Rodolfo Delmonte

Translating with ROSIE

1992
CLI - 92.1.4

University of Venice
Working Papers in Linguistics

Centro Linguistico Interfacoltà
Università degli studi di Venezia
S.Croce 2161 - 30135 Venezia
TEL.39-41-5241642  FAX 39-41-718259
E-MAIL CLI@IVEUNCC
TRANSLATING WITH ROSIE

Rodolfo Delmonte

University of Venice - Department of Linguistics - Ca' Garzoni Moro - San Marco
3417 - 30124 VENICE (It)
E-mail: delmonte@lveuncc.bitnet
Phone: 39-41-5298459/5204477/5287683
FAX: 39-41-5287683

1. Abstract
We present a system for machine translation which is based on Rosie a system for reference retrieval in texts (see Delmonte & Bianchi 1991a/1991b) which in turn works on LFG. In this sense, the system is neither an interlingua nor a transfer system because it must allow for modularity which is an inherent feature of LFG-based systems (see Kaplan et al., 1989). Rosie is a very powerful analyzer which computes anaphora resolution at discourse level: it is crucial for Romance languages which possess typologically the option of pro-drop. One of the main reasons for using a system with anaphora resolution at text level, is that pro-drop causes gender to be available only by means of anaphoric binding. More examples are provided in the text below.

2. Introduction
In our opinion, the implementation of any natural language processor and/or generator, should follow consistently a linguistic theory. Linguistic theory makes available to the computer scientist the tools to implement elegant and and simple systems, which in turn may also result at explaining why a given psychological process "performs" that way. The foundation of a theory of performance is stated in Bresnan's (1982, xxii-xxiii) introduction, which we endorse entirely. When building up a translation system, choice of the theoretical framework in which the system will be embedded is of paramount importance. As well be shown in the paper, a completely different set of procedures may have to be set up and these are strictly dictated by choices made at a theoretical level. Thus, in principle, it will be possible to reach an explanatory theory of translation only in as far as the underlying linguistic theory is capable to support it. In compliance with that, we take performance theory to be the founding criterion of translation systems; implementations that comply with this criterion will not only
be elegant and simple but also show a high level of integration in their internal organization. As in Kaplan et al. (1989) we use LFG as a theoretical backbone; however, differently from their approach, our system is not a hybrid approach in that we only use LFG representations enriched by a number of submodules for the computation of temporal-aspectual relations, as well as a module for logical form, and a system for the resolution of anaphora in discourse which all bear on a final f-structure output. This output is then used as the source and a number of transfer rules are used to map it onto a target f-structure which is feeded on to the target grammar. No use is made of a semantic representation, nor can we define our system as been both transfer-based and interlingua-based.

3. Modularity within the Theory
We follow strictly LFG theory in that we make use of a highly modular system made up of these levels of linguistic representation:
- c-structure
- f-structure
Onto f-structure the following modules are made to interact:
- a binding module, for pronominal binding within the sentence (see Delmonte & Bianchi, 1991a);
- a module for temporal aspectual computation based on tense and aspect feature values, which in turn is cast on Reichenbach's tripartite system of relation and Allen's system of temporal inferences;
- a module for scope assignment to quantifiers based on Shieber and Hobb's algorithm with a number of additions and modifications (see Delmonte, 1990);
- a module for the resolution of anaphora in discourse (see Delmonte & Bianchi, 1991b);
Finally, there is a module for generation,
- the generation algorithm, GraFo.

4. The pronominal argument
In particular, Italian possesses a set of free pronouns made up of three types of lexical elements: clitics, i.e. unstressed pronouns; independent pronouns, i.e. stressed personal pronouns which may also be used for contrast and emphasis;
and empty subject and object pronouns. Clitics and empty pronouns may be used as variables at c-structure levels when required. In addition, Italian also possesses a set of reflexive pronouns which are made up of two different types of lexical elements: short anaphors, i.e. Nuclear reflexives not Subjective, however; and long anaphors, i.e. not Nuclear and Subjective.

As to the set of possessive pronouns, Italian has the same variety as English but with internal differences: as opposed to what happens in English, free pronouns in Italian cannot be bound to a quantifier, but anaphoric pronouns can. Also, arbitrary or generic reading is only allowed with anaphoric pronouns which however could be bound to a local antecedent, in case there is one. This must also be computed differently from what happens in English, where the pronoun "one's" is unambiguously arbitrary in reading.

Possessive pronouns must be assigned an antecedent in Italian in order to be assigned gender adequately: whereas in Italian Gender is the result of syntactic agreement with the local Noun head, in English it is the result of a binding rule which assign Gender on the basis of the possessor's. The Possessor might as well be located in a separate sentence, hence the need to proceed to anaphora resolution at discourse level. Besides, quantifier bound possessive anaphors must be translated as normal possessive pronouns in English, and this will ensue from the computation of the binding module at sentence level.

