Nominalizations in the LKB Framework

Irene Castellón Masalles

Report LSI–93–45–R

NOMINALIZATIONS IN THE LKB FRAMEWORK

Irene Castellón Masalles
Universtitat Politècnica de Catalunya
e-mail: cirene@lsi.upc.es

NOMINALIZATIONS IN THE LKB FRAMEWORK

Irene Castellón Masalles
cirene@lsi.upc.es
Universitat Politècnica de Catalunya

Abstract

One of the main purposes in current computational lexicography is to determine methods to allow the acquisition of lexical knowledge. One of the facilities of the Acquilex Knowledge Base is the possibility of defining lexical rules. These rules are used for generating new lexical entries from the existing ones, capturing linguistic generalizations which will be reflected in the new entries. In this paper we focus on one kind of lexical rules which allows the generation of noun entries from verbs. We present the nominalization problem in Spanish and how to represent it in LKB. The nouns created refer to the agent, patient and action of the origin verbs.

NOMINALIZATIONS IN THE LKB FRAMEWORK

Irene Castell6n Masalles
Universtitat Politècnica de Catalunya
e-mail: cirene@lsi.upc.es

1. Introduction

One of the main purposes in current computational lexicography is to determine methods to allow the acquisition of lexical knowledge.The reusability of already existing conventional sources as MRDs and analized Corpora seems adequate to this purpose. This methodology has been followed in Acquilex I and II[1] and by now we have available a partial Lexical Knowledge Base built up following this methodology.

One of the facilities of the Acquilex Knowledge Base is the possibility of defining lexical rules. These rules are used for generating new lexical entries from the existing ones capturing linguistic generalizations which will be reflected in the new entries. The rules allow the modification of the form, syntax and semantic content of source entries.

In this paper we focus on one kind of lexical rules which allows the generation of noun entries from verbs, we present the nominalization problem in Spanish and how to represent it in LKB. The nouns created refer to the agent, patient and action of the origin verbs.

2. Lexical Rules

A lexical rule[2] is defined in the LKB as a feature structure that represents two lexical signs. One of them codifies the input and the other the output sign of the rule. A lexical rule makes possible to change the content of a sign, and in this way, to generate a new sign, so this mechanism allows the creation of new entries from the existing ones.

The changes that a lexical rule performs should refer to any kind of information contained in the lexical entry. Examples of different kind of lexical phenomena that can be captured by means of these rules are: the generation of the plural in nouns involving both orthographical and morphological changes, the verbal inflection (present, conditional, etc.), the

[1] [Acquilex 89], [Acquilex-II 92].

[2] [Copestake 91]

verbal diathesis (syntactic uses), derivational phenomena (pan -- panadero, subir --subida), semantic changes as metaforic uses (animal-person), grinding (animal-meat),etc..[3]

A basic lexical rule is represented in the LKB as:

```
lexical-rule (rule)

  < 0 > = lex-sign
  < 1 > = lex-sign.
```

The feature <1> codifies the input sign and <0> the output one.

A Lexical rule allows to establish some **restrictions** in its application. This is performed by the assignment of specific values in the input lexical sign. As an example we can consider the plural lexical rule:

```
plural
  lexical-rule
  < 1 > = lex-count-noun
  < 1 : cat : m-feats : agr : num > = sg
  < 0 : cat : m-feats : agr : num > = pl
  < 0  > = lex-count-noun
  < 0  : orth : orth1 > = < 1  : orth >
  < 0  : orth : orth2 > = "+s"
  < 1 : sem > = < 0  : sem >
  < 1 : rqs > =  < 0  : rqs >.
```

The restriction established by this rule is that the input sign has to be both a countable and a singular noun. This type of specifications allows the blocking of the rule application.

Another facility allowed by lexical rules is the possibility of adding new values to the resulting entry. The values are always conditioned by the input sign. In the example above the new values are displayed in the <0> feature. The new sign would be a plural countable noun and its form is characterized by the plural suffix '+s'.

A lexical rule makes also feasible to establish equalities between both signs, and so the information from the input sign will be added to the output one:

```
< 1 : sem > = < 0  : sem >
< 1 : rqs > =  < 0  : rqs >.
```

[3] [Copestake 91]

3. Nominalization in Spanish

In Spanish, nominalization can be performed by means of several morpholexical processes, mainly by adding an article before the infinitive form of the verb or deriving nouns from verb forms:

verb	noun
comer (to eat)	(el) comer
comer	(la) com-ida
generar (to generate)	(el) generar
generar	(la) gener-ación

Here we will focus on derivational mechanisms. Derivation consists on the addition of a suffix (in this case a nominal one) to a stem producing a new lexical item. The addition of the suffix is conditioned to the POS of the source entry. The derivational suffix determines the POS and other syntactic and semantic features of the resulting one:

correr (verb)	corr-edor (noun)
comer (verb)	com-ida (noun)
trampa (noun)	tramp-oso (adj)
fresco (adj)	fresc-ura (noun)

Suffixes can often have more than one meaning. In the example above 'corr-edor' means both the 'actor' and the 'place' where the action takes place. These ambiguities can be dealt with by defining one lexical rule for each meaning.

Derivation is a very productive lexical mechanism in Spanish but, to the contrary to inflection, it is not systematic, that is, there is not explicit information in the source form specifying which subset of derivational suffixes it accepts. Inflection groups lexical items in classes but derivation is a phenomenon that has to be dealt particularly for each stem, thus the combination of stems and suffixes doesn't depend on the stem form or on its meaning. The only restriction that can be explicity stablished affects the POS both of stems and suffixes, i.e.'-ador', '-ción', '-miento', etc. produce nouns from verbal stems and '-ura', '-or' produce nouns from adjective stems.

This fact would constrain the application of lexical rules, in the sense that some forms woud be blocked.

3.1. Nominalization suffixes

In Spanish it is possible to derive adjectives and nouns and other verbs from verbs. Here we focus on derivative nouns. Nominal derivates can express the name of the activity or state or the name of an argument: the 'action' ("generaci6n" from "generar"), the 'result' of the action (as "firma" from "firmar"), the agent of the action (as "escritor" from "escribir"), the patient of the action as ("cocido" from "cocer"), the place where the action is taking place, etc.:

VERB	actions	results	agents	patients	places
firmar	firma	firma	firmante	firmado	-
comer	comida	comida	comedor	comida	comedor
destruir	destrucci6n	destrucci6n	destructor	destruido	-
mirar	mirada	mirada	mirador	mirado	mirador

We centre our study on 'actions', 'agents' and 'patients'. The action-nouns and result-nouns share the same form in many cases. The main difference between action nouns and result nouns is their semantic content: a result-noun acts as a pure noun, that is, it can have noun complements; whereas an action-noun has semantic arguments, like a verb . This phenomena can produce ambiguities in the interpretation of a NP[4] , for example:

1.a "La firma del contrato no es válida"
1.b "La firma del contrato por el gerente..."

In 1.a the noun 'firma' refers to the result-noun and so 'del contrato' acts as a noun-complement, in 1.b 'firma' refers to the action, 'del contrato' is a noun which acts as patient of the action and "por el gerente" is a pp acting as agent.

Each nominalization class is performed by different suffixes, such as '-ci6n', '-ida', '-miento', etc. for "action", '-nte', '-dor', etc.for "agent". Each stem selects more than one, one or zero suffixes in each class. For instance 'cantar' (to sing) accepts as agent 'cantante' as well as 'cantor'. The selection of the suffix is not determined by any feature stem, thus we can characterize this process as free one. This fact raises some problems in the generation process, because lexical entries lack of the information necesary for selecting the correct derivatives.

Besides, another added difficulty is the overlaping of suffixes belonging to different classes or different POS. For instance, '-edor' can mean both the actor and the place of the action, i.e.: "comedor" means both 'the agent of eating' and the 'dining-room'; ' -ida-' produces nouns ("sub-ida","desped-ida") as well as adjectives ("dorm-ida", "perd-ida").

[4] In [Grimshaw 90] we can find a test to detect it.

As a final remark we want to point out that grammaticality in derivates is a problem of degree. So we can distinguish between correct, acceptable, possible and understandable forms, but in many cases it is difficult to decide to which of these classes a derivate belongs. This faces us to the problem of how to discriminate forms that we will be acceptable as nominalizations. This question is discussed in the next section.

4. How to build the nominalization entries

In this section we present the different methods for building nominal lexical entries from verbs. In the first part we explain how to build the lexical entries starting from taxonomies[5], in the second part we propose what would be an "ideal" solution and finally we explain the methodology currently adopted. We present the different strategies with the example of 'action nouns', although the agent nouns and patient nouns differ in some points, the strategy should be the same.

