TULE
Documentation

THE VERBAL SUBCATEGORIZATION FRAMES

TULE Internal Report #1, December 2008

Author: Leonardo Lesmo

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF BOXES

LIST OF TABLES

1. INTRODUCTION

The final goal of the procedures described here is obtain an assignment of case labels to the verbal dependents. This is accomplished on the basis of a compiled representation of the verbal subcategorization frames (henceforth SF). This representation is the one used by the chunk parser to match the set of dependents of a given verb in the input sentence against the expectations embodied in the lexical knowledge for the involved verb.

For example, if verb V_i is transitive, and the chunk parser has established that its surface dependents are (* marks the position of the verb)

*(D1 * D2 D3)* [John kissed Mary in the park]

and if *D1* is unmarked and respects the constraints for acting as a subject, *D2* is unmarked, and *D3* is marked by a locative preposition (or suffix), then the procedures must be able to attach *D1* to V_i via an arc labelled as *VERB-SUBJ*, D2 to V_i via an arc labelled as *VERB-OBJ*, and D3 to V_i via an arc labelled as *PREP-RMOD-LOC*[1]. In order to get this result, the involved procedures must know:

- That V_i is transitive
- That a transitive verb licenses a subject and an object
- That a subject must respect some constraints (i.e. be in the nominative case, or precede the verb, or agree in number with the verb, ...)
- The same for the object

Note that the same holds for other complements (expected by the verb, as in case of movement verbs, or verbs licensing an indirect object), but not for the adjuncts (as the *PREP-RMOD-LOC* in the example above).

In some cases, however, the expected dependents can appear in different forms or play different syntactic roles. For instance, if the verb is passive, then the expected subject becomes the agent complement, while the object becomes the subject. In this case, we say that the verb has undergone a "transformation", i.e. it has been transformed from its base form to its passivized form. Consequently, the subcategorization frame must change, since the object is now absent. The example above becomes now:

(D1' $*_{[+passive]}$ D2' D3') [Mary was kissed by John in the park]

and the parser must know:

- That V_i is transitive
- That in the sentence under analysis it appears in the passive form
- That a passive transitive verb licenses a subject and an agent complement
- That the subject must respect the standard constraints (i.e. be in the nominative case, or precede the verb, or agree in number with the verb, ...)
- That the agent complement must be marked someway (e.g. via preposition, as "by" in English)
- That the subject actually corresponds to the "original" (or "deep") object
- That the agent complement corresponds to the "deep" subject

The procedures described in chapters 2 and 3 of this report must generate all the knowledge needed to carry out the match described above, except the subcategorization frame of the different verbs, which is given in the lexicon. They take as input:

- A set of base (deep) subcategorization classes (organized in a hierarchy)

[1] Actually, "in the park" is labelled only as *PREP-RMOD*, since it is an adjunct, and the analysis of adjuncts is currently only partial. Adjuncts are covered just marginally in this report.

- A set of transformations

And produce as output

- A set of possibly transformed surface subcategorization classes

It must be observed that not only the procedures described below, but also the two knowledge sources mentioned above (base classes and transformations) are defined as language-independent. In the description below, it is said how the obvious language-dependence of some construct is implemented in the model.

2. THE BASE CLASSES

The subcategorization classes are organized in a hierarchy, whose top-level nodes are shown in fig.1. This is a multiple inheritance hierarchy, although inheritance is not the standard one. The principles on which the hierarchy is based are presented in §2.3.

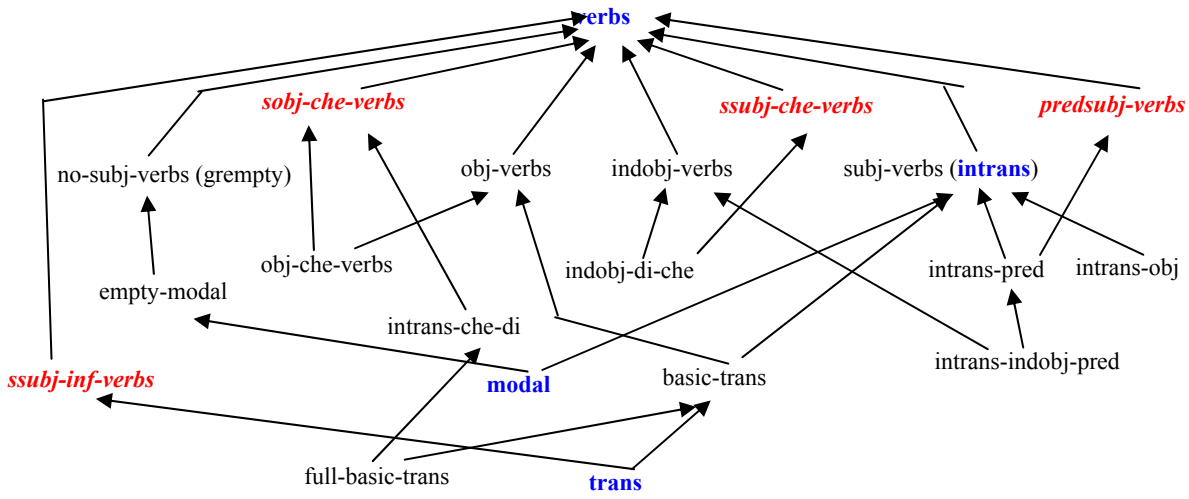

Figure 1 – Some top-level nodes of the hierarchy of base subcategorization classes. In red, abstract classes (see §2.2). In blue some standard labels for classes[2].

In box 1 below, I reported the definition of some classes. They will be useful for the description that follows.

2.1. LOGICAL BASIS OF THE HIERARCHY

The hierarchy actually merges two different hierarchies, having different meanings:

- the hierarchy of dependent cases
- the hierarchy of surface realizations

The first one has to be read as follows:

> "Each node includes a specification of all 'obligatory' cases. If a case is missing, this doesn't mean that it isn't obligatory, but just that is undefined with respect to obligatoriness"

So, *BASIC-TRANS* is below *SUBJ-VERBS*: *SUBJ-VERBS* defines all frames having, possibly among other cases, an obligatory subject. *BASIC-TRANS* is more restricted (i.e. specific, and so below it) because the direct-object is not free with respect to obligatoriness.

For example, the root *VERBS* (that does not include any specification) must be seen as a description as:

[2] Any class, identified by a name, can have synonyms. Actually, this is only used for *Intrans*, being a synonym of *subj-verbs*, and *no-subj-verbs* (synonym of *grempty* – grammatical relations empty).

