SEMANTIC INTERPRETATION OF GAPPED COORDINATION IN FRENCH[1]

Louisette Emirkanian* **Lorne H. Bouchard****

* *Département de linguistique*
** *Département d'informatique*
Université du Québec à Montréal

Abstract: Based upon the Generalized Phrase Structure Grammar (GPSG) (Gazdar, *et al.*, 1985) account of gapped coordination which is put forward by Bouchard, *et al.* (Forthcoming), we describe the implementation of a λ-PROLOG prototype designed to explore the computational strategies for recovering the semantic interpretation of the gaps. A validation of the GPSG account is a by-product of the implementation.

Keywords: coordination, gapping, GPSG, semantic, interpretation.

1. INTRODUCTION

Coordination is an interesting phenomenon of natural languages which must be dealt with in a natural language processing system. In this article we shall focus on the type of coordination which is illustrated in examples (1a-b) below.

(1)	a	Pierre offre des fleurs et Luc des chocolats. *Peter offers flowers and Luke chocolates.*
	b	Pierre offre des fleurs pour Noël et Luc pour Pâques. *Peter offers flowers for Christmas and Luke for Easter.*

1 Research conducted under a programme funded by the SSRC (410-97-0419) and the FCAR (98-ER-1198).

ISBN: 0 08 043 438X

This type of coordination is called *non-constituent coordination,* since a list of constituents follows the *et (and)* instead of a single constituent. In the coordinated sentence[2] to the right of the *et,* when the verb is omitted, other constituents can or must also be omitted, but some must be kept.

Intuitively coordination establishes a parallelism between propositions. In our GPSG account, isomorphic analysis trees are a reflection of the syntactic parallelism between coordinating and coordinated sentences and this syntactic parallelism is used to induce a semantic parallelism.

2. LINGUISTIC CONSTRAINTS

We briefly review the constraints which are described in Bouchard, *et al.* (Forthcoming)[3]. First of all, gapping is not unique to verbs but applies to all elements marked +V, that is adjectives and adverbs also (Bouchard, *et al.*, 1996), and only those elements. Without biasing the analysis, we can visualize the elision as a gap which can propagate in certain cases. We call the element which triggers the gap the *principal elided element.* The analysis has shown that at least an *emphasis* (subject, controller in the case of infinitives or extracted complement) and a complement or adjunct (in canonical position) of the principal elided element must always remain. Starting from the principal elided element, heads of other +V constituents which directly dominate the principal elided element must be omitted as well as all other constituents on the level except the controller of a gapped constituent. The following examples illustrate these constraints.

(2) À 6 heures, Pierre boit son pastis et à 10 heures, _ son cognac.
At 6, Peter drinks his whisky and at 10, his brandy.

(3) Marie est contente de sa voiture et Louise _ de sa moto.
Mary is happy with her car and Louise with her bike.

(4) Pierre promet aux enfants de décorer la citrouille et Jacques _ le gâteau.
Peter promises the children to decorate the pumpkin and Jack the cake.

(5) Pierre permet à ses enfants de manger la glace et à ses parents _ la tarte.
Peter allows his children to eat the ice cream and his parents the pie.

(6) Pierre permet à ses enfants de manger la glace et Luc _ à ses parents _ la tarte.[4]
Peter allows his children to eat the ice cream and Luke, his parents the pie.

(7) Pierre demande que son fils envoie des fleurs et Luc _ sa fille _ du parfum.
Peter requests that his son send flowers and Luke, his daughter perfume.

As examples (6) and (7) above illustrate, the emphasis of the matrix sentence may be kept, but all other elements of the matrix must be omitted.

---

2 In order to simplify the presentation, we only treat binary coordination with *et.* We use the term *coordinating sentence* to refer to the sentence to the left of the *et* and *coordinated sentence* to refer to the sentence to the right.

3 This paper compares GPSG and HPSG analyses of the phenomenon.

4 In our opinion, this sentence is as grammatical as the following one in which three elements are also taken up:
À 6 heures, Pierre boit son pastis et à 10 heures, Paul _ son cognac.
At 6, Peter drinks his whisky and at 10, Paul his brandy.
We should point out however that this judgment is not shared by all speakers.

ISBN: 0 08 043 438X

Our GPSG account addresses two fundamental questions which are developed here.

1- How can the analysis of the coordinating sentence be used *effectively* to analyze the coordinated sentence?

2- How can the semantic interpretation of the elements omitted in the coordinated sentence be recovered from the analysis of the coordinating sentence?

3. GPSG ACCOUNT

Generalized Phrase Structure Grammar as defined in Gazdar, *et al.* (1985) is a theory of universal grammar which uses a single level of representation, that of surface strings. GPSG factors the rewriting rules of traditional context-free grammar into Immediate Dominance (ID) and Linear Precedence (LP) rules. Categories are defined as under-specified sets of features-values, a feature value being either atomic or a set of feature-values. Feature Specification Defaults (FSD) declare default values for features. Feature Coocurrence Restrictions (FCR) are used to constrain categories. Universal feature instantiation principles — Head Feature Convention (HFC), Foot Feature Principle (FFP) and the Control and Agreement Principle (CAP) — are used to determine if the distribution of categories in a local tree form a legal projection of a given ID rule. Metarules generate new ID rules from lexical ID rules.