As to empty pronominals, i.e. little pro's, Gender features must be recovered from the antecedent given that it may or it may not be made available according to the Verb structure. In case a compound verb has been used in Italian, past participle may assign gender to the subject pro whenever auxiliary "be" has been used, be it with a reflexive use of a transitive verb, be it with a passive or middle use. Also auxiliary "be" is always used with unaccusatives and a subset of intransitive unergatives like "correre"/run which may assume either "be" or "have" as auxiliary according to its aspecual value. Some examples follow,

These examples are all taken from Cardin(1982):
1. Backward pronouns bound by a quantifiers or generic, however in Italian only little pro is allowed,
a. each victim expects that he will survive (ciascuna vittima si aspetta che pro sopravviverà)
b. each victim expects that they will survive ≠
   the variable reading is available: ("x ∈ the victims) (x expects that x will survive)
   * the set reading is not available in Italian: ("x ∈ the victims) (x expects that the victims will survive)
2. Singular pronouns bound by singular quantifiers require a possessive anaphor in Italian,
c. there are six legally operated and licensed poker cardrooms... As its(propria/*sua) major source of income, each club collects a playing fee from the players every half hour...
d. Al Bowling and Tom Rentschler... In his(propria/*sua) own way, however, each man is petitioning for the same kind of Administration.
e. Scrimshawing took time. And, once his(propria/*sua) ship had reached the whaling grounds, time was something every whaleman had a great plenty of.
3. Plural pronouns with the variable reading are allowed also in Italian,
a. When their(loro/proprio) government tenure ends, many officials simply move to new offices...
b. Did you know that when their(loro/proprie) wives leaves them, two men in five go bananas?
4. Bound by a quantifier: variable reading also requires the anaphoric pronoun in Italian,
   The students noted that before PRO entering his(propria/*sua) plea, each defendant was advised by Judge Doyle of his(propri/*suoi) rights.
5. Little pro can be bound by a definite NP which has generic reading,
   - By the time he(pro/*egli) reaches the age of 70, the average American consumes 13 tons of beef...
As can be seen, possessive pronouns may be bound by a quantifier in Italian only when plural is used; otherwise, "proprio" must be used in place of "suo". As to subject pronouns, they can never be bound to quantifiers if lexical: only little pro's can.
5. Thematic Divergencies in LFG
We shall now proceed by comparing our system to the one presented by Dorr (1990). The UNITTRAN system solves the thematic divergencies (hence TDs) problem by mapping an underlying lexical-conceptual structure to a syntactic structure (and vice-versa), on the basis of a set of general linking routines and their associated mechanisms (see Dorr, 128). According to her analysis, there are two types of TDs that show up in the translation of a source language to a target language: the first type consists of a reordering of arguments for a given predicate, and the second type consists of a reordering of predicates with respect to their arguments or modifiers.

In her system predicate-argument structures are deep structures constituted by the base phrase marker representing a set of maximal projections produced by the X-bar system. The nuclear information used by the system is constituted by a different set of uninstantiated lexical conceptual structures, called RLCS, associated to each verbal predicate, for each language which include language specific information. In turn this RLCS is instantiated in order to represent the input string from the source language and this is called CLSC, which is used as an interlingua or language independent form. Once the lexical mapping has been successfully carried out, the CLCS must be mapped onto the syntactic structure of the target grammar and then the final sentence form will be generated. The mapping is achieved by a set of General Linking Routines which must cover both the mapping from source syntactic structure to CLCS positions, and the mapping from the CLCS to target syntactic structure. As appears, in Dorr's system three steps are crucial in order to achieve the final result: from RLCS to CLCS, from CLCS to target syntactic representation, from source syntactic representation to target syntactic representation. In particular, in her "like-gustar" example (ibid, 132), she has to allow the syntactic realization of the logical subject "w" and the syntactic realization of a non-subject argument (say z_k) to switch places between the source and the target language. This is achieved in the phrase marker by an argument reversal.

In our system only two steps are required: from source f-structure onto the General Mapping Routine (hence GMR),
and from the GMR onto the target grammar to generate the output translated sentence form.

The problem of thematic divergencies or translational equivalents in LFG has a different status from what happens in GB framework as discussed by Dorr. The range of variation may affect the order in which s-roles and GFs are associated, or the type of GF associated to a given s-role. Differences in GFs are automatically encoded into differences of constituency at c-structure level. Consider the case of the English counterpart of the verb "piacere", i.e. "like", c) pred_v(like,psych,state,emotive,np/subj/experiencer, np/object/theme_emot).

The two lexical forms in a) and b) differ only in the association of s-roles to GFs: in particular, in the Italian lexical form the experiencer is associated to an OBL, whereas in the English one it is associated to an OBJ. This fact is simply accounted for by the GMR internal rationale which gives priority to s-roles in the transfer process and disregards GFs.

Consider now a constituency divergency as represented by the case of "be hungry" as opposed to the Italian (but also French and German) corresponding predicate "avere fame", where the verb BE has been turned to HAVE. The two lexical forms are listed below,

d) pred_v(be,copul,state,evaluative,np/subj/theme_bound/[_], acomp/prop/[_]).