VERBS:	*subject*:	present Y or N (default N)
	direct-object:	present Y or N (default N)
	indirect-object:	present Y or N (default N)
	to-loc:	present Y or N (default N)
	...	present Y or N (default N)

```
(obj-verbs          *** verbs with an object. Definition of object
  ()                                    *** no synonym (see footnote 1)
  (verbs)                               *** the parent in the hierarchy
          *** the possible surface realizations of a direct object *********
  ( (noun
     (art (not (is-a-date)))            *** the determiner is head of NP
     (adj (semtype £geogr) (number pl)) *** ex: "They meet Italians and French"
     (adj (type (indef demons deitt  interr  poss)))  *** these types of adjectives can be head of NP
     (pron (not (type refl-impers)) (case lobj))  *** pronouns (reflexives are handled apart)
     num))                              *** numerals can be head of NP
          *** the arc label in the dependency trees for direct objects *******
   (verb-obj))
(subj-verbs         *** verbs with a subject. Definition of subject
  (intrans)                             *** "intrans" is synonym of subj-verbs
  (verbs)                               *** the parent in the hierarchy
          *** the possible surface realizations of a direct object *********
  (((noun (agree))
    (art (agree) (not (is-a-date)))
    (pron (not (word-typ &art-relat) (type relat))  *** not all relative pronouns are licensed
          (not (type refl-impers)) (case lsubj) (agree))
    (adj (semtype £geogr) (agree))
    (adj (type (indef demons deitt interr poss)) (agree)
         (nc-head))                     *** the adjective must be the head of a noun-complex
    (num (agree))
    (prep (word-typ &subj-in) (down (cat pron) (type indef)) (agree))))  *** ex. In many have come
          *** the arc label in the dependency trees for direct objects *******
     (verb-subj))
(basic-trans
   ()
   (subj-verbs obj-verbs))
          ***** no extra cases, apart from the ones inherited from the parents  **********
```

Box 1 – The actual definition of the subcategorization classes *verb-obj*, *verb-subj*, and *basic-trans*. These definitions are "local", in the sense that inheritance has not been applied. See Box 2, for the definition of *basic-trans* after inheritance.

So that *SUBJ-VERBS* is simply:

SUBJ-VERBS is *VERBS* with subject restricted to "present Y"[3]

And *TRANS* is:

TRANS is both *SUBJ-VERBS* and *OBJ-VERBS* so that both the subject and the direct object are restricted to "present Y".

If I say that a verb is *TRANS*, then I say that it has a *subject* and a *direct-object* (present Y), but that it does not have a *to-loc* (default: present N).

[3] "present Y" corresponds to the statement that the case is obligatory. However, this applies just to deep structures, since some transformations (e.g. Pro-drop) may "cancel" obligatory cases. Moreover, since the parser aims to being robust, it chooses the best match between the description that is being described here and the actual dependents of a verb; so, it may happen that, in this "best" match, some obligatory cases are missing (see section 4).

The hierarchy of surface realizations applies to verbs licensing a given case. The interpretation is analogous to the previous one:

> "If it is specified that case X has realizations category$_1$ ... category$_n$, this means that these realizations are possible, but nothing is said about the other categories"

For instance, if I say that *BASIC-TRANS* admits an *Noun* subject, and that *TRANS* (below *BASIC-TRANS*) also admits a *Verb* subject, then I state that any *BASIC-TRANS* surface case-frame admits a *Noun*, but it is not said if it also admits a *Verb*, except for the verbs directly assigned to class *BASIC-TRANS* (to which the *Present N* default applies), whereas *TRANS* certainly admits both.

BASIC-TRANS

subject: *Noun:*	admissible Y	
Verb:	admissible Y or N (default N)	
Preposition:	admissible Y or N (default N)	
Conjunction:	admissible Y or N (default N)	
...	admissible Y or N (default N)	

So that *TRANS* is:

TRANS is *BASIC-TRANS* with *subject*: *Verb* restricted to "admissible Y".

Finally, it must be observed that each realization can have different degrees of specificity (e.g. "*Noun [agree]*", specifying that only nouns agreeing in number and person can act as subjects is more specific than *Noun*). This requires some care; in fact, it comes out that "*subject: Noun, Verb*" is neither more generic nor more specific than "*subject: Noun [agree]*". In fact, the first one is more restrictive wrt the admissibility of a *Verb* subject, while the second one is more restrictive wrt the characteristics of *Noun*.

2.2. ABSTRACT CLASSES

In order to exploit the advantages of a hierarchical organization, some "abstract" subcategorization classes are defined. They are assumed to have no verb belonging to them, and are used to define specific behaviors of some verbal classes. For instance, the class *sobj-that-verbs* is associated with all verbs that admit a sentential direct object governed by "that" ("che" in Italian). They include the definition of the form the "that-complement" may assume (i.e. that it is a conjunction which is a specific lexical item: "that" in English). In this way, this definition can be used for intransitives (e.g. "hope") and for Italian reflexives (e.g. "accorgersi" - remark). This is obtained by specifying that the abstract class is "parent" of the actual concrete class, that, in this way, inherits the definition. Of course, inheritance is defined according to the principles described in the previous paragraph, so that the dependent class is more restricted in the sense that the definition is now "admissible Y" in the dependent class, instead of being undefined about its admissibility (but with default "admissible N"). The list of abstract classes appears in Appendix 1.

2.3. APPLYING INHERITANCE TO THE BASE CLASSES

These initialization procedures build the internal representation of the hierarchy, before the application of the transformations. The resulting hierarchy includes only base (i.e. non transformed) frames. In order to obtain the full hierarchy, one must evaluate the function 'transform-hierarchy', which applies the transformations to the base classes.

The definition of a class includes the following pieces of information:

- *loc-cf-def*: the local definition of the class (without inheritance)
- *cf-def*: the complete definition of the class (with inheritance)
- *grclass-parents*: the list of parent classes in the base class hierarchy
- *grclass-deps*: the list of daughter classes in the base class hierarchy
- *transf-source*: the list of transformations applied to get this class. Obviously, it is empty (*nil*) for the base classes.

- *transf-sons*: the set of all classes obtained by applying one or more transformations. Initially empty.
- *appl-cond*: the applicability conditions of this class. The basic patterns are applicable to all verbs of the current class, so that *appl-cond* is initially set to *True*, but this may change after application of the transformations; for instance, if a passive verb imposes a passive pattern, then the application condition of the base (active) class can be applied just in case the verb is not passive.
- *order-cond*: These are conditions that can be posed on the order of verbal complements. (e.g. *(head-precedes verb-obj)*)

It must be noted that while the caseframe definitions (*cf-def*) of the base classes have two components (i.e. the case definition and the case label), here we have three of them. In fact, a "surface" case label list has been added. This is used for transformations, in order to keep trace of the correspondence between deep case and surface case. For a passivized direct object, we have *VERB-OBJ* as (deep) case label and *S-VERB-SUBJ* as surface case label. This enables the parser to label the "subject" arc of the dependency tree as *VERB-OBJ/VERB-SUBJ*.

