CRITTER: a translation system for agricultural market reports

Pierre Isabelle
Marc Dymetman
Elliott Macklovitch

Centre Canadien de Recherches sur
l'Informatisation du Travail
1575 Boul. Chomedey
Laval, Quebec
CANADA H7V 2X2

ABSTRACT

The CRITTER system is being developed to translate agricultural market reports between English and French. It is based on a transfer model, and designed to be reversible. The source and target language texts are described by means of: a) a surface syntactic representation consisting of a tree annotated with feature structures, built by an extraposition grammar; and b) a semantic representation exhibiting predicate argument structures and constrained by type checking, built in parallel with the syntactic structure in compositional fashion. CRITTERS's implementation is still incomplete, but results obtained so far are promising.

TOPIC AREAS: Machine translation, Logic Grammars.

1. Our approach to the translation problem

We are currently developing a translation system with two main objectives in mind: (a) to effectively translate from English to French (and conversely) a real life corpus of texts in a restricted *sublanguage* <Kittredge & Lehrberger, 1982>; and (b) to provide a testbed for a theoretically motivated translation model: insofar as possible, design choices should integrate recent advances in linguistic and semantic theory.

The corpus that we are using for our current experimentation is comprised of weekly reports produced by the Canadian Department of Agriculture, describing the situation of the livestock and meat trade market in the different Canadian provinces. The following excerpts provide a short sample of the language of these reports:

Imports of slaughter cattle from the United States last week dropped 62% compared to the previous week, totalling 334 steers and 50 heifers.

La semaine dernière, les importations de bovins d'abattage ont chuté de 62% en regard de la semaine précédente, totalisant 334 bouvillons et 50 taures.

In terms of its general structure, our translation model may be viewed as being composed of three abstract relations:

(i) the source analysis/synthesis relation:
anasynt_s(T_S, SurfSyn_S, Sem_S)

which defines a set of well-formed triples, where T_S is a source language text, SurfSyn_S and and Sem_S are respectively a surface syntactic structure and a semantic structure for this text, both being source language dependent;

(ii) the target analysis/synthesis relation:
anasynt_t(T_T, SurfSyn_T, Sem_T)

which is the analogue of anasynt_s for the target language; and

(iii) the transfer relation:
tr(Sem_S, Sem_T)

which defines a set of couples, where Sem_S and Sem_T are respectively source and target semantic structures that are considered to be translationally equivalent.

The *anasynt_s* and *anasynt_t* relations are formally and computationally described using the framework of *extraposition grammars* <Pereira 1981>, while the *tr* relation is defined through a set of definite clauses. An important feature of our approach is that each of these *anasynt* relations is in fact reversible (cf the reversibility condition of <Landsbergen 1987>). Practically speaking, this means that a single system is usable for both English to French and French to English translation. From a theoretical viewpoint, reversibility is a strong criterion of linguistic adequacy for a grammar. Typical existing parsers are based on grammars that (sometimes grossly) overgenerate: the grammar writer assumes a high degree of well-formedness in the input text. Conversely, typical existing generators tend to undergenerate: the grammar writer makes arbitrary choices in the paraphrase system of the language. A reversible grammar is of necessity closer to observational adequacy.

2. Representations

The CRITTER system assigns each textual unit a representation that describes both its form (graphological, morphological, syntactic) and its semantic content.