A) One possible way of building the noun lexical entries derived from verbs is to extract all the dictionary entries defined as "acci6n de X..." (action of X...) and build the taxonomy of "action" following a semiautomatic process (like all the lexicons built in Acquilex I in Barcelona[6] with the SEISD enviroment). This is a top-down methodology starting with the top "acci6n"(action) and looking for its hyponymyc senses . The resulting taxonomy is simply a list of nouns specified as "actions". Then in the dictionary we will have only one step in depth. An example of definition follows:

delegaci6n [l. -atione]
acepci6n:1 ** f. ** Acci6n de delegar.

This definition is not an explanation but is simply a reference to the verbal entry. So, the following step is to analyze the definitions of the taxonomy in order to detect the verb (in the example : 'delegar') and then to consult the verbal entry. This method involves a desambiguation process as there can be more than one sense attached to the verb form. Following the example, in the dictionary there are two senses for 'delegar':

delegar [l. -are]
acepción:1 ** tr. ** Dar una persona {a otra} la facultad
o poder que aquélla tiene para que haga sus veces: el
tribunal delegó un juez para instruir el sumario.

[5] In Acquilex I our method for extracting and representing lexical entries started from building taxonomies [Ageno 92].

[6] [Ageno 92].

acepción:2 ** tr. ** Transferir {el poder o autoridad de uno} a otra persona: ~ sus poderes a fulano; ~ la presidencia de una junta a un vocal.3

Once the desambiguation process is accomplished,we can transfer the verbal information to the nominalization entry by means of lexical rule application. Thus this first methodology consists on the following steps:

1.- to build the taxonomy of "action" to isolate the 'action nouns' (the same for agent and patient noun)
2.- desambiguation of the verb sense refered to in the definition.
3.- lexical rule application.

B) The second method we propose, as we mentioned above, is to use, in a first step, lexical rules to generate the nominalizations from verbal entries and to apply a strategy to validate the new forms. We use lexical rules to build this subset of lexicon because the source information is very reduced (in our case, for the moment, it's a Spanish dictionary). Some examples of nominalization definition in Vox follows:

"subida I.1: accion de subir." (action of going up)
"firma I.3: acto de firmarlos" (action of signing them)
"señalización I.1: acción de señalizar" (action of signposting)

These definitions refer to the verbs "subir", "firmar" and "señalizar" to explain the meaning of the action. This pattern ("accion de" + Verb) applies to most of the nominalization definitions. So we have chosen to generate the nominal entry by means of a lexical rule, because the information is more complete and the form of the nominalization entry can be calculated from the verbal entry form.

This methodology requires a filter for the application of the rules. This filter consists in adding morfological information in the LKB entry in order to determine and discriminate the application of the lexical rules by means of the blocking of specific rules. This way involves enriching the LKB with morphological information. To carry out this we can adopt a manual or automatic methodology. A manual acquisition is contradictory with the line of developement adopted by the Acquilex project and it is very expensive in time and human efforts. On the contrary, automatic acquisition would be desiderable, but the MRD[7] has not information about this specific derivational phenomenon, and it would be necessary to consult other sources such as a morphological analyzer.Therefore this methodolgy involves two steps:

[7] The VOX dictionary [Vox 87].

1- to enrich the morphological LKB information
2- to generate the new entries by means of lexical rule application.

This solution seems better because it allows the application of lexical rules and should generate only the correct entries. But currently it is not feasible[8] and we had to explore other strategies to validate the new forms generated.

C) Another way, the one we have adopted, is to make the process in two steps: generation and validation. A first step, generation, will consist on generating the new entries by means of the application of all the nominalization rules to each verbal entry and then, in a second step, the validation one, we will consult the new forms in an external source such as different MRDs or Corpora to validate them. This solution implies to consult several sources and therefore the lexicon is not limited to the one that sould be validated consulting a single MRD.

So we generate all the possible nominalizations from all the verbal entries represented in LKB by means of lexical rule application. Once we have created the nominalization entries, a query to a source (dictionary or corpora) can validate an entry. This methodology consists on:

1) a generation process
2) a validation process, consulting different sources.

We have selected this methodology because it presents several advantages. Let us point out some of them:

a)- it avoids the building of taxonomies that are not relevant from an ontologic point of view;

b)- it allows the query to other sources (in the first way the only source is the dictionary).

c)- it supports the theory of basic lexicon[9], in the sense that the lexical rules expand it, generating new entries.

d)- it is fully automatic

e)- the strategy doesn't differ in the case of other nominalizations (for instance : agent and patient nouns).

In figure 1 we can see a scheme of the strategy that we will follow.

[8] We have not an enviroment or system to do so.

[9] [Pustejovsky 91]

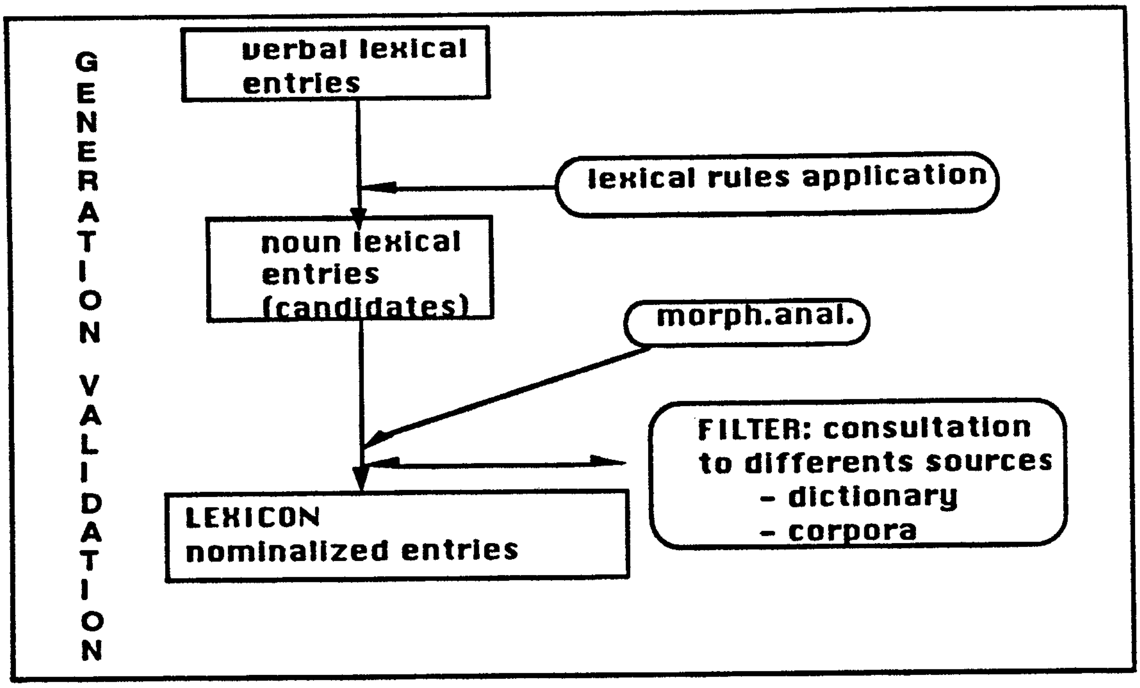

GENERATION VALIDATION
verbal lexical entries
lexical rules application
noun lexical entries (candidates)
morph.anal.
FILTER: consultation to differents sources
- dictionary
- corpora
LEXICON nominalized entries

Figure 1: general strategy

5. From verbs to nouns

As we have mentioned above, this work deals with three kinds of nominalization: actions, agents and patients.They differ in the specific lexical rules that we must apply to the verbal entries so, we describe them in different sections. First we deal with action nouns (section 5.1) , next we explain the agent (5.2.1) and finally patient nouns (5.2.2). Previously we mention the characteristics that they have in common.

The change of category (POS) is the main difference between a verbal entry and its nominalization: a nominal sign derived from a verbal sign will have noun as category.

Thus we can asume that all the nominalizations will have the following equation:

<1 : cat > = complex-cat
<0 : cat : cat-type> = n

This restriction is defined in the input and output sign.

5.1.- Action nouns

The main characteristic of an action noun in front of other nouns is that it has a complex semantic representation. Lets us see an example where there is a verbal phrase and a nominal phrase that denotes the same action:

(sentence) "X moviliza Y"
(X mobilizes Y)

(nominal phrase) "La movilizacion de Y por X"
(The mobilization of Y by X)

From a semantic point of view, these structures have the same representation. It's clear that both expressions denote the same action, the only difference is its syntactic feature. In the first case we deal with a complete sentence, in the second case we deal with a noun phrase that can be a part of a sententce. However both, the nominal phrase and the verbal phrase, have a binary formula with two arguments in the semantic representation, the agent and patient.