In the example reported below, we see the full definition (after inheritance) of *BASIC-TRANS*. This should be compared with the local node definitions in the hierarchy, provided in box.1.

```
(basic-trans            *** basic transitives; no verbal subject or object
 (gr-class-parents   (subj-verbs obj-verbs)) *** the parents in the hierarchy (see Box 1)
 (loc-cf-def   (nil nil nil))                *** the local definition is empty (see Box 1)
 (cf-def (
  ((noun (agree))                  **** the inherited definition of subject ************
    (art (agree) (not (is-a-date)))
    (pron (not (word-typ &art-relat) (type relat))
          (not (type refl-impers)) (case lsubj) (agree))
    (adj (semtype £geogr) (agree))
    (adj (type (indef demons deitt interr poss)) (agree)
         (nc-head))
    (num (agree))
    (prep (word-typ &subj-in) (down (cat pron) (type indef)) (agree))))
   ((noun                          **** the inherited definition of direct object *************
     (art (not (is-a-date)))              *** the determiner is head of NP
     (adj (semtype £geogr) (number pl)) *** ex: "They meet Italians and French"
     (adj (type (indef demons deitt  interr  poss)))  *** these types of adjectives can be head of NP
     (pron (not (type refl-impers)) (case lobj))  *** pronouns (reflexives are handled apart)
     num))                                *** numerals can be head of NP
          *** the labels of the "deep" cases  *******
  (verb-subj verb-obj))
          *** the labels of the "surface" cases  *******
  (s-verb-subj s-verb-obj))
 (gr-class-deps   (trans trans-indobj trans-sinf trans-theme      *** the daughters in the hierarchy
                   trans-sinf full-basic-trans trans-2 trans-a trans-to-loc
                   trans-from-loc trans-indef-loc trans-loc-metaph trans-dest
                   trans-per trans-num-per trans-compar refl-trans trans-a-time
                                                          *** the next are verbal locutions
                   trans-in-pratica prendere-di-mira perdere-di-vista chiamare-in-causa)
 (transf-source nil)
 (transf-sons nil)
 (appl-cond nil)
 (order-cond t))
```

Box 2 – The definition of *basic-trans* after inheritance

2.4. MERGING CASE DEFINITIONS

The basic procedure for inheritance is the merge of two definitions. In fact:

$$cf\text{-}def\ (node_k) = loc\text{-}cf\text{-}def\ (node_k) \oplus cf\text{-}def\ (parent_{k,1}) \oplus \ldots \oplus cf\text{-}def\ (parent_{k,n})$$

The $\oplus$ operator is defined as follows:

$$\begin{aligned} def\ (node_1) \oplus def\ (node_2) = {} & (case\text{-}def_1\ (case_1) \cup case\text{-}def_2(case_1)) \circ \\ & (case\text{-}def_1\ (case_2) \cup case\text{-}def_2(case_2)) \circ \\ & \ldots \\ & (case\text{-}def_1\ (case_m) \cup case\text{-}def_2(case_m)) \end{aligned}$$

Where

- INH-CASES (n1, n2) = $\{case_1, case_2, \ldots case_m\}$ = CASES (n1) $\cup$ CASES(n2)
- $\circ$ is simply a concatenation of case definitions
- $\cup$ is the union of the two definitions

Although this basic idea is simple, it becomes noticeably complex because of two reasons:

1. The merging procedure must apply also to transformed patterns, so it must take into account the correspondence between deep and surface cases and must establish a case label both for surface and for deep labels. In particular, it may happen that a given surface case has an s-label ending with *NIL*. This means that the deep case has been cancelled as the effect of a transformation (e.g. after infinitivization, the subject has the labels *VERB-SUBJ* and *S-VERB-SUBJ/NIL*). In such a case, since the first step is to find corresponding labels, the final *NIL* must be removed.
2. The case labels are, in their turn, organized in a hierarchy, so, there can be situations where two labels are not the same, but one is more generic than the other. Although this situation can be dispensed with, it adds some power to the case matching system. It could happen that in a frame a case is simply labelled as *VERB-INDCOMPL-LOC*, while in another there is a VERB-*INDCOMPL-LOC+TO*. In such situations, I assume that the two labels refer to the same case, but that the more general label is safer (this is perhaps less precise, but the label hierarchy is used for this goal). So, the actually used label is the more general, and the surface realizations is inherited anyway. This applies also in case the two labels have a common ancestor which is more specific than *COMPLEMENT* or *ADJUNCT*: the label used in the result is the most specific common ancestor of the two input labels.

In Boxes 3 and 4, there are two examples of case merging. The first one is standard, and refers to an extension of the possible surface realization of the *VERB-SUBJ*, i.e. $m = 1$, and $case_1$ is *VERB-SUBJ*.

```
def (node1) = ((((verb (mood cong))))
                    (s-verb-subj) (verb-subj))
def (node2) = ((((prep (word-typ &base-subord-prep) (down (cat verb)))
                     (conj (word-typ &base-subord-conj))))
                   (s-verb-subj) (verb-subj))
def (node1) ⊕ def (node2) =
                  ((((verb (mood cong))
                 (prep (word-typ &base-subord-prep) (down (cat verb)))
                    (conj (word-typ &base-subord-conj))))
                   (s-verb-subj) (verb-subj))
```

Box 3 – Merging of two *VERB-SUBJ* definitions

In the first second example, a definition of the *VERB-OBJ* case is merged with a definition of the *VERB-OBJ+Q* case. Since the latter is a sub-label (i.e. it is more specific than) the former, the two definition are merged under the label *VERB-OBJ*.

```
def (node1) = ((((verb (mood (ind cong condiz)) (has-interr))
                  (conj (word-typ &interr-conj))))
                (s-verb-obj+q) (verb-obj+q))
def (node2) = (((noun
                  (art (not (is-a-date)))
                  (adj (semtype £geogr) (number pl))
                  (adj (type (indef demons deitt  interr  poss)))
                  (pron (not (type refl-impers)) (case lobj))
                  num))
                (s-verb-obj) (verb-obj))
def (node1) ⊕ def (node2)  =
              ((((verb (mood (ind cong condiz)) (has-interr))
                  (conj (word-typ &interr-conj))))
                  noun
                  (art (not (is-a-date)))
                  (adj (semtype £geogr) (number pl))
                  (adj (type (indef demons deitt  interr  poss)))
                  (pron (not (type refl-impers)) (case lobj))
                  num))
                (s-verb-obj) (verb-obj))
```

Box 4 – Use of label generalization in case merging

3. TRANSFORMATIONS

This section describes the final form of the verbal subcategorization classes, that is the one actually used by the parser. It also explains how this form is obtained, starting from the base subcategorization classes, to which inheritance has already been applied (i.e the result of the step described above), and from a set of *transformations*.