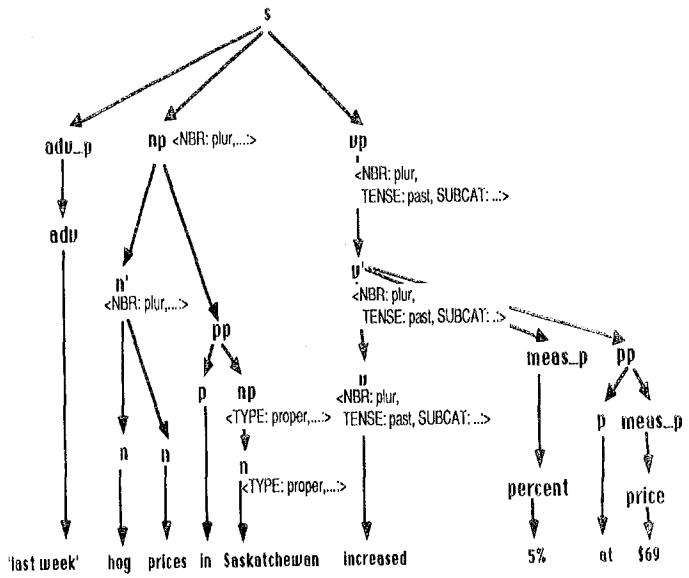
2.1 Syntactic Representations

The syntactic representation associated with a textual object is a fairly standard surface structure tree which may include *traces* in places where a (long-distance) dependency holds between some displaced phrase and a *gap*. Since we adopt a monostratal view of syntax, no other level of syntactic representation is provided for. The representation scheme is based on a variant of the *feature structure* approach (Sag & al., 1986). Each node of the surface tree is represented as a feature structure which includes, among others, *cat* and *daughters* attributes.

Using familiar tree notation, our current grammars would assign sentence (2.a) the representation (2b):

(2.a) Last week, hog prices in Saskatchewan increased 5% at $69.00.

(2.b)



This structure is more or less in line with current syntactic theories. Note that it reflects a three-level X-bar convention. Occasionally, idiosyncratic features are adopted so as to account for the peculiarities of the sublanguage we are dealing with. This is the case for the complements *meas_p* and *pp* under *v'*, which do not correspond to the usual subcategorization pattern for the verb 'increase'.

2.2 Semantic representations

Formally, our semantic representations are trees -- or more exactly, directed acyclic graphs, for structure-sharing is allowed in cases of coreference -- in which nodes are labelled with semantic units that often, but need not, correspond to the lexemes of the language represented. We introduce abstract semantic units to account for some lexical gaps, morphologically marked semantic notions, etc. The arcs are labelled either by *argument numbers* (1), (2), (3), ..., or by *"inverse" argument numbers* (inv-1), (inv-2), ...

The interpretation of this notation is made easier if one considers as an example the semantic structure in Figure (a), which is associated with sentence (2.a): "Last week, hog prices in Saskatchewan increased 5% at $69":

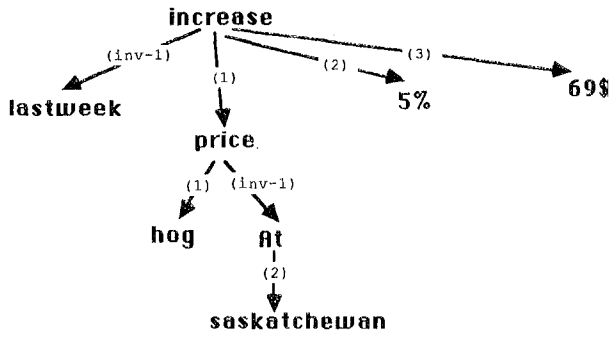


Figure (a)

In this structure:

- 'At', '5%', '69$' are abstract semantic units;
- 'lastweek' is treated as a single unit, which is justified by the fact that it plays the role of a frozen indexical in our sublanguage (as 'yesterday' does in the standard language);
- the (1), (2), (3) labels correspond to argument positions, either of predicates (like 'increase', treated as a 3-place predicate) or of functions (such as 'price', which takes a commodity as first argument);
- the (inv-1) labels correspond to "inverted" argument relations, which implies that 'increase' is in first argument position relative to 'lastweek', and that 'price' is in first argument position relative to 'At', ('Saskatchewan' being the second argument of 'At').