As can be seen from a comparison of the two lexical forms, the purported thematic divergency simply amounts to a difference in the association of a GF to the same s-role, the PROP role. The theoretical framework then provides a natural and simple way to adjust this difference in terms of change of constituency. In particular, a difference in a GF which is an open function will certainly carry over to a difference in the lexical realization of a given s-role, because open functions are encoded in terms of their lexical head: i.e. an XCOMP varies in the lexical head according to the value of X which ranges over lexical categories, P,V,A,N. Differently then from what happens with a closed function, an OBL which is turned into an OBJ by the GMR, a propositional argument will require the substitution of the
GF associated to it and this in turn will cause the instantiation of a different lexical head. To find the adjective corresponding to the noun 'fear' the GMR checks every functional change triggering categorial mapping if necessary. Notice that the mapping between different lexical categories is supposed to be an intralingual one.

1. OBJ2:experiencer, VERB, SUBJ:theme_emot
   
   ==> 
   
   SUBJ:experiencer, VERB, OBJ:theme_emot

italian: al lupo piace mart\a
english: the wolf likes mart\a

---------source FS---------
prop:main
modo:ind
pred:piacere
tempo:pres
cat:emotivo
funcs:
   obj2:
      role:experiencer
      pred:lupo
cator(animate, human)
num:sing
gen:mas
pers:3
spec:
def:+
indice:'.1'
tab_ref:
   + ref - pro - ana + class
subj:
   role:theme_emot
   pred:marta
cator(animate, human)
gen:fem
pers:3
num:sing
indice:'.2'
tab_ref:
   + ref - pro - ana - class
idiom:
funcs:[]
adj:[{}]

---------target FS---------
prop:main
pred:like
tempo: pres
funcs:
    subj:
        role: experiencer
        num: sing
        pred: wolf
        cat: or([animate, human])
        pers: 3
        spec:
            def: +
            ind: '1'
            tab_ref:
                + ref - pro - ana + class
    obj:
        role: theme_emot
        pred: maria
        pers: 3
        cat: or([animate, human])
        num: sing
        ind: '2'
        tab_ref:
            + ref - pro - ana - class
idiom:
    func: []
adj: []

2. SUBJ: experiencer, VERB, NCOMP: state
\[\Rightarrow\]
SUBJ: theme_bound, VERB, ACOMP: state

italian: tommy aveva paura
english: tommy was afraid

prop: main
modo: ind
pred: avere
tempo: imp
cat: state
funcs:
    subj:
        role: theme_bound
        pred: tommy
        cat: or([animato, umano])
        gen: mas
        pers: 3
        num: sing
        ind: '1'
        tab_ref:
            + ref - pro - ana - class
    ncomp:
        role: state
prod:paura
num:sing
gen:fem
pers:3
cator([emotive])
idiom:
funcs:[]
adjx:[]

_________________target  FS_________________

prop:main
tempo:past
pred:be
funcs:
subj:
  role:theme_bound
  pred:tommy
  pers:3
  num:sing
  indice: '_1'
tag_ref:
  + ref - pro - ana - class
acomp:
  role:state
  pred:afraid
  num:sing
  pers:3
cator([emotive])
idiom:
  funcs:[]
  adjx:[]

The same would apply to the couple of predicates, "volere" and "want" which have a PROP argument realized as a closed function, an SCOMP, in the Italian lexical form and as an open function, a VCOMP, in the English one. As Eckert and Heid(1988) also remark in their paper, equivalence conditions in the lexical transfer within LFG may involve control structures, (see, ibid.,181), and this is what will happen when the divergence regards a transfer between a closed PROP function and an open one. The theory has a default lexical rule for functional control which applies to open functions like VCOMP, but no such rule will be called for SCOMPs. In this case, however, the GMR will have the additional task to suppress one argument from the lexical form where the OBJ is the lexically assigned controller of the open function, and this will be computed as the SUBJ of the closed function argument. Here below we list the two lexical forms,
f) pred_v(volere, trans, state, subjective, np/subj/actor/ [+human], scomp/prop/[_]/[subj=x]).
g) pred_v(want, trans, state, subjective, np/subj/actor/ [+human], np/obj/theme_unaff/[+human], vcomp/prop/to/ [subj=objc/theme_unaff]).

There are two more interesting cases which are naturally dealt with in the theoretical framework we propose: the first is the inchoativized or intransitivized form of an otherwise transitive verb like "muovere"/move or "nutrire"/feed. Lexical forms may undergo restructuring in case a lexical redundancy rule has applied and has turned a transitive predicate which has two arguments into an intransitive one, with only one argument. In the sentences,

h) La casa non si mosse/The house did not move

i) Il lupo si nutriva di porcellini/The wolf fed on little pigs

the predicates MUOVERE and NUTRIERE may be computed as intransitive predicates in both languages because the GMR takes as input the information available in f-structure: in turn, the f-structure realization of a lexical form may be the same or be different from the underlying one. The English equivalent naturally treated as intransitives because they are ambiguously classified. The choice of one or the other form depends in our case on the requirements imposed by the source f-structure representation.