3.1. LOGICAL BASES OF TRASFORMATIONS

A transformation is an operation that takes as input the definition of a subcategorization class and produces the definition of a new subcategorization class. Transformations can be applied both to base classes and to transformed classes. Usually, a transformation does not apply to all base classes, but is somewhat constrained in its application. From a procedural point of view, transformations take as starting point the result of the step described in §2.4, i.e. the base classes obtained by the application of inheritance.

Currently, there are 17 transformations, listed in Box 5.

The general form of a transformation definition is as follows:

```
(transformation-name
 transformation-type
 list-of-base-classes-to-which-this-transformation-can-be-applied
 application-conditions
 order-preferences
 list-of-case-modifications)
```

passivization		"*She IS LOVED by John*"
insert-expletive	(English)	"*It IS interesting to go*"
infinitivization		"*I like to RUN in the fields*"
causal-infinitivization		"*I make him RUN*"
da-infinitivization		"*the things to DO*"
null-subj	(not English)	"*parto domani*" *([I] LEAVE tomorrow")* [pro-drop]
null-agt-compl		"*This IS often TOLD*"
null-indobj		"*John GAVE a coin*"
null-obj		"*Mary IS EATING*"
object-raising		"*Lo voglio vedere*" *([I] him want SEE)*
impersonalization	(not English)	"*Si dice che …*" *(Impers-Pronoun TELLS that …)*
reflexivization		"*Maria si lava*" *(Maria WASHES herself)*
inf-reflex		"*Aldo ama lavarsi*" *(Aldo loves to WASH himself)*
cliticization	(not English)	"*Lo vedo spesso*" *([I] see him often)*
clit-impers	(not English)	"*Lo si è visto nel parco*" *(Him impers-pronoun is seen in the park)*
fronting-doubled-obj		"*Luigi, molti lo hanno visto*" *(Luigi, many him have seen)*
dative	(English)	"*Bill gave Mary a flower*"

Box 5 – The list of transformations (with examples). Some of them are restricted to specific languages. Though they have been applied to Italian, English, Spanish and Catalan, only Italian and English have been developed to a reasonable level: this is why "not English", instead of "Italian" appears in the box.

The transformation type is a parameter that can take just two values: *replacing* or *extending*. The first of them means that the when the conditions of application are satisfied the original class is no longer valid. For instance, in *passivization*, there is the condition that the verb is passive. When *passivization* is applied to a class (e.g. *basic-trans*) the derived class *basic-trans+passivization* is obtained. Since the transformation is *replacing*, the original class (*basic-trans*) will include the condition that the verb is not passive. On the contrary, with *extending* transformation, the conditions of application of the source class are not modified.

With respect to the list of base classes, it must be noted that, in a sense, inheritance is applied also to transformations. In other words, if a transformation is applicable to class *C*, then it will be applied also to all subclasses of *C*.

The application conditions specify conditions to be verified on the actual input to license the application of a transformed frame. These include, for example, the already mentioned condition on the passive verb of passivized sentences.

The preferences on the order tells when a given dependent should come before or after the verb. Currently, they are not used by the parser (see §4.2).

The following predicates are used to express conditions of application and order preferences:

is-passive ()
is-inf-form ()
has-a-lobj-clitic ()
gov-by-modal-with-visitor ()
has-gramm-type (arg1 &vgov-prep-2) *(used for "da" in "da-infinitivization")*
head-precedes (arg1) *(arg1 is a case label)*
head-follows (arg1) *(arg1 is a case label)*

The predicates without arguments apply to the verb being analyzed, while the order predicates compare the position of the verb with that of the specified argument. *has-gramm-type* requires that the dependent identified by the case label *arg1*, is of type *&gov-prep-2*, which for Italian, includes the preposition "da"[4].

The actual transformations on the cases are carried out by the following operators:

cancel-case (surface-case-label)
add-case (case-description)
modif-case (change-description)

For cancellation, it is enough to specify which case must be cancelled. This is made via the surface label, so that, for instance, *infinitivization* cancels the *s-verb-subj* case. Note that this means that *infinitivization* must be applied after *passivization*, in order to cover both "I love to run" and "I love to be kissed". In the second sentence, the surface subject of "kiss" (i.e. its deep object) is cancelled.

Adding a case is a rather simple operation, since the description of the new case has the same format we have already seen for the class definitions. For instance, in *insert-expletive*, the new pronoun is licensed via a case description of the form:

((pron (word-typ &pron-neuter))) s-verb-subj verb-expletive))

Where *&pron-neuter* is a grammatical type (see footnote 4) having as its unique member the pronoun "it". Note that both the surface and the deep case labels are provided. This produces, when an expletive is matched, a case label of the form *verb-expletive/verb-subj*.

Much more complex is the modification of an existing case. This is because the modifications can be at different levels: a given category may be added or cancelled in the surface realization, or a category may, in turn, be subject to internal modifications.

3.2. THE GENERATION OF TRANSFORMED CLASSES

A transformation is an operation that takes as input the definition of a subcategorization class and produces another subcategorization class. This obtained by simply applying the definition of the transformation, given in the form outlined in the previous paragraph, i.e. by cancelling, adding, or modifying case definitions in the original frame.

Since this operation is rather clear, what remains to be said concerns the way transformations are combined. In fact, there are ordering constraints and mutual exclusion constraints that rule the sequence of applications. Both the order and the mutual exclusion constraints are expressed by a list of pairs, whose first element is a transformation and the second element is a list of transformations that are not compatible with the first one. The complete list is reported in Table 1. The constraints on order are obtained by the fact that the table is not symmetric. For instance, in the row of *passivization* it is specified that *null-subj* is incompatible with it, but the converse is not true (see the row about *null-subj*). This means that in case *nulls-subj* has already been applied, *passivization* is blocked, but after *passivization*, *null-subj* is permitted.

Note that the transformation mechanism can be used also to implement non-standard combinations. An instance is *inf-reflex*, which, in principle, should obtained by combining *infinitivization* and *reflexivization*. However, *reflexivization* inserts a clitic that must occur *before* the verb (Luigi *si* lava – Luigi *himself* washes), while, with infinitives, the enclitic occurs *after* the verb (Luigi ama lavar*si* – Luigi loves to wash *himself*). Of course, various solutions are possible, but the introduction of *inf-reflex* is the easiest one.

[4] "Grammatical types" are special word classes. I do not believe they have any special linguistic significance, but are used both by the POS tagger and by the parser to improve the performances. They could be replaced by simple world lists, but the tagger and the parser are multilingual; consequently, a "grammatical type" as *&agent-compl-prep* can be used in a single multilingual rule if it includes "da" for Italian and "by" for Engliah.

Another case worth noting is the one of *da-infinitivization* ("the things *to do*"). In this case, the transformation acts as a second-level transformation, in the sense that it is applied only after *infinitivization* has already been applied, by cancelling just the *verb-obj* (do *the things*), while the *verb-subj* was already cancelled by *infinitivization*. The definition of an autonomous *da-infinitivization* is not possibile in the current implementation, since it should be applied to *trans* (transitive verbs), but after *infinitivization*, *trans* got the condition that the verb must not be infinitive (*infinitivization* is *replacing*), so that *da-infinitivization* could not be applied.