Our GPSG account of gapped coordination follows the analysis of Russell (1987). In this analysis, the presence of a gap is signaled by a +G feature. One of the distinguishing features of our account is the isomorphism induced between the analysis trees for the coordinating and coordinated sentences which reflects the *syntactic parallelism* between the sentences. In the analysis tree of the coordinated sentence, omitted verbal heads are marked +NULL and other omitted elements, +ELIDED.

The coordinating sentence is analyzed using the standard grammar rules. A TEMPLATE feature is used to encode the subcategorization information observed in the coordinating sentence; the value of this feature is then used to constrain the analysis of the coordinated sentence as can be seen in the ID rules of Figure 1 which introduce gapped coordination.

X^2[+V,+COORD] $\rightarrow$ H[TEMPLATE α,-G],H[CONJ et,TEMPLATE α,+G] *gapped coordination*

X^2[+V,+COORD] $\rightarrow$ H[TEMPLATE α,-G],H[CONJ et,TEMPLATE β,-G] *ungapped coordination*

Fig. 1. ID rules for the coordination of +V constituents.

In the case of gapped coordination, the TEMPLATE of the coordinating sentence constrains the coordinated sentences to share the same value α.

3.1 The metarules

The coordinated sentence is analyzed using ID rules which are generated by the metarules of Figure 2.

MR1 $X_1^2[+V] \rightarrow H, X_2^2, W_1, W_2$

$\Downarrow$

$X_1^2[+V] \rightarrow H[+NULL], X_2^2, W_1[+ELIDED], W_2$

MR2 $X_1^2[+V] \rightarrow H, X_2^2, X_3^2[+V], W_1$ *if X_2^2 controls X_3^2*

$\Downarrow$

$X_1^2[+V] \rightarrow H[+NULL], X_2^2, X_3^2[+V,+G], W_1[+ELIDED]$

MR3 $X_1^2[+V] \rightarrow H, X_2^2[+V], W_1$ *if X_2^2 has no local controller*

$\Downarrow$

$X_1^2[+V] \rightarrow H[+NULL], X_2^2[+V,+G], W_1[+ELIDED]$

where W_1 and W_2 are (possibly empty) multisets of categories and W_1[+ELIDED] means that all the categories in W_1 are marked +ELIDED.

Fig. 2. The gapping metarules.

By the first metarule, the principal elided element, of which at least one complement must remain, is marked +NULL. An FCR will instantiate the +G head feature on this element since it is marked +V. The HFC is responsible for the propagation of the G feature.

(8) FCR : $[+G] \supset [+V]$
FCR : $[+NULL] \supset [SLASH] \vee [+G]$

We illustrate the effect of the metarule MR1 on the typical ID rule shown in (9). Note that we treat adjuncts as the most oblique complements of the head, the results of which are bushier analysis trees.

(9) $V^2 \rightarrow H[3], N^2, (X^2[+ADJUNCT])$

Since (9) contains an optional constituent, it is expanded into the two ID rules as shown in (10).

(10) a $V^2 \rightarrow H[3], N^2$
b $V^2 \rightarrow H[3], N^2, X^2[+ADJUNCT]$

Application of the metarule MR1 to the ID rules (10a-b) yields the ID rules shown in (11), in which the positions of the omitted complements or adjuncts are marked +ELIDED.

(11) a $V^2 \rightarrow H[3,+NULL], N^2$
b $V^2 \rightarrow H[3,+NULL], N^2[+ELIDED], X^2[+ADJUNCT]$
c $V^2 \rightarrow H[3,+NULL], N^2, X^2[+ADJUNCT,+ELIDED]$
d $V^2 \rightarrow H[3,+NULL], N^2, X^2[+ADJUNCT]$

As an example, Figure 3 shows the analysis tree for sentence (1b), a case of simple gapping: MR1 was the metarule used to generate the ID rule which licenses the coordinated sentence.

ISBN: 0 08 043 438X

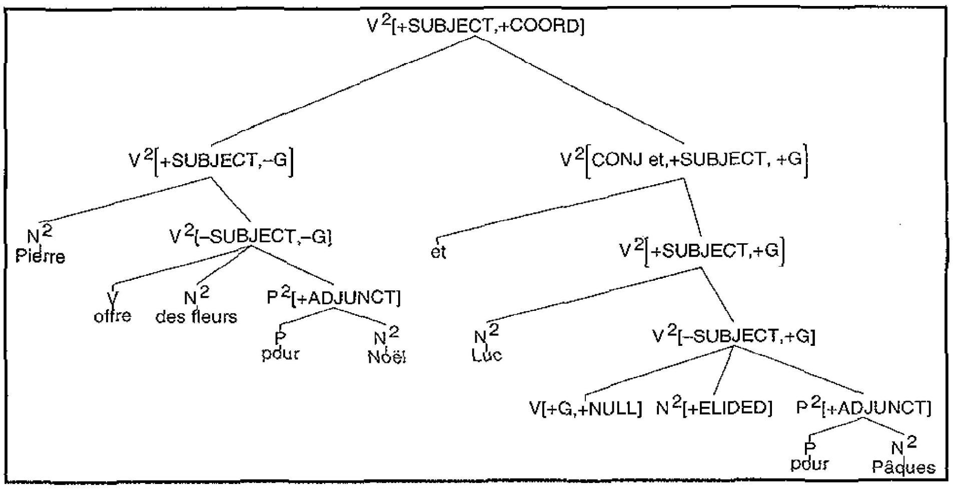

Fig. 3. Pierre offre des fleurs pour Noël et Luc pour Pâques *(Peter offers flowers for Christmas and Luke for Easter).*

As a further example, we illustrate in Figures 4a and 4b the result of the interaction of the ID rules which license the analysis tree of sentence (6), repeated below for easy reference.