The other grammatically interesting case to be discussed regards the case in which two preds subcategorize for a different number of roles/functions, following systematic differences between the two languages.

This is a case of systematic divergence between Italian and English. Italian can freely add a reflexive BENEFactive role to the lexical form of transitive verbs in a systematic way, as shown by the following example,

l) I porcellini si costruirono una casa/The little pigs built a house

Notice that the original lexical form of COSTRUIRE does not contain a benefactive role. This is added by a lexical redundancy rule in the grammar that modifies dynamically the lexical form of certain classes of verbs. As this do not apply to English, the GMR systematically drops the added role from the target FS.
3. SUBJ:agente, OBJ2:benef, VERB, OBJ:tema_aff

==>

SUBJ:agente, VERB, OBJ:tema_aff

Italian: i porcellini si costruirono una casa

English: the little_pigs built a house

prop:main
mood:ind
pred:costruire
tense:past
cat:accomplishment
subj:
  role:agente
  pred:porcellino
cator((animato, umano))
gen:mas
pers:3
num:plur
ind:'_1'
spec:
def:+
tab_ref:[+ ref - pro - ana + class]

obj2:
  role:benef
  pred:si
case:dat
pers:3
ind:'_2'
gen:mas
num:plur
spec:
def:+
tab_ref:[ - ref - pro + ana + me - subj]
antecedent:'_1'
interpretation:specific

obj:
  role:tema_aff
  pred:casa
cator((obj, place))
nun:sing
gen:fem
pers:3
spec:
def:-
ind:'_3'
tab_ref:[ + ref - pro - ana + class]

idiom:
func:[{}]
adj:[{}]

prop:main
pred: build
mood: ind
tense: past
subj:
  role: agente
  pred: little pig
  pers: 3
  num: plur
  ind: '1'
  spec: def +
  tab_ref: [+ ref - pro - ana + class]
obj:
  role: tema aff
  pred: house
  pers: 3
  num: sing
  spec: def -
  ind: '3'
  tab_ref: [+ ref - pro - ana + class]
idiom:
  funcs: []
  adjst: []
6. Transferring Idiomatic Chunks

In her paper "The passive in Lexical Theory" (1982, chapter 1, pp. 45-50), Bresnan discusses the problem of idiom chunks in terms of nonsemantic selection of idiom objects. An idiom object is a grammatical object that is not an argument of a predicate. In our framework, a nonargumental function is encoded as "Theme_Bound", whereas in the theory it is simply not associated to a predicate argument, given the fact that a-structure is a separate lexical representation from lexical form, where grammatical functions are assigned. However, the final net result is the same; i.e. the OBJect is grammatically inherited by the corresponding passive lexical form via a lexical redundancy rule but is assigned no semantic interpretation. In particular, then, in the case of idiomatic objects, this will be detectable by the additional information that the lexical head associated to the OBJ is a FORM and that it must be a particular word, this one recorded in a special lexical equation. The lexical entry for "keep tabs on" is as follow,(her number)

(86) a. keep: V,'KEEP-TABS-ON((SUBJ),(ON OBJ))'(OBJ FORM) =c TABS

This case is represented by the first example we will discuss from our text, which is constituted by an idiomatic verbal pred corresponding to a non idiomatic one.

To explain this example, we should explain first how parser and generator handle idiomatic chunks. An idiomatic sentence is supposed to exhibit grammatical functions that do not correspond to actual semantic roles. These functions are characterized by the fact that they contain 'form' attributes instead of preds. Idiomatic sentences need also a reinterpretation of the literal roles, i.e. roles that would be subcategorized by a literal use of the verb. The information needed to detect an idiomatic pattern is contained in the 'idiom:funct' attribute associated to verbs in the lexicon. Idiomatic information will typically contain instructions not to associate semantic roles to grammatical function that would be interpreted in the literal use (for ex. 'role:to_nil(locative)' and 'form:calcagna') and instructions to associate to a function a role different from the literal one (for example to(theme_bound,agent)).
If an Italian idiomatic verb correspond to a non idiomatic one in the target language, the GMR will drop the non semantically interpreted functions from the f-structure and try a match between the source reinterpreted roles (that are actually present in the f-structure) and the target literal roles. If this match will succeed, all we need to do is to handle possible functional divergences between the corresponding roles.