Transformation	**Incompatible with**
passivization	*impersonalization reflexivization inf-reflex cliticization clit-impers da-infinitivization object-raising null-subj*
insert-expletive	*infinitivization*
impersonalization	*passivization reflexivization inf-reflex cliticization clit-impers null-subj*
reflexivization	*passivization impersonalization infinitivization inf-reflex cliticization fronting-doubled-obj clit-impers null-obj da-infinitivization*
inf-reflex	*passivization reflexivization impersonalization cliticization clit-impers*
infinitivization	*reflexivization null-subj*
da-infinititivization	*passivization reflexivization null-agt-compl null-obj object-raising cliticization*
null-subj	*infinitivization impersonalization*
null-obj	*cliticization da-infinitivization clit-impers object-raising fronting-doubled-obj passivization reflexivization*
null-agt-compl	*da-infinitivization object-raising*
fronting-doubled-obj	*reflexivization clit-impers object-raising null-obj*
clit-impers	*passivization impersonalization reflexivization inf-reflex fronting-doubled-obj null-subj object-raising*
object-raising	*passivization cliticization clit-impers fronting-doubled-obj null-agt-compl null-obj da-infinitivization*
cliticization	*passivization impersonalization reflexivization inf-reflex null-obj object-raising da-infinitivizazion*

Table 1 – List of incompatible transformations

In the current version of the system, the application of all transformations to the basic classes produces a set of 3083 transformed classes. These are represented as a hierarchy analogous to the one of base classes. This is made automatically and is completely transparent to the developer of the parser. As an example, the final definition of *basic-trans+passivization* is reported in Box 6.

4. THE MATCH BETWEEN DEFINITIONS AND DEPENDENTS IN A SENTENCE

While the previous sections described a kind of *pre-compilation work*, that is carried out once and for all in order to put at disposal of the parser a knowledge source easy to use in a direct way, this section describes how this is done, i.e. how the parser uses the pre-compiled result (the hierarchy of transformed classes) to assign the case labels to the dependents of a verb in an input sentence.

```
(basic-trans+passivization
 (loc-cf-def ((((noun (agree))
                 (art (agree) (not (is-a-date)))
                 (adj (agree) (semtype £geogr) (number pl))
                 (adj (agree) (type (indef demons deitt interr poss)))
                 (pron (agree) (not (type refl-impers)) (case lsubj))
                 num)
                ((prep (word-typ &agent-compl-prep) (down (cat noun)))
                 (prep (word-typ &agent-compl-prep) (down (cat art) (not (is-a-date))))
                 (prep (word-typ &agent-compl-prep)
                       (down (cat pron) (not (word-typ &art-relat) (type relat))
                                        (not (type refl-impers) (case obl)))
                 (prep (word-typ &agent-compl-prep) (down (cat adj) (semtype £geogr)))
                ((prep (word-typ &agent-compl-prep)
                       (down (cat adj) (type (indef demons deitt interr poss)))
                 (prep (word-typ &agent-compl-prep) (down (cat num)))))
            (s-verb-subj  s-verb-indcompl-agent)
            (verb-obj verbs-subj)))
 (cf-def   (  …..
             *** this definition is identical to loc-cf-def above ***
             *** this depends on the fact that "passivization" is defined on "basic-trans"
             *** so that it is applied on its cf-def definition, which becomes local
             …..  ))
(grclass-parents (verbs))
(grclass-deps (full-basic-trans+passivization  trans+passivization
            btrans-caus+passivization          btrans-compar+passivization
            btrans-for+passivization           btrans-indobj+passivization
            btrans-loc-dest+passivization      btrans-loc-from+passivization
            btrans-loc-indef+passivization     btrans-loc-metaph+passivization
            btrans-loc-to+passivization        btrans-multdiv+passivization
            btrans-pred-objinf+passivization   btrans-pred-sinf+passivization
            btrans-that-to-theme+passivization btrans-to-2+passivization
            btrans-to-time+passivization       refl-btrans+passivization
            chiamare-in-causa+passivization    perdere-di-vista+passivization
            prendere-di-mira+passivization     trans-in-pratica+passivization))
(transf-source (passivization   basic-trans))
 (transf-sons (basic-trans+passivization+null-agt-compl
               basic-trans+passivization+null-subj
               basic-trans+passivization+infinitivization)
 (appl-cond (and (is-passive) (not (is-inf-form))
 (order-cond T))
```

Box 6 – The definition of *basic-trans+passivization*

4.1. THE PROBLEM

The goal of the parser is to find the best match between the dependents of the verb (d_1, d_2, … d_n) in the input sentence and the expected dependents, as defined by the verbal class(es) of the actual verb and the corresponding transformed subcategorization classes. The first list is determined via heuristic rules that attach the various chunks identified in the sentence to one of the verbs present there. The second element of the match is a set of lists, obtained from the transformed classes associated with the verb. The procedure for retrieving those classes is:

1. *Find all base classes associated with the verb*
2. *Using the graph of transformed classes and its transf-sons link, get all possible transformed classes*
3. *Filter all classes which are not applicable, using the application conditions*

The result is a set of the following form:

$\{ (vtc_{1,1}, vtc_{1,2}, \ldots vtc_{1,n_1}), (vtc_{2,1}, vtc_{2,2}, \ldots vtc_{2,n_2}), \ldots, (vtc_{m,1}, vtc_{m,2}, \ldots vtc_{m,n_m}) \}$

Where each $vtc_{i,j}$ includes the case-name and the case definitions (both deep and surface). The task is to determine the index k such that

$(vtc_{k,1}, vtc_{k,2}, \ldots vtc_{k,n_k})$ matches $(d_1, d_2, \ldots d_n)$ in the best way

In order to get this result, one must first repeat on the result of 3 above the following step:

4. *Match each remaining class to the input*

Although the precompilation of class definitions (i.e the result of the process described in §3) tries to simplify this match, still the work to do is not easy. It must take into account that any dependent can be either a complement or an adjunct and that some complement can be missing.

4.2. THE APPROACH

Since the parser is intended to be a robust parser, since the $vtc_{i,j}$ may include adjuncts, and since the parser should also work on free-order (non-configurational) languages, the match may be partial and the preferences are enforced by heuristic rules.