(6) Pierre permet à ses enfants de manger la glace et Luc _ à ses parents _ la tarte.
Peter allows his children to eat the ice cream and Luke, his parents the pie.

The V^2 constituents in the trees 4a and 4b are distinguished by a double digit index. The first digit of the index *i* is 0 for the coordination as a whole, 1 for the coordinating sentence and 2 for the coordinated sentence. For any given *i*, the second digit ranks along a given branch the V^2 constituents starting from 1 at the leaves of the tree.

As observed in figures 4a and 4b, there is a one to one correspondence between the V^2 nodes in the coordinating and coordinated sentences.

$$V^2_{1i} \leftrightarrow V^2_{2i} \quad 1 \le i \le 4$$

Constituents which have been omitted in the coordinated sentence are marked as +NULL if verbal heads, and +ELIDED otherwise. It is in this sense that the analysis trees of the GPSG account are said to be *isomorphic.*

The constituent V^2_{23} is licensed by an ID rule produced by MR2 since *à ses enfants (his children)* controls *manger la glace (to eat the ice cream)* and the constituent V^2_{21} is licensed by an ID rule generated by MR1.

Similarly, metarule MR3 generates ID rules which are used to account for sentences such as (4) above in which the embedded clause has no local controller.

The ID rules generated by the metarules over generate greatly. A new TEMPLATE feature is used to constrain the possible gapped coordinated sentences. The TEMPLATE feature is a crucial part of our analysis since it explains how the analysis of the coordinating sentence effectively constrains the analysis of the coordinated sentence.

ISBN: 0 08 043 438X

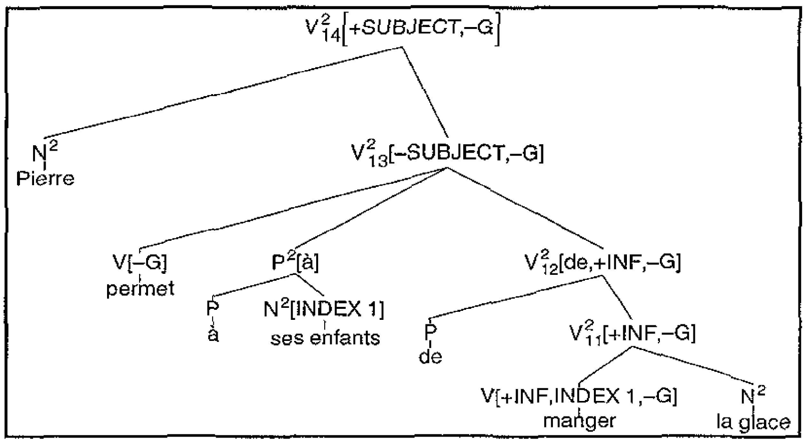

Fig. 4a. Pierre permet à ses enfants de manger la glace *(Peter allows his children to eat the ice cream).*

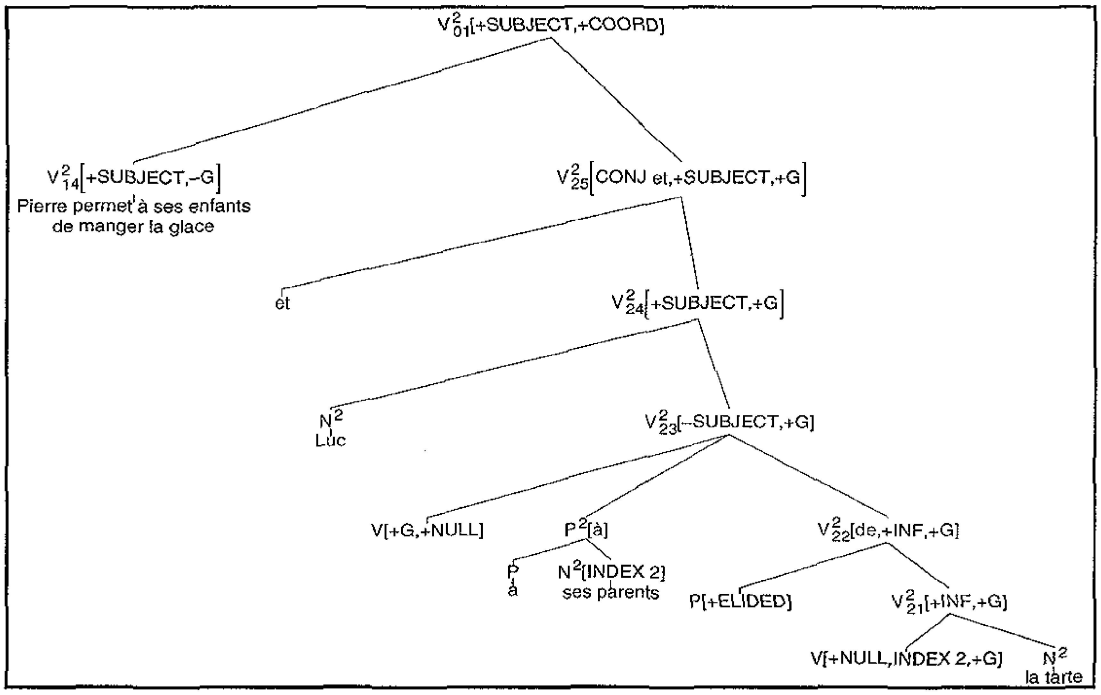

Fig. 4b. Pierre permet à ses enfants de manger la glace et Luc, à ses parents la tarte *(Peter allows his children to eat the ice cream and Luke, his parents the pie).*

In the analysis of Bouchard, *et al.* (Forthcoming) TEMPLATE is a FOOT feature the propagation of which is governed by a new TEMPLATE principle patterned on the FFP. At the expense of slightly more complicated ID rules, we can dispense with the TEMPLATE principle altogether as we show here.