4. SUBJ:to(actor,theme_aff),VERB, NCOMP:to(theme_bound,agent), OBL:to_nil(locative)
==>
SUBJ:agent,VERB,OBJ:theme_aff

Italian: i porcellini avevano il lupo alle calcagna
English: the wolf was chasing the little pigs

prop:main
mood:ind
pred:averre
tense:imp
cat:state
obj:
  form:calcagna
  cat:or([luogo, oggetto])
  num:plur
gen:fem
pers:3
spec:
  def:+
  ind:'_1'
tab_ref: [+ ref - pro - ana + class]

ncomp:
  role:theme_bound
  pred:lupo
  cat:or([animate])
  num:sing
gen:mas
spec:
  def:+
  ind:'_2'
tab_ref: [+ ref - pro - ana + class]
pers:3

subj:
  role:actor
  pred:porcellino
  cat:or([animate])
gen:mas
pers:3
num:plur
Another interesting example is constituted by the idiom chunk "to pull someone's legs" which has the idiomatic meaning "to play a joke on, or tease someone" as discussed in Bresnan (ibid.48). As she notes, the idiomatic reading does not arise with verbs other than "pull", or with objects other than "[possessor's] leg". What is interesting in this predicate is the fact that both the literal and the idiomatic meaning
must be preserved and allowed to be assigned simply from
the same underlying lexical form. This case is also present
in our text and is an example with an idiomatic verbal pred
corresponding to a pred with a different idiomatic pattern.
This example is similar to the previous one but now the
 correspondence is between two idiomatic verbs. A
reinterpreted role can now correspond to another
reinterpreted role. In this case the latter is included in the
target FS. Furthermore, if the idiomatic information of the
target pred contains non interpretable functions these too are
added to the target FS. Notice that two systematic difference
show up in this example as well. First the addition of the
reflexive benef role, that is in fact eliminated by the GMR.
Second, the necessity to express in English the owner of a
body part which in order to preserve the idiomatic meaning
must be bound to the SUBJect. This is handled as an
intralingual constraint by the generator, that would
automatically generate a possessive in such cases. However,
the literal meaning would be automatically generated
whenever the benefactive is associated to a clitic other than
the reflexive one and consequently the OBJect’s possessor
would be computed as a different individual from the
SUBJect of the main predicate.

5. SUBJ:to(actor,experiencer),OBJ2:to_nil(benef),
   VERB,OBJ:to_nil(theme_unaff)
   ==> SUBJ:to(actor,experiencer),VERB,OBJ:to_nil(theme_unaff)

Italian: il lupo si leccava i baffi
English: the wolf was licking his lips

prop:main
mood:ind
pred:leccare
tense:imp
cat:activity
subj:
    role:experiencer
cat:or([animate, human])
gen:mas
pers:3
num:sing
ind:_'1'
spec:
def:+
tab_ref:[ + ref - pro - ana + class]
pred: lupo

obj2:
  form: si
  ind: '2'
  cator([animate, human])
  gen: mas
  num: sing
  case: dat
  pers: 3
  spec:
    def: +
  tab_ref:[ - ref - pro + ana + me - subj]
  antecedent: _]
  interpretation: specific

obj:
  form: baffo
  cator([obj])
  num: plur
  gen: mas
  pers: 3
  spec:
    def: +
    indice: '3'
  tab_ref:[ + ref - pro - ana + class]

subj:
  role: poss
  cator([__])
  gen: _
  pers: 3
  num: _
  ind: '2'
  spec:
    def: +
  tab_ref:[ + ref - pro - ana + class]
  pred: vbl

idiom:
funcs:
  subj:
    role: to(actor, experiencer)
  obj2:
    role: to_nil(benef)
form: si
obj:
  role: to_nil(theme_unaff)
form: baffo
num: plur
spec:
  def: +
  adjs:[ ]

__________________________target FS__________________________

prop: main
mood: ind
tense: past_prog
7. GraFo: the Generation Algorithm
One can think of GraFo (for a more detailed description see Pianta, 1992) grammar rules producing a syntactic tree that is further passed to an interpretation algorithm producing a semantic representation. Nevertheless we feel that GraFo does its best in a framework akin to that proposed by Fenstad et al. (1987: 15): “In our approach, all levels of linguistic description have equal theoretical status. They all stand in a mutually constraining relationship. There are principles internal to each level which determine the properties and well-formedness of the representation on that level.” Thus there are indeed different description levels with their separate principles, but they are not built one on the top of the other but rather they mutually constrain each other. “In the constraint based theory the focus is not on the derivation or construction of one level (say semantics) on the basis of another (typically syntax). The ‘inter-level’ constraints ... are declarative descriptions of the relationships holding between
aspects of linguistic form and the semantic representation itself. From an execution point of view this amount to think of Natural Language Generation Process as exploiting in every moment all the (syntactic, semantic, pragmatic...) available information.

It should be pointed out that with respect to a strictly constraint based approach GraFo encourages to afford a central status to constituency. All other constraints and description levels are attached to rules specifying first of all constituents or word order. We do not feel this is a flaw. Linear order is indeed a central characteristic of human language heavily conditioning both parsing and generation. Rewriting rules (more or less context free) may not be the best way to represent word order, but they seem very effective at least for the most part of Romance and Germanic languages. The generation transducer starts from Rosie's source grammar and follows basically a top down strategy. This is not very far from what happens when one uses DCGs for generation purposes, apart of course from the unification mechanism. Two important remarks should be made, however: one regarding the selection of rules, the other regarding the flow of information from rule to rule. Rewriting rules cannot apply blindly, without any exit condition: suppose that rules are applied starting from a (syntactic or semantic) structure that is progressively "consumed" during computation and that no rule can be applied if such structure is void.