The match of a single expected frame against the input is made via two nested loops. Given

$Scf_i = (vtc_{i,1}, vtc_{i,2}, \ldots vtc_{i,n_i})$

and

$D = (d_1, d_2, \ldots d_n)$

Then

- For each $vtc_{i,j}$ in Scf_i, look for a match with one or more of the d_i. Let's call the found matches $FM_i = \{d_{i,1}, d_{i,2}, \ldots d_{i,m_i}\}$, with $m_i \geq 0$. Both in case $m_i = 0$ (no match found) and $m_i \geq 0$ (some matches found), build the set $FM'_i = \{dummy, d_{i,1}, d_{i,2}, \ldots d_{i,m_i}\}$
- For each element d_k in FM'_i, add $< vtc_{i,j}, d_k >$ to the global found match and continue the search on $(vtc_{i,2}, \ldots vtc_{i,n_i})$ and $(d_1, d_2, \ldots, d_{k-1}, d_{k+1}, \ldots. d_n)$: In case d_k = *dummy*, D remains unchanged.

At the end of the loop, we have collected a tree, whose nodes are matching pairs (see Fig.2). Each path leading from the root to one of the leaves is a global match. Each leaf is associated with a set $(d_1, d_2, \ldots d_n)$ of input elements that have not been matched against the expectations. All of them are marked as *RMOD* (i.e. adjuncts). Each path may include elements of Scf_i that are associated with *dummy*. these elements are the complements not found in the input.

It must be observed that in case of a simple sentence as "John runs", we get two analyses, the first of which is the one where "John" is *VERB-SUBJ*, while in the second one, "John" is an adjunct and the *VERB-SUBJ* label is not assigned. Although this seems awkward, in pro-drop languages as Italian, we can have sentences as "Martedì corre" (Tuesday runs, i.e. "on Tuesday [s/he] runs"), where the second hypothesis is the correct one[5].

The final step is to compare all hypotheses and choose the best one. This is made via heuristic criteria (rules) that are currently expressed in a procedural way and are listed below.

[5] This does not mean that the preferences are actually able to choose the second reading, but here, the main goal was to take apart the two phases: hypotheses generation and best hypothesis selection.

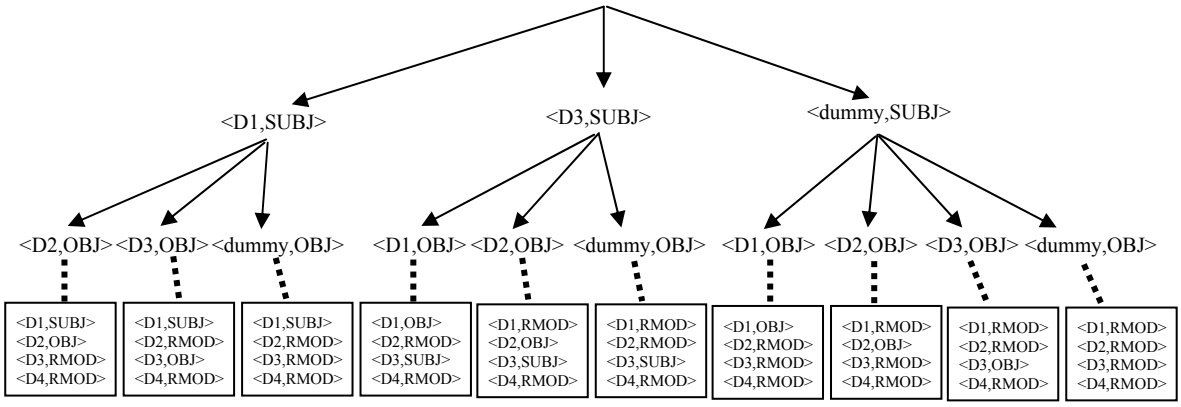

Figure 2 – Mapping between expected cases and dependents in the sentence. Expected cases are referred to by the case label (*SUBJ=VERB-SUBJ*, *OBJ=VERB-OBJ*), but they actually include the definition of the possible surface realization. Each box represents a hypothesis of assignment of cases to dependents. If, in a box, an expected case is missing, it is assumed to be missing in the sentence.

1. A 'verbal-locution' label is assigned[6]
2. The *EMPTYCOMPL* label is assigned (pseudo-reflexives)
3. The *VERB-SUBJ* is not assigned to a temporal expression
4. The *VERB+INDCOMPL-MODAL* label is assigned (modals)
5. ENGLISH: The subject does not follow the verb, unless the sentence is interrogative or *PRON-RMOD-LOC+METAPH* is assigned or VERB-EXPLETIVE is assigned
6. The *PRON-RMOD-LOC+METAPH* label is assigned
7. The *VERB-SUBJ+IMPERS* label is not assigned to an enclitic
8. The *VERB-SUBJ+IMPERS* label is assigned, and the verb has preference for impersonals wrt reflexives
9. There are less adjuncts
10. The verb is imperative, and the *VERB-SUBJ* is not assigned
11. There are less complements not found in the sentence
12. A *VERB-PREDCOMPL* case is assigned to an adj
13. Less transformations have been applied
14. There is no competition between *VERB-SUBJ* and *VERB-OBJ*, but *VERB-OBJ* is assigned
15. There is competition between *VERB-SUBJ* and *VERB-OBJ*, but and *VERB-SUBJ* is assigned to a case before the verb or *VERB-OBJ* is assigned to a case after the verb (currently, the covered languages are SOV – at least preferentially – languages. This preference could easily be made language-dependent)
16. There is competition between *VERB-INDOBJ* and *VERB-OBJ*, and the verb is a *£predic3* verb (dire, spiegare, ...)
17. In both matches there are a *VERB-OBJ* and a *VERB-PREDCOMPL-OBJ*: choose the one where the *VERB-OBJ* comes before the other

Note that criteria 9 and 11 are different, since the comparison is often made between two different subcategorization frames. So, it may happen that scf_1 includes just 2 complements, while scf_2 includes three complements (perhaps scf_1 was obtained by scf_2 via *null-subj*), and if two unlinked

[6] Verbal locution labels are identified by having as their last component "**LOCUTION*"; so, we can have *VERB-OBJ*LOCUTION*, *VERB-INDOBJ*LOCUTION*, *VERB-INDCOMPL-LOC*LOCUTION*, etc… See Appendix C for a short description of the way verbal locutions are dealt with.

words satisfying the constraints appear in the sentence, then both matches are ok (without adjuncts), but the first is better, since it is a 'complete' match. However, the main observation concerns the fact that it is not required that all complements (or arguments, or obligatory cases) are assigned. The fact that complements are actually present is just a preference (criterium 11). It may seem strange that such a basic principle (obligatoriness of complements) has such a low preference, but no extended test has been made in order to sort the criteria in the best way, so that the choices have been made on the basis of the parser results, and of the intuitions of the developers.

APPENDIX A. The verbal classes

In the following table, we list all the base subcategorization classes currently in use, apart from the verbal locutions (see Appendix C). For some of them, comments are provided. In blue, abstract classes (see Appendix B) used in the definition. For instance, *intrans-from-theme*, inherits the defintions of *intrans* and *theme-from-verbs*. Where no abstract class appears in a "more specific" definition (as in *intrans-compar*, which a subclass, i.e. it is more specific than *intrans*), the extra case is defined locally, instead of being inherited.