ISBN: 0 08 043 438X

3.2 The template feature

The TEMPLATE feature is a FOOT feature, the value of which is a multiset of categories, sets of feature-value pairs. This feature propagates upwards under the FFP up to the point where it is explicitly specified in an ID rule.

Each ID rule which introduces a X^2(+V) constituent must be modified in order to generate and propagate the TEMPLATE feature.

Lexical ID rules which introduce a V^2 constituent must specify the initial value of TEMPLATE.

(12) a V^2[TEMPLATE {H[3],N^2}] $\rightarrow$ H[3],N^2
b V^2[TEMPLATE {H[3],N^2,X^2[+ADJUNCT]}] $\rightarrow$ H[3],N^2,X^2[+ADJUNCT]

Notice that the SUBCAT feature is included in the TEMPLATE.

A metarule can be used to automate this rewriting.

(13) MR0 X^2[+V,~TEMPLATE] $\rightarrow$ H,W
$\Downarrow$
X^2[+V,TEMPLATE H•W] $\rightarrow$ H,W

in which the • operator is used to *append* the set of features of H to the multiset W.

There is no need to modify the metarules MR1- MR3 if the lexical ID rules which introduce a V^2 constituent are modified in this way. However, all non-lexical ID rules which introduce a V^2 constituent need to be modified to specify how the template feature is propagated.

This alternative technique for generating and transmitting the TEMPLATE feature relies on the FFP and completely does away with the need for a special TEMPLATE principle.

4. λ-PROLOG PROTOTYPE

The strong syntactic parallelism between coordinated sentences which is highlighted by the GPSG analysis can be used to induce a semantic parallelism. In order to empirically test our GPSG account and to evaluate strategies for the semantic interpretation of gapped coordination, we implemented a prototype written in λ-PROLOG, a strongly-typed higher-order logic programming language (Miller, 1989) which is particularly well-suited to the task of defining semantic interpreters.

This prototype makes precise:

1- how the syntax analysis of the coordinating sentence can be used to pilot the analysis of the coordinated sentence;

2- how the semantic interpretation of the gapped constituents in the coordinated sentence can be recovered from the semantic interpretation of the corresponding constituents in the coordinating sentence.

A DCG grammar (Pereira and Warren, 1980) describes a fragment of the grammar of French which we have developed (Emirkanian, *et al.*, 1996). The effect of the GPSG gapping metarules is simulated by expanding by hand the +V ID rules into ID rules which allow gaps. The DCG is then "decorated" to implement the TEMPLATE feature and to produce analysis trees and semantic translations. Since the coordinated sentence is analyzed using a gapped ID rule, its analysis tree records the position of all the constituents of the gap which are marked

ISBN: 0 08 043 438X

as +NULL or +ELIDED. The analysis of a gapped coordinated sentence is constrained by the TEMPLATE *feature which percolates first up (in the coordinating sentence)* and then down (in the coordinated sentence).

Semantic interpretation in the prototype is based on a simple truth functional model in which the semantic interpretation of a sentence is represented by a logical formula. The formula corresponding to a sentence is a conjunction of atomic formulae which are existentially quantified explicitly (Davidson, 1989; Higginbotham, 1989; Parsons, 1990). The semantic translation of a constituent is recursively assembled from the semantic interpretations of its sub-constituents, the assembly instructions being typically λ-expressions (Warren, 1983). During this operation, the prototype assigns unbound logical variables as the semantic interpretations of the gaps.

The semantic interpretation of the gapped constituents in the coordinated sentence is obtained by transferring the semantic interpretations of the corresponding elements in the coordinating sentence.

The prototype can be used to explore how the semantic interpretation of the gapped constituents in the coordinated sentence can be recovered from the interpretation of the coordinating sentence. Using the prototype, we have studied three ways to implement the semantic transfer.

1- Transfer based on the syntactic unification of the analysis trees.
2- Transfer based on the default unification of the semantic interpretations.
3- Transfer based on the application of the anti-unifiers (generalizations) of the interpretation of the coordinating sentence to the interpretations of the fragments of the coordinated sentence.

4.1 λ-PROLOG as a programming language

The prototype is implemented in LPSML93 version of λ-PROLOG, an experimental *higher-order logic programming* language which is built upon the strong logical foundation of hereditary higher-order Harrop clauses (Miller, *et al.*, 1991). As a logical programming language λ-PROLOG is a direct descendant of PROLOG. λ-PROLOG is a strongly typed language which adopts the prefix notation favored by LISP but in which most of the parenthesis can be suppressed because of the strong typing discipline and the built-in operator priorities: this syntax is borrowed from SML which is the underlying implementation language of LPSML93, the version of λ-PROLOG we are using. The syntax of programs in λ-PROLOG is roughly that of pure PROLOG written in *prefix notation*. All instructions are terminated by a ".", identifiers starting with a lower case letter denote constants, whilst those starting with an upper case letter denote variables. There is no automatic type inference (Milner, 1978) in LPSML93 and thus the types of all logical constants must be explicitly declared by a "type" declaration: type expressions, which are called *signatures*, are constructed of type constants and variables using the "->" operator and parentheses. As in pure PROLOG, a function of two variables "f(a,b)=c" is represented as a relation on three variables "r(A,B,C)". λ-PROLOG is said to be *higher-order* since all relations are *curryed*: assuming that "Ta" is the type of argument "A", "Tb" the type of argument "B" and "Tc" the type of argument "C", the signature of the relation "r A B C" is "type r Ta -> Tb -> Tc -> o", where "o" is the built-in type of expressions in λ-PROLOG and "->" is right-associative. The LPSML93 version of λ-PROLOG is implemented using the ELP system (Elliot and Pfenning, 1991). Based upon the type