The lexicalization of the arguments depends on the head form. In the case of verbal heads the arguments are noun phrases and in the case of noun heads (nominalization) the arguments are prepositional phrases.

Therefore, neither the verbal sign nor the noun sign can represent the action-nouns. In the first case, a verbal sign has a complex category as syntactic value and in the second case, a noun has a unary semantic representation and so it cannot represent arguments. Thus we need to define a new sign in the LKB that allows to represent these nominalization entries: deverb-noun-sign. It is also necessary to define the lexical rules to copy the verbal semantic information (formula) into the new entry (nominalization).

5.1.1. Deverb-noun-sign

'Deverb-noun-sign' is a type that represents the nominalization sign. As all the lexical signs, it has orthographical, syntactic and semantic information. Because it is a noun it would also be necessary to include a rqs feature with the value 'action' [10]. We focus our attention in the syntactic and semantic features that are entirely defined

In the syntactic information, the nouns resulting from a nominalization process, the 'deverb-noun-sign' type, have "noun-cat" as value of the feature cat."Noun-cat" is a simple category (not a compose one like 'np-raised' in 'verb-sign') because nominalized entries act as common nouns. In 'm-feats' the feature 'nominal-form' has "deverbal" as value in order to diferentiate simple nouns form nouns derived form verbs. Semantically, 'deverb-noun-sign' has a binary formula as value of its feature <sem>.

[10] We are currently developing the 'abstract' type classification of 'nomrqs' where 'action' will be included. We have also to define how the verbal rqs is transformed in noun rqs.

Figure 2 shows the specifications of a type 'deverb-noun-sign'

```
deverb-noun-sign (lex-sign)
    <cat> = noun-cat
    <cat : m-feats : nominal-form> =deverbal
    <sem > = binary-formula
    <sem : arg1 > = eve-noun-formula.
```

Figure 2.- Deverb-noun-sign.

The level of representation of 'deverb-noun-sign' is the same as the sign "verb-sign'. We don't specify the number of arguments for this sign, only the category and the kind of formula being specified. So we have built the classification of deverb signs in a parallel way to the classification of verbs.

We distinguish two subtypes of 'deverb-np-sign': 'deverb-intrans-sign' and 'deverb-trans-sign'. The first class defines the nominalization that derives from intransitive verbs. This is the case of the Spanish noun "buceo" :

buceo
acepci6n:1 ** m. ** acci6n de **bucear.**(action of diving.)
(bucear
acepci6n:1 ** intr. ** Nadar o mantenerse debajo del agua,
conteniendo el resuello.)

"Buceo" as its corresponding verb ('bucear') are intransitive, therefore its semantic formula represents only an argument, the p-agent, so it's necessary an specific sign to represent it. Figure 3 shows the 'deverb-intrans-sign' specification.

```
deverb-intrans-sign (deverb-noun-sign)
 < sem : arg2 : prep-formula arg1> = p-agt-formula
 < sem: arg2: arg1 :ind> = < sem : arg2 : ind>
 < sem: arg2: arg1 : ind> = < sem: arg1: ind>.
```

Figure 3: deverb-intrans-sign

In a similar way it's necessary to define the sign corresponding to the nominalizations of transitive verbs. These nominalizations wil have two arguments corresponding to p-agent and p-patient [11]. Figure 4 shows 'deverb-trans-sign'.

```
deverb-trans-sign (deverb-noun-sign)
 <sem : arg2 : arg1 : prep-formula arg1> = p-agt-formula
 <sem : arg2 : arg2 : prep-formula arg1> = p-pat-formula.
```

Figure 4: deverb-trans-sign

Both signs express their arguments by means of a formula (p-agent or p-patient) embedded in a prepositional formula. This is because, in a noun phrase whose head is an action-noun, the action arguments are expressed by a prepositional phrase . A prepositional formula is an unary formula composed by: a preposition as 'pred' value, an index with the value 'event' and an arg1 with a theta-formula as value describing the thematic argument.

```
prep-cn-formula (unary-formula)
<pred> = string
<ind> = eve
<arg1> = theta-formula.
```

5.1.2.Deverb Lexical Rules

We have developed a class of lexical rules in order to generate deverbal entries. This class (deverb-lex-rule) has two daughters: deverb-intrans-lex-rule and deverb-trans-lex-rule. We think that these rule types should also be added to the "link.types" file with the purpose of generalizing some common characteristics of nominalizations . Besides we have developed a lexical-rules file where we specify the orthographical information as well as other semantic and syntactic features that are specific to each language. In appendix 1 there is a proposal of some new types that should consequently be added to the current type system . We can observe as an example the type 'deverb-trans-lex-rule', an specification of 'deverb-lex-rule':

```
deverb-trans-lex-rule (deverb-lex-rule)
<1> = strict-trans-sign
<0> = deverb-trans-sign
<1 : sem : arg2 : arg1> = <0 : sem : arg2 : arg1 : arg1>
```

[11] Up to now we only have defined 'deverb-intrans-sign' and 'deverb-trans-sign' as subtypes of 'deverb-noun-sign', but it is clear that it will be necessary to go in depth in the 'deverb' classification.

<1: sem: arg2: arg2> = <0 :sem: arg2: arg2: arg1>

In this type of rule we have expressed the restrictions involving the input and output sign: the input must be a transitive verbal entry (as “comer”) and the output will be a deverbal transitive sign. In this type we also specify that the arguments of transitive verbs are the same for the action noun, so we transfer the argumental selection.

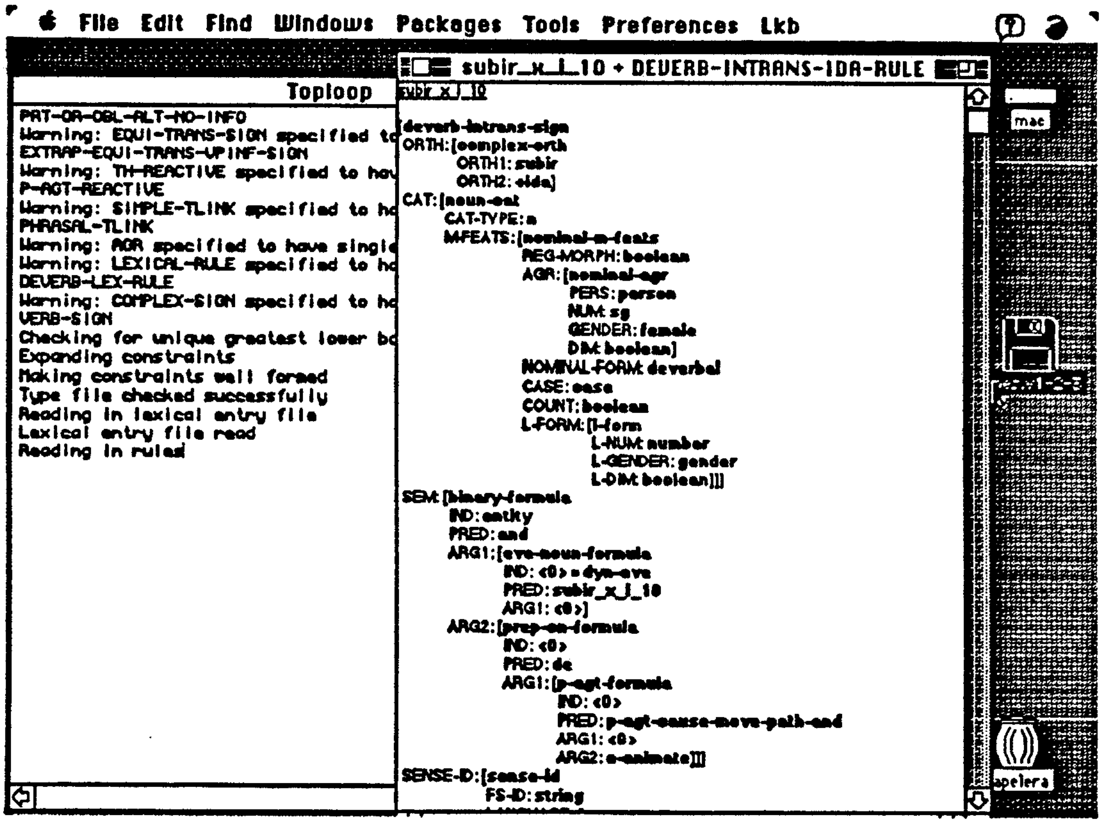
The lexical rules file , as we mentioned above, is more dependent on each language because in it we express changes referent to surface information (orthographical and morphological ones).