CLASS NAME	PARENTS	COMMENTS
verbs		
no-subj-verbs (grempty)	verbs	
empty-modal	no-subj-verbs	VERB+MODAL-INDCOMPL (verb inf)
doubt-be-verbs	verbs	"it IS not clear if …"
obj-verbs	verbs	VERB-OBJ (noun, art, adj, num, pron)
obj-che-verbs	obj-verbs, sobj-that-verbs	The Italian form "ecco"
indobj-that-to	indobj-verbs ssubj-that-to-verbs	+ VERB-SUBJ (verb subjunct)
subj-verbs (intrans)	verbs	VERB-SUBJ (noun, art, adj, num, .. [agree])
colour-pred	subj-verbs ssubj-that-inf-verbs	
modal	empty-modal subj-verbs ssubj-inf-verbs	
intrans-compar	intrans	VERB-INDCOMPL-THEME (&as)
intrans-for	intrans	
intrans-from-theme	intrans theme-from-verbs	
intrans-goal	intrans	
intrans-in-on-theme	intrans	
intrans-indobj	intrans indobj-verbs	
intrans-indobj-pred	intrans-indobj intrans-pred	
intrans-indobj-that-to	intrans-indobj intrans-that-to	
intrans-indobj-yn	intrans-indobj sobj-yn-verbs	
intrans-indobj-q	intrans-indobj sobj-q-verbs	
intrans-indobj-ssubj	intrans-indobi intrans-ssubj	
intrans-loc	intrans loc-in-verbs	
intrans-loc-from	intrans loc-from-verbs	
intrans-loc-indef	intrans	
intrans-loc-metaph	intrans	
intrans-loc-to	intrans loc-to-verbs	
intrans-num	intrans	
intrans-obj	intrans	The "Object" is not a syntactic dir. object
intrans-on	intrans	
intrans-on-theme	intrans	
intrans-partitive	intrans	
intrans-poss	intrans	
intrans-pred	intrans predsubj-verbs ssubj-that-inf-verbs	
intrans-pred-ssubj	predsubj-verbs ssubj-that-to-inf-verbs	Used for expletives
intrans-pred-no-ssubj	intrans predsubj-verbs	
intrans-pred-che	intrans ssubj-inf-verbs	
intrans-pred-prep	intrans pred-prep-subj-verbs ssubj-that-inf-verbs	
intrans-progress	intrans	Predicative complement headed by prep
intrans-q	intrans sobj-q-verbs	
intrans-result	intrans result-verbs	
intrans-so	intrans	
intrans-sobjinf	intrans sobj-inf-verbs	

intrans-ssubj	intrans ssubj-that-to-inf-verbs	
intrans-ssubj-sobjinf	intrans-ssubj sobj-inf-verbs	
intrans-that	intrans sobj-that-verbs	
intrans-that-to	intrans sobj-that-to-verbs	
intrans-theme	intrans theme-in-verbs	
intrans-time	intrans	
intrans-time-2	intrans	
intrans-time-dur	intrans	
intrans-to	intrans stheme-to-verbs	
intrans-to-2	intrans stheme-to2-verbs	
intrans-topic	intrans topic-verbs	
intrans-with	intrans	
intrans-yn	intrans sobj-yn-verbs	
trans-neg-che	subj-verbs	
trans-dir-disc	subj-verbs	
trans-dir-disc-indobj	trans-dir-disc indobj-verbs	
basic-trans	subj-verbs obj-verbs	
full-basic-trans	basic-trans sobj-that-to-verbs	
trans-full-indobj	full-basic-trans indobj-verbs	
trans-full-topic	full-basic-trans	???? theme
trans-full-topic-2	full-basic-trans topic-verbs	
trans	basic-trans ssubj-inf-verbs	
trans-from-theme	trans stheme-from-verbs	
trans-in-theme	trans theme-in-verbs	
trans-pred	trans predobj-verbs ssubj-that-verbs	
trans-result	trans result-verbs	
trans-separ	trans theme-from-verbs	
trans-to-with-theme	trans	
btrans-caus	basic-trans	
btrans-compar	basic-trans	
btrans-indobj	basic-trans indobj-verbs	
btrans-for	basic-trans	
btrans-loc-dest	basic-trans	
btrans-loc-from	basic-trans loc-from-verbs	
btrans-loc-indef	basic-trans	
btrans-loc-metaph	basic-trans	
btrans-loc-to	basic-trans loc-to-verbs	
btrans-multdiv	basic-trans	
btrans-pred-objinf	basic-trans sobj-inf-verbs predobj-verbs	
btrans-pred-sinf	basic-trans predobj-verbs	
btrans-that-to-theme	basic-trans stheme-that-to-verbs	
btrans-to-2	basic-trans stheme-to2-verbs theme-in-verbs	
btrans-to-time	basic-trans	
refl	subj-verbs	EMPTYCOMPL (for the reflexive pron)
refl-btrans	refl basic-trans	
refl-from-theme	refl theme-from-verbs	
refl-in-obj	refl	
refl-in-theme	refl theme-in-verbs	
refl-loc-in	refl loc-in-verbs	
refl-loc-to	refl loc-to-verbs	
refl-manner	refl	
refl-name	refl	
refl-of-theme	refl	
refl-of-topic	refl topic-verbs	
refl-on-topic	refl	
refl-pred	refl predsubj-verbs	
refl-result	refl result-verbs	
refl-that-to	refl sobj-that-to-verbs	
refl-to-obj	refl sobj-to-verbs	
refl-to2	refl stheme-to2-verbs	
refl-with	refl	
refl-to-with	refl-with	