We can think thus that every GraFo rule must have a particular feature containing information that guides rule selection and finally ends the generation process. Although this seems quite sensible, a problem arises from the unification mechanism we have chosen. Subsumption-based unification is not adequate to guide rule selection simply because the empty constraint (sub)unifies with everything, and more generally the absence of structure does not prevent two features to unify. Two solution are available in GraFo. The first consists in using a Prolog term as value of the generation guiding attribute. In GraFo the value of an attribute can in fact be a set of features or any Prolog term. In the latter case standard Prolog unification is called. The second solution is to invoke a special unification mechanism for the generation guiding attribute. GraFo allows assigning
to an attribute one of two special unification mechanism: equiv and equiset. The assignment can be persistent, through a global declaration, or temporary, through the use of a local operator. ‘equiv’ unification recursively requires strict equivalence between attributes clusters, while in the ‘equiset’ based unification the equivalence relation must obtain only for the topmost nodes of the cluster. For the lowest nodes only a ‘subunify’ relation is required.

What we mean by the problem of information flow in sentence generation can be made clear by considering the generation of the sentence: “nello stesso luogo viveva un terribile lupo”/in the same place lived a terrible wolf. Suppose that you have a set of context free rules with attributes, that you have chosen situation semantics as semantic representation formalism and LFG as syntactic framework. The situation semantics representation of the sentence uses semantic roles to label arguments and works as generation guiding attribute. In the LFG based framework, information about how to realize predicate arguments are to be found in the lexicon, namely in a verb entry. This means that in all languages where verb is not the first element of the sentence (i.e. most languages) there is a flow of information from verb to the preverbal argument inverse to the word order. One is tempted to generate the verb before its arguments, in order to retrieve information about the grammatical function that must be used to express semantic roles. But now consider that in Italian there is subject/verb agreement and that in most cases NP subject comes before verb. This means that if you want to generate the right verbal form you should generate the preverbal subject before the verb. In other words there is a cross information flow between verb and the preverbal argument.

One could think that postverbal arguments do not suffer from information flows contrary to word order, but this is false because in Italian NP subjects can be found freely in postverbal position (and in English when locative inversion takes place), as in the example above where again we have a cross information flow between verb and postverbal argument.

If we use a top down generation strategy, inverse flows of information can be solved by a generate and test behaviour. But this will cause the system to work in a highly inefficient
and unnatural way. A solution to this problems could be abandoning the depth-first, left-to-right traversal of the generation tree typical of the top down strategy, in favour of a "post-order traversal (with prediction acting as a pre-order filter)". A detailed discussion of this proposal is out of the scope of this paper. Let's just mention three problems we see: 1) For the mechanism to work we are bound to choose a certain semantic formalism and use it in a fixed way. 2) Some of the problems the algorithm tries to solve arise from a wrong use of the generation guiding attribute. 3) The mechanism seems not to handle cross information flows. 4) The order in which words are generated is unnatural.

The solution we propose tries to generate words in the natural left to right order. This means that preverbal arguments are generated before verbs and so on.

8. THE GENERAL MAPPING ROUTINE
The following algorithm is the outcome of the conjoined effort of a team including a number of people, among whom the actual programmer E.Pianta, and the compiler of a doctoral dissertation on Machine Translation, P.Rossi. The process of translation is thought to have the following structure: source sentence ==> parser ==> source FS ==> FS mapping (with source and target subcat list) ==> target FS ==> generator ==> target sentence.

*******************************************************************************
fs_map(+SourceFS, -TargetFS) takes as input SourceFS represented as a list of lists and recursevely maps all the attribute contained in it. Certain attributes are handled contextually, for example pred and funcs, i.e. the verbal predicate and the list of grammatical function.
*******************************************************************************
End of recursion. The whole FS has been mapped
*******************************************************************************
fs_map([], []).
*******************************************************************************
If a functional structure contains a pred and a funcs attribute, i.e a predicate/arguments pattern, map them contextually, then map the rest of the structure
*******************************************************************************
fs_map(
    [Fun |S],
void pred_preds_map(Pred, Preds, Funcs, FuncsPreds, IdiomsPreds) {
    remove(pred:Pred, [Fun | S], Preds),
    remove(funcs:Funcs, RestFuncs, RestFuncs1),
    !,
    pred_preds_map(Pred, Funcs, Preds, FuncsPreds, IdiomsPreds),
    (IdiomsPreds=""->Idioms=[funcs([]) ; Idioms=IdiomsPreds],
    fs_map(RestFuncs1, RestFuncsPreds).
}