Labels of the "inv" kind are a notational device which makes it possible to simultaneously read two representational levels off a single semantic structure: a first level which expresses predicate-argument relations; and a second level which is reminiscent of the *subordination* of syntactic groups. Thus 'lastweek' is a syntactic dependent of 'increased', and 'in Saskatchewan' is a syntactic dependent of 'prices'. There are two reasons for choosing to reflect subordination in the semantic structure: first, we want to maintain a treelike character for the semantic structure (unique root, no cycles). This is technically related to the fact that transfer crucially depends on a root-to-leaves recursive traversal of semantic structures. Second and much more important is the fact that subordination does have semantical import, although in a way which is not currently very well understood.

Semantic structures have to obey a well-formedness criterion, which consists of the checking of semantic type agreement between a predicate (or functional) node in the structure and its argument nodes. Defining semantic well-formedness involves a *semantic lexicon*, a *semantic type subsumption hierarchy*, and *semantic well-formedness rules*. These are briefly described in sections 3.3 and 4.3 below.

3. The lexicon

CRITTER's lexical component is made up of a basic dictionary of morpho-syntactic lexical units; a rule component which extends the morpho-syntactic dictionary; and a dictionary of semantic-level units.

3.1 The morpho-syntactic dictionary

This dictionary lists lexical items in citation form and assigns them morphological and syntactic properties. It is also responsible for effecting the mapping of these lexical units onto the semantic-level units, whose properties are described in a separate dictionary. Morphological properties include an inflectional class and an indication of any morpho-syntactic idiosyncrasies. Syntactic properties are comprised of a *subcategorization frame* and a collection of syntactic features. The subcategorization frame of a lexical head describes the number and syntactic type of the phrases governed by that head. These frames refer directly to positions in surface structure, since this is the only level of syntactic representation admitted in our system. Verbs, for example, can be marked for a maximum of three positions: a subject and up to two complements.

The mapping onto semantic units is effected by associating each lexical entry with a semantic schema. This schema is made up of a semantic unit (represented as a functor with a fixed arity), and an indication of the relationship of the arguments to the lexical unit's syntactic dependents.

In each lexical entry, this complex of morphological, syntactic and semantic information is specified as a feature structure. This feature structure is encoded as a Prolog term that we describe indirectly, by means of predicates which access the relevant attribute values. For example, the syntactic and semantic properties of the verb 'promise' could be represented as a term T, described as follows:

(3.a)
```
citation_form(T, promise),
subcat(T, [NP1, NP2, VCOMP]),
cat(NP1, np),
sem_form(NP1, A),
cat(NP2, np),
sem_form(NP2, B),
cat(VCOMP, vp),
vform(VCOMP, infinitive),
sem_form(VCOMP, C),
control(VCOMP, NP1),
sem_form(T, promise'(A, B, C))
```

The predicates *citation_form*, *cat*, *subcat* and *sem_form* simply access the value of an attribute of the same name in the term T. The *subcat* attribute has a value of the *list* type, as in <Pollard & Sag (1988)>. Syntactic rules will unify the elements of this list with the complements of the lexical head, thereby enforcing the appropriate subcategorization restrictions (e.g. the "cat" of the second complement of 'promise' is vp).

Taken together the *subcat* and *sem_form* attributes account for an essential part of the syntax/semantics mapping. According to the description given for 'promise', the semantic objects associated with the subject, the object and the infinitival complement are respectively mapped onto the first, second and third argument of the semantic predicate promise'.

All the resources of clausal logic can be invoked to enforce complex relationships between several feature structures. For example, the predicate *control* used in (3.a) above is defined in such a way as to ensure that the *sem_form* and *agree* values of the controller match those of the subject slot in the subcat of the controlled verb phrase:

(3.b)
```
control(CONTROLLEE, CONTROLLER) :-
    subcat(CONTROLLEE, [SUJ, X, Y]),
    sem_form(SUJ, SF),
    sem_form(CONTROLLER, SF),
    agree(SUJ, AG),
    agree(CONTROLLER, AG).
```

The *control* statement of (3.a) will thereby ensure that the subject of 'promise' and the understood subject of its infinitival complement are the same entity.