ISBN: 0 08 043 438X

declarations of the logical constants, all program constructs are type-checked at compile time. There is no syntactic notation for lists which must be constructed using the built-in "::" operator: for example, the PROLOG list "[a,b,c]" is written as "(a :: b :: c :: nil)". The most innovative construct in λ-PROLOG is that of a well-typed *lambda expression*: the lambda expression "λxλyλz.f(x,y)=z" is written as "x\y\z\f x y z".

The advantage of using a strongly-typed programming language is an increased confidence in the results produced by the program, based upon the knowledge that the program code is type secure. Coding in λ-PROLOG is rather straightforward for programmers versed in the symbolic computation style of LISP and PROLOG. One small irritant perhaps is the handling of polymorphic data types which are to be defined by discrete enumeration as for example the *tree* data type used in our prototype the signature of which is presented in Figure 5.

Assuming that *formula* is the type of a formula in our logic, the signature of a polymorphic λ-expression of a single variable is "type A -> formula" and that of a polymorphic λ-expression of two variables is "type A -> B -> formula". Based upon these observations, there can be no signature for an arbitrary *n* argument λ-expression, a serious drawback indeed! The problem is solved in the prototype by defining for our object logic a new closure operation "lambda", analogous to the "exists" (∃) and "all" (∀) operators used in first-order logic, which has the signature "type lambda (A-> formula) ->formula". Thus, the signature of any arbitrary λ-expression can be reduced to that of *formula* by repeated application of this operator. Hence, for example, a λ-expression of three variables "x\y\z\r x y z" is represented as "lambda x\ lambda y\ lambda z\ r x y z" which has the signature *formula*.

4.2 Data structures used in the prototype

In order to follow the implementation of the prototype it is useful to understand the data structures which are used to represent words and sentences, analysis trees and semantic interpretations.

The prototype represents a sentence as a list of words. Analysis trees are represented as instances of the polymorphic `tree` data type, the signatures of which are shown in Figure 5.

```
module trees.

% polymorphic n-adic trees
% NOTE : typically A is of type string

kind tree type -> type.

type empty (tree A).
type word A -> A -> (tree A).
type t0 A -> A -> (tree A).
type t1 A -> (tree A) -> (tree A).
type t2 A -> (tree A) -> (tree A) -> (tree A).
type t22 A -> A -> (tree A) -> (tree A) -> (tree A).
type t3 A -> (tree A) -> (tree A) -> (tree A) -> (tree A).
```

Fig. 5. Definition of the signature of polymorphic trees.

ISBN: 0 08 043 438X

The `kind` declaration introduces a new type name or sort in λ-PROLOG, whilst the `type` declaration is used to assign signatures to constants. The `tree` data type is defined as a discrete union of nodes: each node is characterized by a string label and the different node types distinguish the number of daughters a given node may have: `t0` or `word` represents a leaf node, `t1` a node with one daughter and so forth. Each item in such a data structure is thus stored inside a wrapper which identifies the type of the object which is contained in it: a data item must be wrapped when stored as the node of a `tree` and unwrapped when retrieved from the `tree` for processing.

```
module logic.

% Patterned upon the modules
% "ot_logic.mod" and
% "ot_constants.mod" written by Dale Miller

kind formula type.
kind term type.

% definition of logical constants

type and formula -> formula -> formula.
type exists (A -> formula) -> formula.
type lambda (A -> formula) -> formula.

% definition of the non-logical constants

% definition of term constants
kind identifier type.
kind individual type.
kind spatial-position type.
kind object type.
kind event-type type.
...
% definition of atomic formula constants
type agent identifier -> individual -> formula.
type recipient identifier -> individual -> formula.
type propositional-content identifier -> formula -> formula.
type destination identifier -> spatial-position -> formula.
type event identifier -> event-type -> formula.
type location identifier -> spatial-position -> formula.
type source identifier -> spatial-position -> formula.
type theme identifier -> object -> formula.
...
```

Fig. 6. Definition of the object logic.

The data type `formula` which is characterized by the signatures of Figure 6, defines the syntax of the object logic which is used to encode semantic interpretations in the prototype.

The inductive definition of the object logic hinges on the definitions of `term` and `formula`. The non-logical constants partition the domain of semantic interpretation. The definition of this object logic is interpreted using the built-in metalogic of λ-PROLOG which provides the classical logic constants: pi and sigma for ∀ and ∃ respectively, "->" for ⇒, "," for ∧ and ";"

ISBN: 0 08 043 438X

for v. This particular approach to programming closely follows the path of constructive proof theory (Troelstra and van Dalen, 1988) in mathematics.