/kg
If a functional structure doesn't contain a pred and funcs attribute, map the attributes one by one
/kg
fs_map([Fun | Funcs], [TrFun | TrFuncs]) :-
    fs_map(Fun, TrFun),
    !
    fs_map(Funcs, TrFuncs).
/kg
If there is no explicit information about how to treat a specific attribute, drop it from the target FS
/kg
fs_map([_ | Funcs], TrFuncs) :-
    !.
    fs_map(Funcs, TrFuncs).
/kg
The prop, index, pers, num, tab_ref and spec attr are kept unchanged. Notice that for at least num, tab_ref and spec these is an evident oversimplification. But the mapping of such attributes is out of the scope of the present work
/kg
fs_map(prop:Prop, prop:Prop) :-
    !.
fs_map(index:Ind, index:Ind) :-
    !.
fs_map(pers:Pers, pers:Pers) :-
    !.
fs_map(role:Role, role:Role) :-
    !.
fs_map(num:Num, num:Num) :-
    !.
fs_map(tab_ref:TabRef, tab_ref:TabRef) :-
    !.
fs_map(spec:Spec, spec:Spec) :-

The pred is mapped by the pred_map function. Notice that the preds treated by this definition are those that have not an explicit predicate/arguments pattern, i.e., there is no funcs attribute at the same FS level in which they are.

```prolog
fs_map(pred:Pred, pred:PredTr) :- !,
    pred_map(Pred, PredTr). 
```

The tempo attribute is mapped by the tens_map function

```prolog
fs_map(tempo:Tense, tempo:TenseTr) :- !,
    tens_map(Tense, TenseTr). 
```

The mods (modifiers) and adjs (adjuncts) attributes are mapped by recursively mapping the FS they label

```prolog
fs_map(mods:Mods, mods:TrMods) :- !,
    fs_map(Mods, TrMods).
fs_map(adjs:Adjs, adjs:TrAdjs) :- !,
    fs_map(Adjs, TrAdjs). 
```

To map the pred & funcs attributes we first search for a specialized function. Specialized functions are supplied for all those couples of source and target verbal preds that have functional and thematic discrepancies not reducible to generalized patterns

```prolog
pred_funs_map(Pred, Funcs, PredTr, FuncsTr, IdiomTr) :-
    pm(Pred, PredTr, Funcs, FuncsTr, IdiomTr), !.
```

If there isn't a specialized function to convert a certain verbal pred then use the general rules. To do this, map the verbal pred, then find the subcategorization lists of the source and target pred and supply them to the funcs_map function along with the source functions and the (possibly
/********************************************
The pred is mapped by the pred_map function. Notice that the preds treated by this definition are those that have not an explicit predicate/arguments pattern, i.e., there is no funcs attribute at the same FS level in which they are.

********************************************
fs_map(pred:Pred, pred:PredTr) :-
!,
pred_map(Pred, PredTr).

********************************************
The tempo attribute is mapped by the tens_map function

********************************************
fs_map(tempo:Tense, tempo:TenseTr) :-
!,
tense_map(Tense, TenseTr).

********************************************
The mods (modifiers) and adjs (adjuncts) attributes are mapped by recursively mapping the FS they label

********************************************
fs_map(mods:Mods, mods:TrMods) :-
!,
fs_map( Mods, TrMods).

fs_map(adjs:Adjs, adjs:TrAdjs) :-
!,
fs_map(Adjs, TrAdjs).

********************************************
To map the pred & funcs attributes we first search for a specialized function. Specialized functions are supplied for all those couples of source and target verbal preds that have functional and thematic discrepancies not reducible to generalized patterns

********************************************
pred_funcs_map(Pred, Funcs, PredTr, FuncsTr, IdiomTr) :-
  pm(Pred, PredTr, Funcs, FuncsTr, IdiomTr),
!.

********************************************
If there isn't a specialized function to convert a certain verbal pred then use the general rules. To do this, map the verbal pred, then find the subcategorization lists of the source and target pred and supply them to the funcs_map function along with the source functions and the (possibly
null) idiomatic constraints on func of the target pred. You'll get the func to be included in the target FS as output of the mapping

```
/********************
pred_funcs_map(Pred, Funcs, PredTr, FuncsTr, IdiomTr) :-
pred_map(Pred, PredTr),
select_grammar(italian),
constraints_pred(Pred, verbo..., _, _, SubCat, _),
select_grammar(english),
constraints_pred(PredTr, verbo..., _, _, SubCatTr, IdiomTr),
extract_funcs(IdiomTr, IdiomFuncsTr),
funcs_map(
    SubCat, SubCatTr, IdiomFuncsTr, Funcs, FuncsTr).
```

```
/********************
funcs_map
(+SubCat, +SubCatTr, +IdiomFuncsTr, +Funcs, -FuncsTr).
/********************
```