Here follows an example of nominalization rules. ‘Deverb-intrans-ida-rule’ generates a deverb-np-sign (as "subida") from an intransitive verbal entry (as "subir_X_1_1"), adding the suffix ‘-ida’ to the verbal stem:

```
deverb-intrans-ida-rule
    deverb-intrans-lex-rule
    <0 : orth : orth2> = "+ida"
    <0 : cat : m-feats : agr : num> = sg
    <0 : cat : m-feats : agr : gender> = female
    <0 : sem : arg2 : prep-cn-formula pred> = "de".
```

In this rule we express that the nominalization suffix is ‘-ida’ and its morphological values are singular for number and female for gender. Finally the rule specifies that the argument (p-agent) is introduced by the preposition ‘de’.

Figure 5 shows the result of applying this rule to the verbal entry **subir_X_1_1**:



```
PRT-OR-OBL-ALT-NO-INFO
Warning: EQUI-TRANS-SIGN specified to
EXTRAP-EQUI-TRANS-VPINF-SIGN
Warning: TH-REACTIVE specified to hav
P-AGT-REACTIVE
Warning: SIMPLE-TLINK specified to ha
PHRASAL-TLINK
Warning: AGR specified to have single
Warning: LEXICAL-RULE specified to ha
DEVERB-LEX-RULE
Warning: COMPLEX-SIGN specified to ha
VERB-SIGN
Checking for unique greatest lower bo
Expanding constraints
Making constraints well formed
Type file checked successfully
Reading in lexical entry file
Lexical entry file read
Reading in rules
```

```
subir_x_l_10

[deverb-intrans-sign
ORTH:[complex-orth
      ORTH1:subir
      ORTH2:ida]
CAT:[noun-cat
     CAT-TYPE:n
     M-FEATS:[nominal-m-feats
              REG-MORPH:boolean
              AGR:[nominal-agr
                   PERS:person
                   NUM:sg
                   GENDER:female
                   DIM:boolean]
              NOMINAL-FORM:deverbal
              CASE:case
              COUNT:boolean
              L-FORM:[l-form
                      L-NUM:number
                      L-GENDER:gender
                      L-DIM:boolean]]]
SEM:[binary-formula
     IND:entity
     PRED:and
     ARG1:[eve-noun-formula
           IND:<0> = dyn-eve
           PRED:subir_x_l_10
           ARG1:<0>]
     ARG2:[prep-on-formula
           IND:<0>
           PRED:de
           ARG1:[p-agt-formula
                 IND:<0>
                 PRED:p-agt-cause-move-path-end
                 ARG1:<0>
                 ARG2:e-animate]]]
SENSE-ID:[sense-id
          FS-ID:string
```

Figure 5: application of deverb-trans-ida-rule to subir_X_1_1

Sometimes we cannot specify the value of the preposition and therefore some general rules don't specify this value. An example of this rules is given in appendix 2

5.2. Agents and Patients nominalizations

We have extended the study of nominalization to the agent and patient nouns derived from verbs. The main difference between these nouns and the noun action is that the first ones act as a pure noun, that is to say, they have not argument structure, and so can be represented as a lex-noun-sign, with a basic category and with a simple (unary) semantic formula, while the second one behaves as a verb. Therefore we only had to develop the rules that generate an agent or a patient from a verb.

5.2.1. Agent nouns

An agent noun can be derived from a a transitive or an intransitive verb, and the p-agent argument can be extracted from the verbal entry by means of a lexical rule. We can transfer orthographical, morfological and semantic information from the p-agent argument to its nominalization. For example:

(a) El cocinero prepara la ensalada (The cook prepares the salad)

In the LKB lexicon we have declared the verb 'preparar' as a transitive verbal lexical entry which has an NP as p-agent argument (declared in the syntactic part) with several morphologic and semantic features:

```
prepara X_I_1
   STRICT-TRANS-SIGN
   < lex-sign sense-id : sense-id dictionary > = ("VOX")
   < lex-sign sense-id : sense-id word > = ("preparar")
   < lex-sign sense-id : sense-id homonym-no > = ("I")
   < lex-sign sense-id : sense-id sense-no > = ("1")
   < orth > = 'preparar'
   < cat : active: np-sign  cat : cat-type > = n
   < cat : active: np-sign  cat:  m-feats : agr : num > = number
   < cat : active: np-sign  cat : m-feats : agr : gen> = gender
   < sem : arg1 : arg1 > = proc
   < sem : arg2 : arg1 : pred > = p-agt-cause
   < sem : arg2 : arg1 : arg2 > = e-human
   < sem : arg2 : arg2 : pred > = p-pat
   < sem : arg2 : arg2 : arg2 > = e-inanimate.
   < rqs > = cook.
```

Features in bold caracterized the agent noun. They are the orthographical information 'preparar', the morphological and syntactic features as 'n', 'number', 'gender' (not specify in this entry) and the semantic restriction 'human'. ?? caracterize the agent noun and, thus, these features must be transfered to the new sign.

The values from the genetared entry are sometimes conditioned by the values of the input sign. The agent noun derived from some classes of verbs (like "cook") have the source verb as value of the "telic"[12] feature in "rqs" (see figure 7).

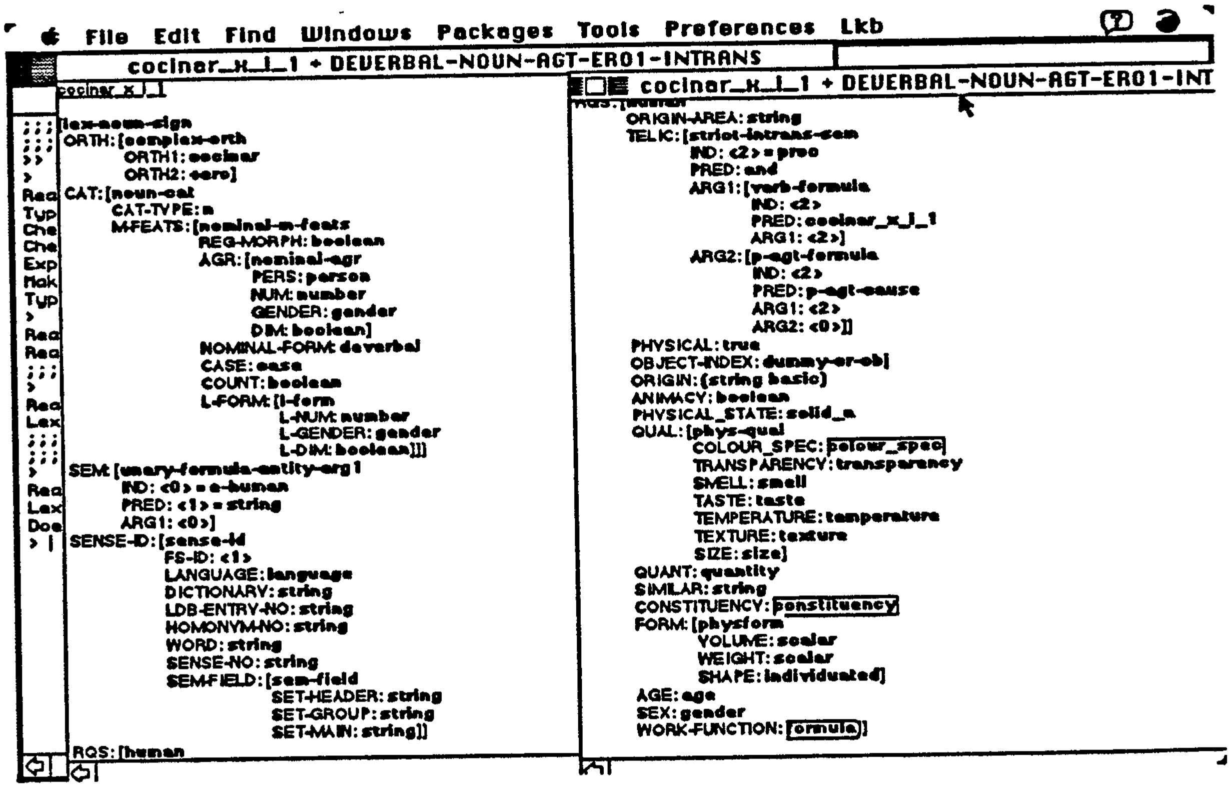
Another way to producing new values is the assignment of values in the lexical rule. That is the case of the value 'deverbal' for the feature 'nominal-form', or the assignment of the suffix orthographical form (-ante, -ero...). Figure 6 shows a lexical rule for producing derived agent nouns by adding the suffix '-ero' to the verbal stem.