4.3 The heart of the prototype

In order to grasp the gist of the prototype, a study of the `s-coord` rule licensing the gapped coordinated sentence is instructive. The `Pi` variables, the first two arguments of `s-coord` and the second and third of `s`, record the input and output lists of words of the grammar rules. In the first call to `s`, the fourth parameter (`non-gapped`) indicates that a gap is not authorized in the coordinating sentence, whilst, in the second call, the `gapped` authorizes a gap in the coordinated sentence. The first argument of the `s` rule, the `G` parameter, is the TEMPLATE feature which is synthesized during the analysis of the coordinating sentence and then used to constrain the analysis of the gapped coordinated sentence. An analysis of the behavior of the prototype has confirmed that this constraint effectively eliminates any spurious analyses using the gapped ID rules. The third argument of `s-coord`, the last but one of `s`, is used to record the analysis tree of the sentence: the analysis tree for the gapped coordinated sentence consists of a `s-coord` level built on top of the analysis trees of the coordinating (`P1t`) and coordinated (`P2t`) sentences. The last argument of `s-coord` is the semantic translation of the gapped coordination sentence: the semantic translation of the coordination conjunction *et* is used to join the translations of coordinating and coordinated sentences.

```
s-coord P1 P4 (t2 "s-coord" P1t P2t) (Cs P1s P2sE) :-
   s G P1 P2 non-gapped P1t P1s,
   conj-coord P2 P3 Ct Cs,
   s G P3 P4 gapped P2t P2s,
   transfer P1t P1s P2t P2s P2sE.
```

The exact coding of the final line of `s-coord`, i.e. the **transfer**, simply depends upon the particular technique chosen to implement the transfer from coordinating to coordinated sentence.

5. SYNTACTIC TRANSFER

In this case the translation of the coordinated sentence is the result of the tree unification (`unify-tree`) of the analysis trees of the coordinating and coordinated sentences. The full form of the coordinated sentence is recovered from the analysis tree and resubmitted for analysis in order to obtain the semantic analysis of the complete (i.e., non-gapped) coordinated sentence.

The process of tree unification involves recursively destructuring in parallel the analysis trees of the coordinating and coordinated sentences in order to synthesize the resulting unified tree.

We can illustrate the behavior of the prototype by showing how sentence (6), repeated here for convenience, is analyzed.

(6) Pierre permet à ses enfants de manger la glace et Luc, à ses parents la tarte.
Peter allows his children to eat the ice cream and Luke, his parents the pie.

ISBN: 0 08 043 438X

```
?- unify-tree
       (t2 "s" (t1 "np" (word "pnoun" "Pierre"))
               (t3 "vp" (word "verb" "permettre")
                        (t2 "pp" (word "prep" "à")
                                 (t2 "np" (word "det" "ses")
                                          (word "noun" "enfants")))
                        (t2 "vp-inf"
                            (word "prep" "de")
                            (t2 "vp" (word "verb" "manger")
                                     (t2 "np"
                                         (word "det" "la")
                                         (word "noun" "glace"))))))
       (t2 "s" (t1 "np" (word "pnoun" "Luc"))
                        (t3 "vp"
                            (word "verb" "null")
                            (t2 "pp" (word "prep" "à")
                                     (t2 "np"
                                     (word "det" "ses")
                                     (word "noun" "parents")))
               (t2 "vp-inf" (word "prep" "elided")
                   (t2 "vp" (word "verb" "null")
                            (t2 "np" (word "det" "la")
                                     (word "noun" "tarte")))))) T.

T = t2 "s" (t1 "np" (word "pnoun" "Luc"))
           (t3 "vp" (word "verb" "permettre")
                    (t2 "pp" (word "prep" "à")
                             (t2 "np" (word "det" "ses")
                                      (word "noun" "parents")))
                    (t2 "vp-inf" (word "prep" "de")
                                 (t2 "vp" (word "verb" "manger")
                                 (t2 "np" (word "det" "la")
                                          (word "noun" "tarte"))))).

?- tree->string
   (t2 "s" (t1 "np" (word "pnoun" "Luc"))
           (t3 "vp" (word "verb" "permettre")
                    (t2 "pp" (word "prep" "à")
                             (t2 "np" (word "det" "ses")
                                      (word "noun" "parents")))
                    (t2 "vp-inf" (word "prep" "de")
                                 (t2 "vp" (word "verb" "manger")
                                 (t2 "np" (word "det" "la")
                                          (word "noun" "tarte"))))))
   S.

S = "Luc" :: "permettre" :: "à" :: "ses" :: "parents" ::
    "de" :: "manger" :: "la" :: "tarte" :: nil.
```

The resulting full version of the coordinating sentence is then resubmitted to syntax analysis to produce an analysis tree and a semantic interpretation of the coordinated sentence as is shown by the following output.

ISBN: 0 08 043 438X

```
?- s G ("Luc" :: "permettre" :: "à" :: "ses" :: "parents" :: "de" ::
        "manger" :: "la" :: "tarte" :: nil)
     R non-gapped T S.

S = exists y\
      and (event y permission)
          (and (agent y luc)
               (and (recipient y parents)
                    (propositional-content y
                        (exists y\
                           and (event y consumption)
                               (and (agent y parents)
                                    (theme y tarte)))))),

T = t2 "s" (t1 "np" (word "pnoun" "Luc"))
           (t3 "vp" (word "verb" "permettre")
                    (t2 "pp" (word "prep" "à")
                             (t2 "np" (word "det" "ses")
                                      (word "noun" "parents")))
                    (t2 "vp-inf"
                        (word "prep" "de")
                        (t2 "vp" (word "verb" "manger")
                                 (t2 "np" (word "det" "la")
                                          (word "noun" "tarte"))))),

R = nil,
G = l1 "N2" (l2 "verb22" (l0 "P2[à]") (l1 "verb1" (l0 "N2"))).
;
no more solutions
```

The principal advantage of this solution is that it based entirely on syntax, even if the double analysis of the coordinated sentence admittedly lacks a certain elegance. However this purely syntactic solution is not completely satisfactory as it accepts sentences which we would like to disallow, as for example sentence (14), in which the adjuncts do not have the same semantic function.