APPENDIX B: The abstract subcategorization classes

Abstract class	Parent class	Comments
indobj-verbs	verbs	verbs with an indirect object
loc-from-verbs	verbs	verbs with a 'from' location (movement)
loc-in-verbs	verbs	verbs with a 'in' location (static) (governed by "in, before, behind, beyond" - English - or "in, a, davanti, dietro, oltre" - Italian)
loc-to-verbs	verbs	verbs with a 'to' location (movement)
predsubj-verbs	verbs	verbs with a (unmarked) predication applied to subject
pred-prep-subj-verbs	predsubj-verbs	the predicator can also be marked by a preposition
pred-obj-verbs	verbs	verbs with a predication applied to direct object
result-verbs	verbs	verbs with a result (e.g. "change")
sent-obj-verbs	verbs	verbs with a sentential direct object
sobj-that-verbs	sent-obj-verbs	verbs with sentential object governed by 'che' (I) or 'that' (E)
sobj-di-verbs	sent-obj-verbs	verbs with sentential object governed by 'di'
sobj-inf-verbs	sent-obj-verbs	verbs with sentential object rooted in an infinite verb
sobj-yn-verbs	sent-obj-verbs	verbs with sentential object implementing a y/n question
sobj-q-verbs	sent-obj-verbs	verbs with sentential object implementing a standard question
sent-subj-verbs	verbs	verbs with a sentential subject
ssubj-che-verbs	sent-subj-verbs	verbs with sentential subject governed by 'che'
ssubj-di-verbs	sent-subj-verbs	verbs with sentential subject governed by 'di'
ssubj-inf-verbs	sent-subj-verbs	verbs with sentential subject rooted in an infinite verb
theme-verbs	verbs	verbs with a theme (sentential or not)
stheme-a-verbs	theme-verbs	verbs with a "theme" sentential object governed by &theme-prep-1 ('a' - I -, 'in' - E -)
stheme-to-verbs	theme-verbs	verbs with a "theme" sentential object governed by &base-subord-prep ('di' - I -, 'to' - E -)
stheme-from-verbs	theme-verbs	verbs with a "theme" sentential object governed by &theme-prep-5 ('da' - I -, 'from' - E -)
stheme-in-verbs	theme-verbs	verbs with a theme governed by 'in' (English) or 'a' (Italian)
stheme-da-verbs	theme-verbs	verbs with a theme governed by 'in' (English) or 'a' (Italian)
topic-verbs	verbs	verbs with a topic

APPENDIX C. Verbal locutions

The standard way for dealing with verbal caseframes, outlined in this document, is also used to cope with verbal locution, as "prendere di mira" or "kick the bucket". A verbal locution is represented as a separate base subcategorization class, and undergoes all phases of preprocessing described above. So, there is not much to say about the way they are handled, except that one of the "cases" appearing in the subcategorization frame is fixed, in the sense that its "surface description" is frozen as a specific word or subtree,

For instance, in "kick the bucket" nothing special happens until the dependents of the verb under analysis (to kick) are matched against the known frames. The process is exactly the same described in section 4.3,; consequently, in "Most of our friends kicked the bucket", the parser finds a match both with the standard transitive form of "to kick" and with the verbal locution. However, the latter match is better, since it satisfy the preference criterion 1, i.e. the label assigned to "the bucket" is a "verbal locution" label (in particular *VERB-OBJ*LOCUTION*)..

APPENDIX D. The Files

In this section, we list the actual files mentioned in this report, where the data and the procedures reside. In these files, one can find all details not covered by the present report, as the actual definition of all verbal classes and transformations.

The rectangles in Fig.3 are directories; the string in violet are loading files, the ones in blue are data files, the ones in red are procedures. All relevant data are in the directory *SYSTEM*. The *DATI*

directory does not contain the subcategorization data, but parsing data (input text and resulting trees). *DESCR* contains documentation.

The relevant files are:

- tule: loads the parser (not used in case the GUI is the interaction mean)
- tule-debug: loads the parser in non-compiled form
- lohier: loads the procedures for generating the transformed subcategorization data
- subc-classes.dat: the definition of the base subcategorization classes
- transf-def.dat: the definition of the transformations
- lab-hierarchy.dat: the hierarchy of arc-labels
- newhier.out: the final result of the process of generating transformed subcat classes
- subc-hier.lisp: the procedures for applying inheritance to the base classes
- transf-hier.lisp: the procedures for applying transformations to the base classes
- chunk-parser.lisp: the parser, which does what described in ch.4 of this report
- hier-funct.lisp: low-level functions for accessing the hierarchy of subcat classes
- verbclass-ita.dat: the assignment of verbs to verbal classes for Italian
- verbal-locutions-ita.dat: Italian verbal locutions
- grammtypes-ita.dat: grammatical types for Italian (see footnote 4)
- verbclass-eng.dat: the assignment of verbs to verbal classes for English
- verbal-locutions-eng.dat: English verbal locutions
- grammtypes-eng.dat: grammatical types for English (see footnote 4)

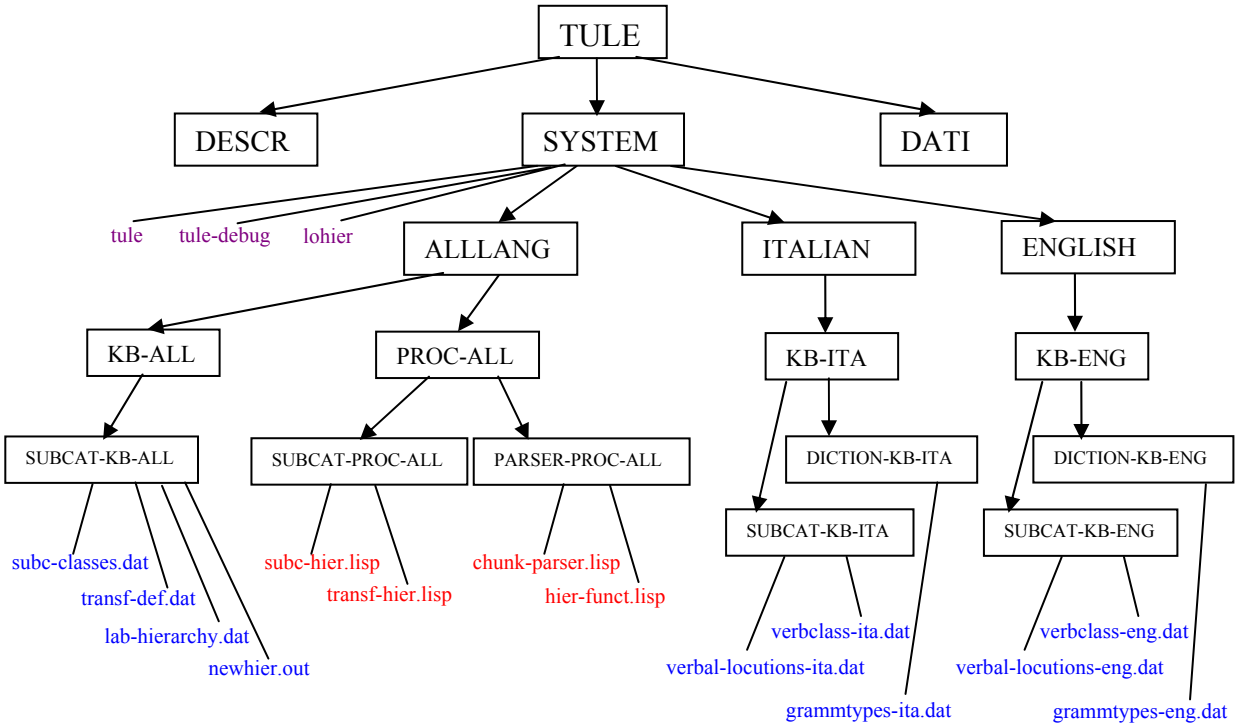

Figure 3 – Directories and position of the relevant files