[12] [Pustejovsky 91]

```
deverbal-noun-agt-ero1-intrans
deverbal-noun-agt-intrans
<1> = strict-intrans-sign
<0> = lex-noun-sign
< 0: orth: orth1> = <1: orth>
< 0: orth: orth2> = "+ero"
<0: cat: m-feats: agr> = <1: cat: active: cat: m-feats: agr>
<0: cat: m-feats: nominal-form> = deverbal
<1: sem: arg2: arg2> = <0: sem: ind>.
<1: sem> = < 0: rqs: telic>
<1 : rqs> = cook
 <0: rqs> = human.
```

Figure 6: lexical rule of agent noun .

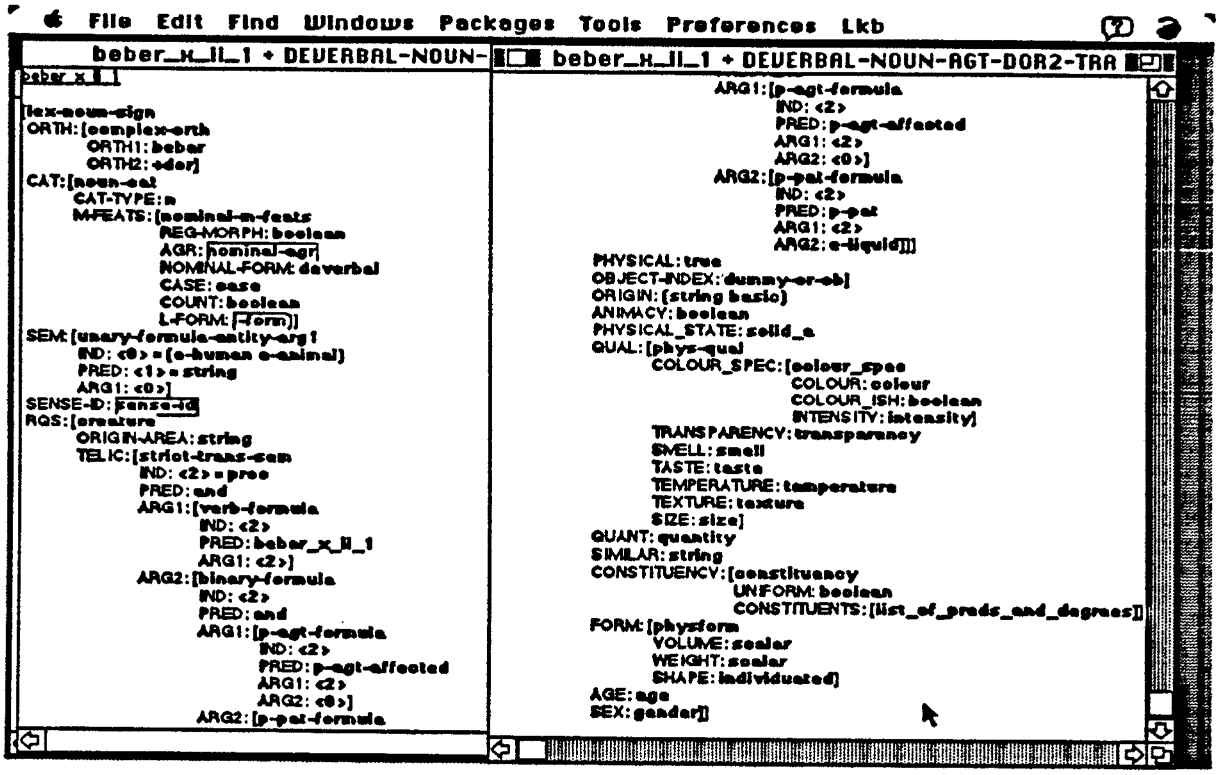
Figures 7 and 8 shows an agent rule application to the verbs 'cocinar' and 'beber'.


File
Edit
Find
Windows
Packages
Tools
Preferences
Lkb
cocinar_x_i_1 + DEVERBAL-NOUN-AGT-ERO1-INTRANS
cocinar_x_i_1
cocinar_x_i_1 + DEVERBAL-NOUN-AGT-ERO1-INT

```
[lex-noun-sign
ORTH:[complex-orth
        ORTH1:cocinar
        ORTH2:+ero]
CAT:[noun-cat
      CAT-TYPE:n
      M-FEATS:[nominal-m-feats
                 REG-MORPH:boolean
                 AGR:[nominal-agr
                       PERS:person
                       NUM:number
                       GENDER:gender
                       DIM:boolean]
                 NOMINAL-FORM:deverbal
                 CASE:case
                 COUNT:boolean
                 L-FORM:[l-form
                          L-NUM:number
                          L-GENDER:gender
                          L-DIM:boolean]]]
SEM:[unary-formula-entity-arg1
      IND: <0> = e-human
      PRED: <1> = string
      ARG1: <0>]
SENSE-ID:[sense-id
           FS-ID: <1>
           LANGUAGE:language
           DICTIONARY:string
           LDB-ENTRY-NO:string
           HOMONYM-NO:string
           WORD:string
           SENSE-NO:string
           SEM-FIELD:[sem-field
                       SET-HEADER:string
                       SET-GROUP:string
                       SET-MAIN:string]]
RQS:[human
```

```
     ORIGIN-AREA:string
     TELIC:[strict-intrans-sem
             IND: <2> = proc
             PRED:and
             ARG1:[verb-formula
                    IND: <2>
                    PRED:cocinar_x_i_1
                    ARG1: <2>]
             ARG2:[p-agt-formula
                    IND: <2>
                    PRED:p-agt-cause
                    ARG1: <2>
                    ARG2: <0>]]
      PHYSICAL:true
      OBJECT-INDEX:dummy-or-obj
      ORIGIN:(string basic)
      ANIMACY:boolean
      PHYSICAL_STATE:solid_a
      QUAL:[phys-qual
             COLOUR_SPEC:colour_spec
             TRANSPARENCY:transparency
             SMELL:smell
             TASTE:taste
             TEMPERATURE:temperature
             TEXTURE:texture
             SIZE:size]
      QUANT:quantity
      SIMILAR:string
      CONSTITUENCY:constituency
      FORM:[physform
             VOLUME:scalar
             WEIGHT:scalar
             SHAPE:individuated]
      AGE:age
      SEX:gender
      WORK-FUNCTION:formula]
```

Figure 7: 'cocinero' from 'cocinar'.



```
beber_x_II_1 + DEVERBAL-NOUN-
beber_x_II_1

[lex-noun-sign
 ORTH:[complex-orth
        ORTH1:beber
        ORTH2:+dor]
 CAT:[noun-cat
       CAT-TYPE:n
       M-FEATS:[nominal-m-feats
                 REG-MORPH:boolean
                 AGR:nominal-agr
                 NOMINAL-FORM:deverbal
                 CASE:case
                 COUNT:boolean
                 L-FORM:l-form]]
 SEM:[unary-formula-entity-arg1
       IND: <0> = (e-human e-animal)
       PRED: <1> = string
       ARG1: <0>]
 SENSE-ID:sense-id
 RQS:[creature
       ORIGIN-AREA:string
       TELIC:[strict-trans-sem
               IND: <2> = proc
               PRED:and
               ARG1:[verb-formula
                      IND: <2>
                      PRED:beber_x_II_1
                      ARG1: <2>]
               ARG2:[binary-formula
                      IND: <2>
                      PRED:and
                      ARG1:[p-agt-formula
                             IND: <2>
                             PRED:p-agt-affected
                             ARG1: <2>
                             ARG2: <0>]
                      ARG2:[p-pat-formula
```

```
beber_x_II_1 + DEVERBAL-NOUN-AGT-DOR2-TRA
                      ARG1:[p-agt-formula
                             IND: <2>
                             PRED:p-agt-affected
                             ARG1: <2>
                             ARG2: <0>]
                      ARG2:[p-pat-formula
                             IND: <2>
                             PRED:p-pat
                             ARG1: <2>
                             ARG2:e-liquid]]]
       PHYSICAL:true
       OBJECT-INDEX:dummy-or-obj
       ORIGIN:(string basic)
       ANIMACY:boolean
       PHYSICAL_STATE:solid_a
       QUAL:[phys-qual
              COLOUR_SPEC:[colour_spec
                            COLOUR:colour
                            COLOUR_ISH:boolean
                            INTENSITY:intensity]
              TRANSPARENCY:transparency
              SMELL:smell
              TASTE:taste
              TEMPERATURE:temperature
              TEXTURE:texture
              SIZE:size]
       QUANT:quantity
       SIMILAR:string
       CONSTITUENCY:[constituency
                      UNIFORM:boolean
                      CONSTITUENTS:[list_of_preds_and_degrees]]
       FORM:[physform
              VOLUME:scalar
              WEIGHT:scalar
              SHAPE:individuated]
       AGE:age
       SEX:gender]]
```

Figure 8: 'bebedor' from 'beber'.