End of recursion. All the func have been mapped

```
funcs_map([], [], [], [], [] :- !.
```

```
/********************
== Eliminating target FORMS.
If a source func contains a FORM attribute, do not include the func in the target FS. If the func is present in the original subcategorization list of the source pred, i.e. it hasn't been added by a general lexical rule, absorb it.
```

```
funcs_map(SubCat, SubCatTr, SubCatTr, SubCatTr, SubCatTr).
```

```
== Adding source FORMS.
```
If in the idiomatic constraints of the target pred there is a function with a 'role:to_nil(') attribute, add the idiomatic function to the target FS. If the original subcat list of the target pred contains the func, absorb it.

```prolog
funcs_map(SubCat,
    SubCatTr,
    IdiomTr,
    Funcs,
    [Fun:RestFsFunIdiom | OtherFuncsTr])
    :-
        remove(Fun:FsFunIdiom, IdiomTr, RestIdiomTr),
        remove(role:to_nil(_), FsFunIdiom, RestFsFunIdiom),
        (remove(Fun:_, SubCatTr, RestSubCatTr)
            -> NewSubCatTr=RestSubCatTr
            ; NewSubCatTr=SubCatTr),
        !,
        funcs_map(
            SubCat, NewSubCatTr, RestIdiomTr, Funcs, OtherFuncsTr).
```

Search a source FS with a certain Role and a certain Function. Then search the same Role in the target subcat list or in the idiomatic constraints list as interpreted role and find the target Function. Map the original into the target Function, changing the argument Pred category if necessary.

```prolog
funcs_map(
    SubCat,
    SubCatTr,
    IdiomTr,
    Funcs,
    [FunTr:[role:Role |- RestFsTr] | OtherFunTr])
    :-
        remove(Fun:FsFun, Funcs, RestFuncs),
        remove(role:Role, FsFun, RestFsFun),
        (remove(IdioFun:IdiomFunFs, IdiomTr, RestIdiomTr),
         remove(role:to(LeterRoleTr, Role), IdiomFunFs, _)
            -> RoleTr=LeterRoleTr,
            NewIdiomTr=RestIdiomTr
            ; RoleTr=Role,
            NewIdiomTr=IdiomTr
        ),
        remove(Fun:_, SubCat, RestSubCat),
        remove(FunTr:_, role:RoleTr |- _,
            SubCatTr, RestSubCatTr),
        !,
        funcs_map(
            SubCat, NewSubCatTr, RestIdiomTr, Funcs, OtherFuncsTr).
```
func_map(Fun, FunTr, RestFsFun, RestFsTr),
func_map(RestSubCat, RestSubCatTr, NewIdiomTr,
RestFuncs, OtherFunTr).

/************************************
If the source Fs contains an obj2 function with [pred:si, caso:dat,
role:benef] attributes and the function is not contained in the subcat lists
of neither the source nor the target pred, do not include it in the target FS
************************************/
func_map([], [], IdiomTr, Funcs, OtherFunTr)
:-
  remove(obj2:FSFun, Funcs, RestFuncs),
  nogen_member(pred:si, FSFun),
  nogen_member(caso:dat, FSFun),
  nogen_member(role:benef, FSFun),
  !,
  func_map([], [], IdiomTr, RestFuncs, OtherFunTr).

/************************************
If an Ncomp function is to be mapped into an Acomp function, then map
the source pred into the target pred and convert the category of the mapped
pred from noun to adjective
************************************/
func_map(ncomp, acomp, FsFun,
[pred:PredTrConv | RestFsFunTr])
:-
  !,
  remove(pred:Pred, FsFun, RestFsFun),
  pred_map(Pred, PredTr),
  name_to_agg_english(PredTr, PredTrConv),
  fs_map(RestFsFun, RestFsFunTr).

/************************************
In all other cases simply map the source semantic form in the target one
************************************/
func_map(_, _, FsFun, FsFunTr) :-
  fs_map(FsFun, FsFunTr).

pred_map(A, B) :-
pm(A, B).

/************************************
Specialized pred mapping
************************************/
pm(nutrire, feed_on, FuncsIn, FuncsOut, [func:[[]]]) :-
equiv(FuncsIn,
  [subj:{indice:X | RestSogg},
   obj:
     [pred:si,
       antecedent:X | _],
   obl:Obl]),
fs_map(RestSogg, RestSoggTr),
fs_map(Obl, OblTr),
FuncsOut = {sogg:RestSoggTr, obl:OblTr}.

***************
Verbal pred mapping
***************

pm(correre, run).
pm(cantare, sing).
pm(vivere, live).
pm(decidere, decide).
pm(proteggere, protect).
pm(essere_cop, be_cop).
pm(piace, like).
pm(costruire, build).
pm(aver_emot, be_emot).
pm(aver_alle_calcagna, chase).
pm(leccarsi_i_baffi, lick_ones_lips).
pm(nutrire, feed).

***************
Nominal and adjectival pred mapping
***************

pm(saggio, wise).
pm(jimmi, jimmi).
pm(marta, marta).
pm(tommy, tommy).
pm(lupo, wolf).
pm(ragazzo, boy).
pm(porcellino, little pig).
pm(canzone, song).
pm(paura, fear).
pm(campagna, country).
pm(casa, house).

References


Delmonte R., Bianchi D. (1991a), Binding Pronominials with an LFG-Based Parser, II° IWTP91, Cancun.


Pianta E. (1992), Generating Answers in Natural Language with GraFo, Ms, University of Venice.