(14) *Pierre boit son café à 10 heures et Luc à Paris.
**Peter drinks his coffee at 10 o'clock and Luke in Paris.*

6. SEMANTIC TRANSFER BASED UPON DEFAULT UNIFICATION

In this version of the prototype, the semantic transfer is implemented by a process which we call *default unification* (`d-unification`) whereby values in the semantic interpretation of the coordinating sentence are used to fill in the gaps in the semantic interpretation of the coordinated sentence. This type of unification is closely related to what is called *credulous default unification* in Carpenter (1993). The process of `d-unification` involves recursively destructuring in parallel the semantic interpretation of the coordinating and coordinated sentences in order to synthesize the result.

The semantic analysis of sentence (6) produces the following result.

```
and (exists y\
       and (event y permission)
           (and (agent y pierre)
                (and (recipient y enfants)
                     (propositional-content y
                         (exists x\
                            and (event x consumption)
                                (and (agent x enfants)
                                     (theme x glace)))))))
    (exists y\
       and (event y permission)
           (and (agent y luc)
                (and (recipient y parents)
                     (propositional-content y
                         (exists x\
                            and (event x consumption)
                                (and (agent x parents)
                                     (theme x tarte)))))))
```

The analysis of the coordinated sentence is constrained by the TEMPLATE which was synthesized during the analysis of the coordinating sentence. The prototype produces the following analysis of the coordinated sentence found in (6).

```
?- p (l1 "N2"
         (l2 "verb22"
             (l0 "P2[à]")
             (l1 "verb1"
                 (l0 "N2"))))
     ("Luc" :: "à" :: "ses" :: "parents" :: "la" :: "tarte"
      :: nil)
     R gapped T S.

S = exists y\
      and (event y E1)
          (and (agent y luc)
               (and (recipient y parents)
                    (propositional-content y
                        (exists x\
                           and (event x E2)
                               (and (agent x parents)
                                    (theme x tarte)))))),
T = t2 "s" (t1 "np" (word "pnoun" "Luc"))
           (t3 "vp" (word "verb" "null")
                    (t2 "pp" (word "prep" "à")
                             (t2 "np" (word "det" "ses")
                                      (word "noun" "parents")))
                    (t2 "vp-inf"
                        (word "prep" "elided")
                        (t2 "vp" (word "verb" "null")
                                 (t2 "np"
                                     (word "det" "la")
                                     (word "noun" "tarte"))))),
R = nil.
```

In the translation S, logical variables E1 and E2 are place holders for the semantic translation of the omitted elements which are labeled +NULL in the analysis tree. The default unification operation assigns to these variables the corresponding elements in the semantic interpretation of the coordinating sentence. In our opinion, this solution strikes a balance between simplicity and adequacy.

ISBN: 0 08 043 438X

7. SEMANTIC TRANSFER BY GENERALIZATION

We show how the prototype can be used to explore a third, more radical, purely semantic solution. In this solution, the semantic transfer from the coordinating to the coordinated sentence is obtained by applying the anti-unifiers (generalizations) of the semantic interpretation of the coordinating sentences to the semantic interpretation of the coordinated sentence. Although the prototype has only been used to explore the use of simple anti-unifiers, this solution is in the spirit of what is proposed by Dalrymple, *et al.* (1991). This solution implements the proposal presented by Sag, *et al.* (1985).

The coordinated sentence of example (6) is analyzed as an iteration of X^2 constituents as follows.

```
?- iter-X2-bar ("Luc" :: "à" :: "ses" :: "parents" :: "la" ::
                 "tarte" :: nil)
                R nil TL nil SL.

SL = is-object tarte :: is-individual parents ::
     is-individual luc :: nil,
TL = t2 "np" (word "det" "la") (word "noun" "tarte") ::
     t2 "pp" (word "prep" "à")
             (t2 "np" (word "det" "ses") (word "noun" "parents")) ::
     t1 "np" (word "pnoun" "Luc") :: nil,
R = nil.
```

The following output show how the semantic interpretation of the coordinating sentence may be generalized upon, given the semantic interpretation of the X^2 fragments in the coordinated sentence.

```
?- genL (exists y\
           and (event y permission)
                (and (agent y pierre)
                     (and (recipient y enfants)
                          (propositional-content y
                               (exists y\
                                  and (event y consumption)
                                       (and (agent y enfants)
                                            (theme y glace)))))))
        (is-object tarte :: is-individual parents ::
         is-individual luc :: nil) R.

R = lambda x\ lambda x1\ lambda x11\ exists y\
      and (event y permission)
          (and (agent y x)
               (and (recipient y x1)
                    (propositional-content y
                        (exists y1\
                           and (event y1 consumption)
                               (and (agent y1 x1)
                                    (theme y1 x11)))))).

;
```

ISBN: 0 08 043 438X

```
R = lambda x\ lambda x1\ lambda x11\ exists y\
      and (event y permission)
          (and (agent y x1)
               (and (recipient y x)
                    (propositional-content y
                        (exists y1\
                           and (event y1 consumption)
                               (and (agent y1 x) (theme y1 x11)))))).
```

Although the generalization process is tightly constrained by the types of the domain elements, two solutions are produced even in this simple case. However, the second solution, which generalizes first on the *recipient* (instead of the *agent*) could easily be ruled out based on the GPSG analysis.