In summaring we have applied tree kinds of operations to generate agent nouns from verbs: restrictions, tranfers and assignements. This operations are caracterized as follows:

- restrictions

    1.- the input sign must be a verb sign.

    2.- the output sign must be a lexical noun sign.

- transfers

    1.- the semantic agent of the verbal formula is transferred to 'noun' semantic formula.

    2.- the semantic formula of the verb sometimes may be tranfer to the value of the feature 'telic' of the 'rqs' feature of the agent noun.

    3.- the morphological information associated to the active sign of verb category is transferred to the morphological features of the noun.

- assignments

    1.- the value 'deverbal' is assigned as value of 'nominal-form' feature of the new sign.

    2.- the suffix value is assigned to the orthographical feature.

5.2.2. Patient nouns

Patient nouns can be extracted partially from the verbal entry .When we talk about patient nouns we refer to the nouns affected by the verbal action, i.e.: 'bebida' from 'beber', 'comida' from 'comer', 'tostada' from 'tostar'.

In this case we apply the same kind of operations as in the agent nouns. Firstly a patient noun can only be derived from a transitive verb (or from an specification of it). Therefore, the first requeriment for all lexical rules of patient nouns is that the input must be a 'strict-trans-sign'. Furthermore, the output will be a 'lex-noun-sign' because a patient noun is a pure noun with an unary formula as semantic value.

The trasferred information is a) some infromation contained in the patient formula of the verbal entry , b) the morphological information of the active sign of the category ((N/S)/N)) specified in the morphological features of the new sign. Depending on the semantic class of the verbal entry, we can transfer the semantic formula of the verb to the 'agentive' feature of the nominal 'rqs'[13] (i.e.: a verb of the class 'cook' has as patient a noun of the class 'c_artifact', and a verb of the class 'ingest' has as patient a noun of the class 'comestible').

The assignment of the value 'deverbal' to the feature 'nominal-form' and the addition of the suffix form to the verbal stem for creating the new orthographical form are also necessary.

The rule for generating a patient noun[14] is:

```
deverbal-noun-pat-do1-trans
deverbal-noun-pat-trans
<1> = strict-trans-sign
<0> = lex-noun-sign
< 0: orth: orth1> = <1: orth>
< 0: orth: orth2> = "+do"
<0: cat: m-feats: agr> = <1: cat: active: cat: m-feats: agr >
<0: cat: m-feats: nominal-form> = deverbal.
<1: sem: arg2: arg2: arg2> = <0: sem: ind>
<1: sem> = < 0: rqs: agentive>
```

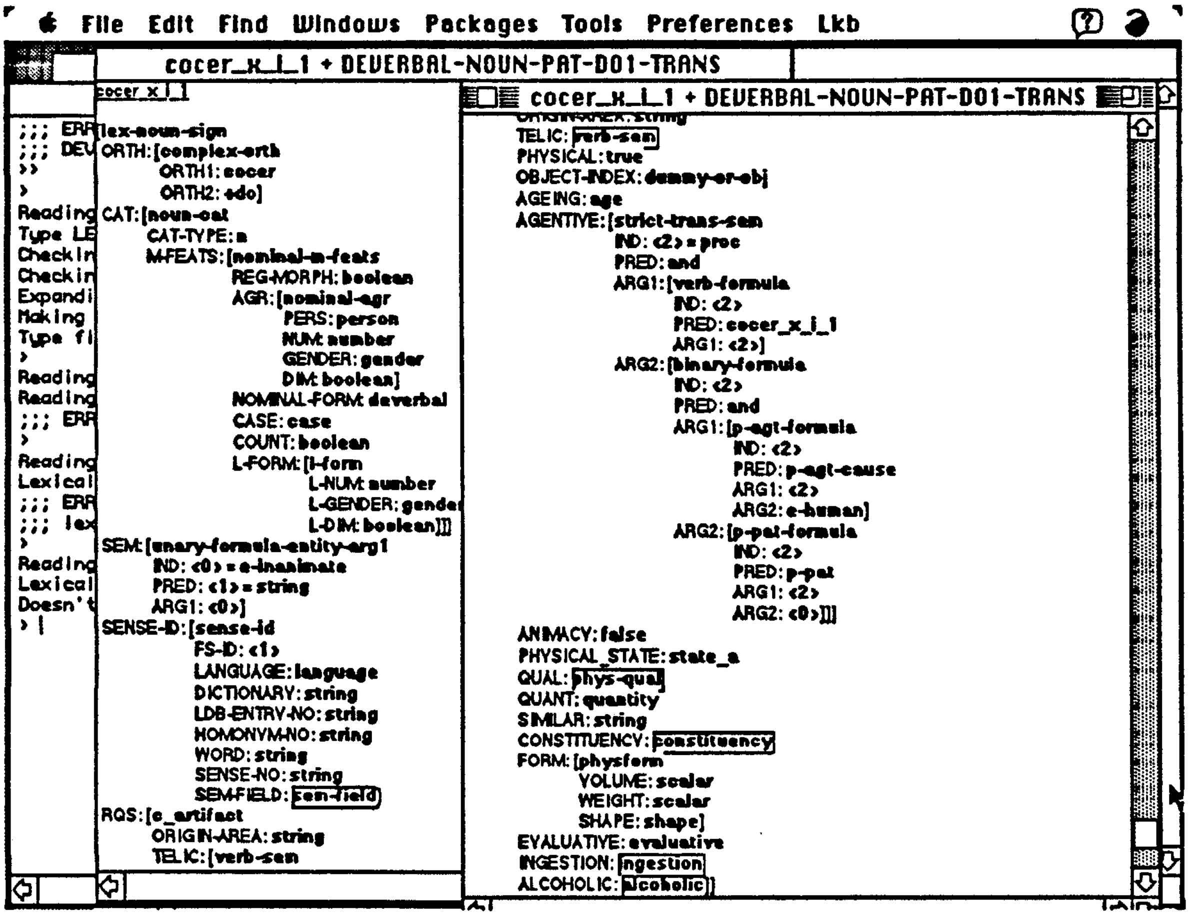
[13] The agentive feature specifies the creation process of the noun described [Pustejovsky 91].

[14] In this example we have made explicit the information inherited from its class. As for the action nouns we have developed a class of rules.

<1: rqs> = cook
<0: rqs> = c_artifact.

This rule produces patient nouns such as 'cocido' from the transitive verbal entry 'cocer'. The orthographical form is the concatenation of 'cocer+do'--> 'cocido'. We observe that 'cocer' is a verb of the class 'cook' and then 'cocido' is a noun of the 'c-artifact' class (comestible artifact).

In figure 8 we can see the result of the application of this rule:



```
cocer_x_i_1 + DEVERBAL-NOUN-PAT-D01-TRANS
cocer_x_i_1
```

```
;;; ERR
;;; DEV
>>
>
Reading
Type LE
Checkin
Checkin
Expandi
Making
Type fi
>
Reading
Reading
;;; ERR
>
Reading
Lexical
;;; ERR
;;; lex
>
Reading
Lexical
Doesn't
> |
```

```
[lex-noun-sign
ORTH:[complex-orth
        ORTH1: cocer
        ORTH2: +do]
CAT:[noun-cat
      CAT-TYPE: n
      M-FEATS:[nominal-m-feats
                REG-MORPH: boolean
                AGR:[nominal-agr
                      PERS: person
                      NUM: number
                      GENDER: gender
                      DIM: boolean]
                NOMINAL-FORM: deverbal
                CASE: case
                COUNT: boolean
                L-FORM:[l-form
                         L-NUM: number
                         L-GENDER: gende
                         L-DIM: boolean]]]
SEM:[unary-formula-entity-arg1
      IND: <0> = e-inanimate
      PRED: <1> = string
      ARG1: <0>]
SENSE-ID:[sense-id
           FS-ID: <1>
           LANGUAGE: language
           DICTIONARY: string
           LDB-ENTRY-NO: string
           HOMONYM-NO: string
           WORD: string
           SENSE-NO: string
           SEM-FIELD: sem-field)
RQS:[c_artifact
      ORIGIN-AREA: string
      TELIC:[verb-sem
```

```
cocer_x_i_1 + DEVERBAL-NOUN-PAT-D01-TRANS
```

```
      TELIC: verb-sem
      PHYSICAL: true
      OBJECT-INDEX: dummy-or-obj
      AGEING: age
      AGENTIVE:[strict-trans-sem
                 IND: <2> = proc
                 PRED: and
                 ARG1:[verb-formula
                        IND: <2>
                        PRED: cocer_x_i_1
                        ARG1: <2>]
                 ARG2:[binary-formula
                        IND: <2>
                        PRED: and
                        ARG1:[p-agt-formula
                               IND: <2>
                               PRED: p-agt-cause
                               ARG1: <2>
                               ARG2: e-human]
                        ARG2:[p-pat-formula
                               IND: <2>
                               PRED: p-pat
                               ARG1: <2>
                               ARG2: <0>]]]
      ANIMACY: false
      PHYSICAL_STATE: state_a
      QUAL: phys-qual
      QUANT: quantity
      SIMILAR: string
      CONSTITUENCY: constituency
      FORM:[physform
             VOLUME: scalar
             WEIGHT: scalar
             SHAPE: shape]
      EVALUATIVE: evaluative
      INGESTION: ingestion
      ALCOHOLIC: alcoholic]]
```

Figure 8: 'cocido' from 'cocer'.

