruz.

[19] Quesada. J.F. 1996. Un Modelo Robusto y Eficiente para el Análisis Sintáctico de Lenguajes Naturales mediante Arboles Múltiples Virtuales. *Procesamiento del Lenguaje Natural*, **19**, 14-29.

[20] Quesada. J.F. 1996. Unificación Constructiva. Estratégica, con Compartición de Estructuras y Post-Copia. *Procesamiento del Lenguaje Natural*, **19**. 148-158.

[21] Sharp, R. 1988. CAT2 - Implementing a Formalism for Multi-lingual MT. In *Proceedings of the 2nd International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language.* Pittsburgh,PA.

[22] Slocum. J. ed. 1988. *Machine Translation Systems.* Cambridge: Cambridge University Press.

[23] Vasconcellos. M. & M. León. 1988. SPANAM and ENGSPAN: Machine Translation at the Pan American Health Organization." In [22]. 187-235.