Finally, the generalization of the semantic interpretation of the coordinating sentence is applied to semantic interpretation of the X^2 fragments in the coordinated sentence to produce the full interpretation of the coordinated sentence.

```
?- bindL (lambda x\ lambda x1\ lambda x11\ exists y\
              and (event y permission)
                  (and (agent y x)
                       (and (recipient y x1)
                            (propositional-content y
                                (exists y1\
                                  and (event y1 consumption)
                                      (and (agent y1 x1)
                                           (theme y1 x11)))))))
         (is-individual luc :: is-individual parents ::
         is-object tarte :: nil)
         R.

R = exists y\
      and (event y permission)
         (and (agent y luc)
              (and (recipient y parents)
                   (propositional-content y
                       (exists y1\
                         and (event y1 consumption)
                             (and (agent y1 parents)
                                  (theme y1 tarte)))))).
```

Although purely semantic in principle, this solution must rely on syntax to weed out extraneous generalizations as was illustrated above. Finally, despite the fact that it makes use of very powerful machinery indeed, this solution does not produce qualitatively better results than those obtained by the hybrid solution based on default unification.

8. CONCLUSION

In our GPSG account of gapped coordination, the parallelism between coordinating and coordinated sentences is reflected in the isomorphic syntax analysis trees. This syntactic parallelism is then used to induce a semantic parallelism between coordinated sentences. The λ-PROLOG prototype was used to compare and evaluate different semantic interpretation strategies for gapped coordination. It allowed us to show that a satisfactory analysis of gapped coordination relies both on syntax and semantics. Of the three solutions presented, we retain the second one based on default unification for its adequacy and simplicity. Finally, the prototype allowed us to validate empirically our GPSG analysis.

ISBN: 0 08 043 438X

REFERENCES

Bouchard, L. H., L. Emirkanian and C. Fouqueré (Forthcoming). Les coordinations avec gapping : formalisation en GPSG et HPSG. *Traitement Automatique des Langues.*

Bouchard, L. H., L. Emirkanian and M. Labelle (1996). Traitement de l'ellipse du verbe dans les coordonnées en Grammaire Syntagmatique Généralisée. *Travaux de linguistique*, **32**, 7-32.

Carpenter, B. (1993). Skeptical and Credulous Default Unification with Applications to Templates and Inheritance. In: *Inheritance, Defaults and the Lexicon* (T. Briscoe, A. Copestake and V. de Paiva, (Ed.)), 13-37. Cambridge University Press, Cambridge UK.

Dalrymple, M., S. M. Shieber and F. C. N. Pereira (1991). Ellipsis and Higher-Order Unification. *Linguistics and Philosophy*, **14**, 399-452.

Davidson, D. (1989). *Essays on Actions and Events*. Clarendon Press, Oxford.

Elliot, C. and F. Pfenning (1991). A Semi-functional Implementation of a Higher-Order Logic Programming Language. In: *Topics in Advanced Language Implementation* (P. Lee, (Ed.)), 289-325. The MIT Press, Cambridge MA.

Emirkanian, L., L. Da Sylva and L. H. Bouchard (1996). The Implementation of a Computational Grammar of French Using the Grammar Development Environment. *COLING'96*, Copenhagen, 1024-1027.

Gazdar, G., E. Klein, G. Pullum and I. Sag (1985). *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge MA.

Higginbotham, J. (1989). Elucidation of meaning. *Linguistics and Philosophy*, **12**, 465-517.

Miller, D. (1989). A Logic Programming Language with Lambda-Abstraction, Function Variables, and Simple Unification. In: *Extensions of Logic Programming*, LNCS No. 475, 253-281. Springer-Verlag, New-York.

Miller, D., G. Nadathur, F. Pfenning and A. Scedrov (1991). Uniform Proofs as a Foundation for Logic Programming. *Annals of Pure and Applied Logic*, **51**, 125-157.

Milner, R. (1978). A Theory of Type Polymorphism in Programming. *Journal of Computer and System Sciences*, **17**, 348-375.

Parsons, T. (1990). *Events in the Semantics of English: a Study in Subatomic Semantics*. The MIT Press, Cambridge MA.

Pereira, F. and D. H. D. Warren (1980). Definite Clause Grammars for Language Analysis: A Survey of the Formalism and a Comparison With Augmented Transition Networks. *Artificial Intelligence*, **13**, 231-278.

Russell, G. (1987). *Verbal Ellipsis in English Coordinate Constructions*. Ph.D. Thesis, University of Sussex.

ISBN: 0 08 043 438X

Sag, I. A., G. Gazdar, T. Wasow and S. Weisler (1985). Coordination and How To Distinguish Categories. *Natural Language and Linguistic Theory*, **3**, 117-171.

Troelstra, A. S. and D. van Dalen (1988). *Constructivism in Mathematics: An Introduction.* North-Holland, Amsterdam.

Warren, D. S. (1983). Using λ-calculus to Represent Meaning in Logic Grammars. *21st Annual Meeting of the ACL*, Massachusetts Institute of Technology, 51-56.

ISBN: 0 08 043 438X