So, the operations needed for generating a patient noun are the following:

- restrictions

1.- the input sign must be a transitive verb sign.
2.- the output sign must be a lexical noun sign.

- transference

  1.- the semantic patient of the verbal formula is transferred to noun semantic formula .

  2.- the semantic formula of the verb may be sometimes tranfer to the value of the feature 'agentive' of the noun 'rqs' feature, depending on the verbal class.

  3.- the morphological information associated to the active sign of the verb category is transferred to the morphological features of the noun.

- assignements

  1.- the value 'deverbal' is assigned as value of 'nominal-form' feature of the new sign.

  2.- the suffix value is assigned to the orthographical feature.

6. Conclusions

In Spanish, the derivation is a very productive process to generate new words. In this paper we focus on the derivation of nouns from verbs by means of lexical rules inside the Lexical Knowledge Base environment.

Lexical rules, a mechanism for the production of new entries from a basic set of existing ones, appears adequate for lexical knowledge representation because they capture lexical generalizations and, at the same time, they can deal with more specific phenomena. Furthermore, lexical rules are a mean to avoid redundant information in the Lexical Knowledge Base. Lexical rules also supports the theory of 'generative lexicon' proposed by J. Pustejovsky.

For these reasons the LKB is showing to be adequate for representing the changes that are produced in derivational processes because it allows to transform all the associated information to the entry, both the form and the content.

The current nominalization rules give account of the action, patient and agent nominalization from verbs. They capture all the changes produced in the nominalization process: orthographical, morphosintactic and semantic. These rules express also the information transferred, that is to say, that information which is the same in the two signs of the rule (input and output) .

One of the main problems we have deal with has been how to block the rule application for avoiding the generation of unexisting spanish nominalizations.In Spanish we don't have any information in the stem allowing the selection of the derivational suffixes. In this paper we propose a methodology for validating the new forms that will be implemented in the immediate future. It has to be considered as a contribution in the lexical acquisition framework. This strategy implies the extension of the the sources to the corpora. In our proposal, corpora are used for validating the nominal generated entries.

The nominalized lexicons would be of great interest for detecting in an exhaustive way the sufixes accepted by each Spanish verb and the verbal entries with the same derivational behavior.

References

[Acquilex 89]
Acquisition of Lexical Knowledge for Natural Language Processing Systems, Esprit Basic Research Project 3030. 1989 .

[Acquilex 92]
The Acquisition of Lexical Knowlwdge Esprit Basic Research Project 7315 ACQUILEX II, Technical Annexe.1992.

[Ageno 92]
Ageno, A., I. Castellón, M.A. Martí, F. Ribas, G. Rigau, H. Rodríguez, M. Taulé, M.F. Verdejo, 1992, "SEISD: an enviroment for extraction of semantic information from on-line dictionaries" en *Proceedings 3rd conference on Applied Natural LanguageProcessing*, Trento Italy. págs 253-256.

[Copestake 91]
Copestake and Briscoe (1991) "Lexical Operation in Unification based framework" *Proceedings of the ACL SIGLEX Workshop on lexical Semantics and Knowledge Representation*, Berkeley, California.

[Grimshaw 90]
Grimshaw, J., 1990 *Argument Structure,* Linguistic Inquiry Monographs, MIT Press, Massachusetts.

[Pustejovsky 91]
Pustejovsky, J., 1991, "The Generative Lexicon" en *Computational Linguistics*, vol 17, nº4. págs 409-441.

[Vox 87]
VOX , 1887, *Diccionario General Ilustrado de la Lengua Española* VOX, Ed.Biblograf S.A.;Barcelona .

Appendix 1

Links types, classes of lexical rules

a) action rule types

```
deverb-lex-rule (lexical-rule)
  < 1 : orth > = < 0 : orth : orth1>
  < 0 : sem : arg1 : pred> = < 1 : sem : arg1 : pred>
  < 0 : sem : pred > = < 1 : sem : pred >.
```

```
deverb-intrans-lex-rule (deverb-lex-rule)
  < 1 > = strict-intrans-sign
  < 0 > = deverb-intrans-sign
  < 1 : sem : arg2 > = < 0 : sem : arg2 : arg1 >.
```

```
deverb-trans-lex-rule (deverb-lex-rule)
  < 1 > = strict-trans-sign
  < 0 > = deverb-trans-sign
  < 1 : sem : arg2 : arg1 > = < 0 : sem : arg2 : arg1 : arg1 >
  < 1 : sem : arg2 : arg2 > = < 0 : sem : arg2 : arg2 : arg1 >.
```

b) agent rule types

```
deverbal-noun-agt-trans (lexical-rule)
<1> = strict-trans-sign
<0> = lex-noun-sign
<1: sem: arg2: arg1: arg2> = <0: sem: ind>
<1: sem> = < 0: rqs: telic>
<0: cat: m-feats: agr> = <1: cat: result: active: cat: m-feats: agr>
<0: cat: m-feats: nominal-form> = deverbal.
```

```
deverbal-noun-agt-intrans (lexical-rule)
<1> = strict-intrans-sign
<0> = lex-noun-sign
<1: sem> = < 0: rqs: telic>
<0: cat: m-feats: agr> = <1: cat: active: cat: m-feats: agr>
<0: cat: m-feats: nominal-form> = deverbal
<1: sem: arg2: arg2> = <0: sem: ind>.
```

c) patient rule types

```
deverbal-noun-pat-trans (lexical-rule)
<1> = strict-trans-sign
```

```
<0> = lex-noun-sign
<1: sem: arg2: arg2: arg2> = <0: sem: ind>
<1: sem> = < 0: rqs: agentive>
<0: cat: m-feats: agr> = <1: cat: active: cat: m-feats: agr>
<0: cat: m-feats: nominal-form> = deverbal.
```

Appendix 2

Lexical rules

a) nominalization: action noun rules (sample)

```
deverb-intrans-ida-rule
  deverb-intrans-lex-rule
   < 0 : orth : orth2 > = "+ida"
   < 0 : cat : m-feats : agr : num > = sg
   < 0 : cat : m-feats : agr : gender > = female
   < 0 : sem : arg2 : prep-cn-formula pred > = "de".

deverb-intrans-cion-rule
  deverb-intrans-lex-rule
   < 0 : orth : orth2 > = "+ción"
   < 0 : cat : m-feats : agr : num > = sg
   < 0 : cat : m-feats : agr : gender > = female
   < 0 : sem : arg2 : prep-cn-formula pred > = "de".

deverb-intrans-miento-rule
  deverb-intrans-lex-rule
   < 0 : orth : orth2 > = "+miento"
   < 0 : cat : m-feats : agr : num > = sg
   < 0 : cat : m-feats : agr : gender > = male
   < 0 : sem : arg2 : prep-cn-formula pred > = "de".


deverb-trans-ida-rule
deverb-trans-lex-rule
   < 0 : orth : orth2 > = "+ida"
   < 0 : cat : m-feats : agr : num > = sg
   < 0 : cat : m-feats : agr : gender > = female
   < 0 : sem : arg2 : arg1 : prep-cn-formula pred > = "por"
   < 0 : sem : arg2 : arg2 : prep-cn-formula pred > = "de".

deverb-trans-cion-rule
deverb-trans-lex-rule
   < 0 : orth : orth2 > = "+ción"
   < 0 : cat : m-feats : agr : num > = sg
   < 0 : cat : m-feats : agr : gender > = female
   < 0 : sem : arg2 : arg1 : prep-cn-formula pred > = "por"
   < 0 : sem : arg2 : arg2 : prep-cn-formula pred > = "de".

deverb-trans-ida-rule
deverb-trans-lex-rule
   < 0 : orth : orth2 > = "+miento"
   < 0 : cat : m-feats : agr : num > = sg
   < 0 : cat : m-feats : agr : gender > = female
  < 0 : sem : arg2 : arg1 : prep-cn-formula pred > = "por"
  < 0 : sem : arg2 : arg2 : prep-cn-formula pred > = "de".
```

a) nominalization: agent noun rules (sample)

```
deverbal-noun-agt-ante0-intrans
  deverbal-noun-agt-intrans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+ante".

deverbal-noun-agt-ante1-intrans
  deverbal-noun-agt-intrans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+ante"
  <1 : rqs> = cook
  <0: rqs> = human.

deverbal-noun-agt-ante2-intrans
  deverbal-noun-agt-intrans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+ante"
  <1 : rqs> = ingest
  <0: rqs> = creature.


deverbal-noun-agt-dor0-intrans
  deverbal-noun-agt-intrans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+dor".

deverbal-noun-agt-dor1-intrans
  deverbal-noun-agt-intrans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+dor"
  <1 : rqs> = cook
  <0: rqs> = human.

deverbal-noun-agt-dor2-intrans
  deverbal-noun-agt-intrans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+dor"
  <1 : rqs> = ingest
  <0: rqs> = creature.

deverbal-noun-agt-dor0-trans
  deverbal-noun-agt-trans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+dor".
```

```
deverbal-noun-agt-dor1-trans
  deverbal-noun-agt-trans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+dor"
  <1 : rqs> = cook
  <0: rqs> = human.
```

```
deverbal-noun-agt-dor2-trans
  deverbal-noun-agt-trans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+dor"
  <1 : rqs> = ingest
  <0: rqs> = creature.

deverbal-noun-agt-ante0-trans
  deverbal-noun-agt-trans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+ante".

deverbal-noun-agt-ante1-trans
  deverbal-noun-agt-trans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+ante"
  <1 : rqs> = cook
  <0: rqs> = human.

deverbal-noun-agt-ante2-trans
  deverbal-noun-agt-trans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+ante"
  <1 : rqs> = ingest
  <0: rqs> = creature.
```

a) nominalization: patient noun rules (sample)

```
deverbal-noun-pat-do1-trans
  deverbal-noun-pat-trans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+do"
  <1: rqs> = cook
  <0: rqs> = c_artifact.

deverbal-noun-pat-do2-trans
  deverbal-noun-pat-trans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+do"
  <1: rqs> = ingest
  <0: rqs> = comestible.

deverbal-noun-pat-do0-trans
  deverbal-noun-pat-trans
  < 0: orth: orth1> = <1: orth>
  < 0: orth: orth2> = "+do".
```

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

List of research reports (1993).

LSI-93-1-R “A methodology for semantically enriching interoperable databases”, Malú Castellanos.

LSI-93-2-R “Extraction of data dependencies”, Malú Castellanos and Fèlix Saltor.

LSI-93-3-R “The use of visibility coherence for radiosity computation”, X. Pueyo.

LSI-93-4-R “An integral geometry based method for fast form-factor computation”, Mateu Sbert.

LSI-93-5-R “Temporal coherence in progressive radiosity”, D. Tost and X. Pueyo.

LSI-93-6-R “Multilevel use of coherence for complex radiosity environments”, Josep Vilaplana and Xavier Pueyo.

LSI-93-7-R “A characterization of $PF^{NP||} = PF^{NP[\log]}$”, Antoni Lozano.

LSI-93-8-R “Computing functions with parallel queries to NP”, Birgit Jenner and Jacobo Torán.

LSI-93-9-R “Simple LPO-constraint solving methods”, Robert Nieuwenhuis.

LSI-93-10-R “Parallel approximation schemes for problems on planar graphs”, Josep Díaz, María J. Serna, and Jacobo Torán.

LSI-93-11-R “Parallel update and search in skip lists”, Joaquim Gabarró, Conrado Martínez, and Xavier Messeguer.

LSI-93-12-R “On the power of Equivalence queries”, Ricard Gavaldà.

LSI-93-13-R “On the learnability of output-DFA: a proof and an implementation”, Carlos Domingo and David Guijarro.

LSI-93-14-R “A heuristic search approach to reduction of connections for multiple-bus organization”, Patricia Ávila.

LSI-93-15-R “Toward a distributed network of intelligent substation alarm processors”, Patricia Ávila.

LSI-93-16-R “The Odissea approach to the design of information systems from deductive conceptual models”, Maria Ribera Sancho and Antoni Olivé.

LSI-93-17-R “Constructing face octrees from voxel-based volume representations”, Robert Juan i Ariño and Jaume Solé i Bosquet.

LSI-93-18-R "Discontinuity and pied-piping in categorial grammar", Glyn Morrill.

LSI-93-19-R "El frau i la delinqüència informàtica: un problema jurídic i ètic", Miquel Barceló (written in Catalan).

LSI-93-20-R "Non-homogeneous solid modeling with octrees. A geological application", Anna Puig, Isabel Navazo, and Pere Brunet.

LSI-93-21-R "Extending a single resolution system towards a distributed society", Karmelo Urzelai.

LSI-93-22-R "LINNEO$^+$: A classification methodology for ill-structured domains", Javier Béjar, Ulises Cortés, and Manel Poch.

LSI-93-23-R "Especificació d'una biblioteca de tipus", Xavier Franch (written in Catalan).

LSI-93-24-R "Proceedings of the Fourth Barcelona-Ulm Workshop on Probabilistic Complexity Classes and Nonuniform Computational Models" (Barcelona, September 13th-17th, 1993), José L. Balcázar and Antoni Lozano (editors).

LSI-93-25-R "Proceedings of the Fourth International Workshop on the Deductive Approach to Information Systems and Databases" (Lloret de Mar, 1993), Antoni Olivé (editor).

LSI-93-26-R "Modelo para el control de calidad en LESD basado en la medición del software", O. Slávkova (written in Spanish).

LSI-93-27-R "On the robustness of ALMOST-$\mathcal{R}$", Ronald V. Book and Elvira Mayordomo.

LSI-93-28-R "Lexicografía computacional: Adquisición automática de Conocimiento Léxico", Irene Castellón Masalles (written in Spanish).

LSI-93-29-R "Anàlisi de les definicions verbals del diccionari Vox", Mariona Taulé Delor (written in Catalan).

LSI-93-30-R "The structure of a logarithmic advice class", Montserrat Hermo.

LSI-93-31-R "Toward a realistic semantics of possible worlds for logics of belief", Gustavo Núñez, Matías Alvarado, and Ton Sales.

LSI-93-32-R "Conocimiento en mundos posibles mediante una relación de posibilidad constructiva", Matías Alvarado (written in Spanish).

LSI-93-33-R "Not-Yet classification algorithm", Josep Roure and Javier Béjar.

LSI-93-34-R "Concatenation versus addition in knapsack problems", Birgit Jenner.

LSI-93-35-R "Research Trends in Volume Modeling", Isabel Navazo, Anna Puig, and Dani Tost.

LSI-93-36-R "On Kobayashi's compressibility of infinite sequences", Montserrat Hermo.

LSI-93-37-R "A global algorithm for linear radiosity", Mateu Sbert and Xavier Pueyo.

LSI-93-38-R "An optimal parallel algorithm for learning DFA", José L. Balcázar, Josep Díaz, Ricard Gavaldà, and Osamu Watanabe.

LSI-93-39-R "DEPUR: a knowledge-based tool for wastewater treatment plants", Pau Serra, Miquel Sánchez, Javier Lafuente, Ulises Cortés, and Manel Poch.

LSI-93-40-R "How to know it all (or how to get rid of logical omniscience and perfect reasoners)", Ton Sales.

LSI-93-41-R "On adaptive dlogtime and polylogtime reductions", Carme Àlvarez and Birgit Jenner.

LSI-93-42-R "Polylogtime and logspace adaptive reductions", Carme Àlvarez.

LSI-93-43-R "Computational power of neural networks: a Kolmogorov complexity characterization", José L. Balcázar, Ricard Gavaldà, and Hava T. Siegelmann.

LSI-93-44-R "The $P_{\log^i}$ and AC^{i-1} operators on the polynomial time hierarchy", Jorge Castro and Carlos Seara.

LSI-93-45-R "Nominalizations in the LKB framework", Irene Castellón Masalles.

LSI-93-46-R "Subcategorization alternances generation via lexical rules", Mariona Taulé Delor, M. Antònia Martí Antonín, and Irene Castellón Masalles.

LSI-93-47-R "Dealing with lexical mismatches", Carmen Soler and Ma. Antònia Martí.

LSI-93-48-R "Translation equivalence via lexicon: a study on tlinks", Anna Samiotou, Irene Castellón, Francesc Ribas, and German Rigau.

LSI-93-49-R "A class-based approach to learn appropriate selectional restrictions from a parsed corpus", Francesc Ribas.

Internal reports can be ordered from:

Nuria Sánchez
Departament de Llenguatges i Sistemes Informàtics (U.P.C.)
Pau Gargallo 5
08028 Barcelona, Spain
secrelsi@lsi.upc.es