3.2 Morphological and lexical rules

The morpho-syntactic dictionary is extended by three sets of rules that handle inflection, derivation and lexical transformations. Our description of French inflection is based on the work of Bourbeau & Pinard (1986), which provides an exhaustive specification of the inflectional properties of more than 50,000 French lexical items. We have also developed a parallel description for English inflection.

We currently employ a rule-based treatment of derivational morphology only for the most productive classes, such as comparative and superlative adjectives, '-ly' adverbs, etc. On the other hand, we make extensive use of lexical transformations to handle phenomena such as passivization, subject-to-subject and subject-to-object raising, intransitivation, dative-shift, etc. Given the scheme described above for lexical subcategorization, most lexical transformations can be seen as simply altering the subcategorization pattern of the lexical entry.

For example, given the lexical specification (3.c) for the object-raising verb 'believe', rule (3.d) has the effect of generating the two "virtual" dictionary entries (3.e) and (3.f).

(3.c)
```
dict(T) :-
    citation_form(T, believe),
    suj_to_obj_raising_verb(T,A,B),
    sem_form(T, believe(A,B)).
```

(3.d)
```
subj_to_obj_raising_verb(T,A,B) :-
    % standard form
    subcat(T, [NP, S, [ ]]),
    cat(NP, np),
    sem_form(NP, A),
    cat(S, sbar),
    complementizer(S, that),
    sem_form(S, B).
subj_to_obj_raising_verb(T,A,B) :-
    % raising form
    subcat(T, [NP1, NP2, VCOMP]),
    cat(NP1, np),
    sem_form(NP1, A),
    cat(NP2, np),
    sem_form(VCOMP, B),
    cat(VCOMP, vp),
    form(VCOMP, infinitive),
    control(VCOMP, NP2).
```

(3.e)
```
citation_form(T, believe),
subcat(T, [NP, S, [ ]]),
cat(NP, np),
sem_form(NP, A),
cat(S, sbar),
complementizer(S, that),
sem_form(S, B),
sem_form(T, believe(A,B)).
```

"Tom believes that Bill is dishonest."

(3.f)
```
citation_form(T, believe),
subcat(T, [NP1, NP2, VCOMP]),
cat(NP1, np),
sem_form(NP1, A),
cat(NP2, np),
sem_form(VCOMP, B),
cat(VCOMP, vp),
form(VCOMP, infinitive),
control(VCOMP, NP2),
sem_form(T, believe(A,B)).
```

"Tom believes Bill to be dishonest."

3.3 The semantic lexicon

The semantic lexicon defines a set of semantic units for each language (whether directly realized by a lexeme or more abstract); describes a *subsumption hierarchy* of *semantic types* (a partial order of types <Sowa, 1983>); and associates with each semantic unit SU an initial semantic type, this having the consequence that SU belongs implicitly to all higher types in the hierarchy. The semantic lexicon also defines a set of *validating predicate-argument schemas*, of which valid predicate-argument structures have to be instances. An example of such a schema is:

MOVEMENT(MEASURE-FUNCTION, INCREMENT, MEASURE)

where MOVEMENT, MEASURE-FUNCTION, INCREMENT and MEASURE are semantic types.

The use of the semantic lexicon to test semantic structure well-formedness is briefly explained in section 4.3.

4. The Grammars

4.1 Syntactic Rules

CRITTER's grammars assign textual units a feature structure describing both their syntactic form and semantic content. As an example, consider rule (4.a):

(4.a)

```
vbar(VBAR) -->
    vb(VB),
    complement(CO1),
    complement(CO2),
    {cat(VBAR, vbar),
    subcat(VB, [SUJ, CO1, CO2]),
    head_of(VBAR, VB)}.
```

The constituent *vbar* is expanded as a verb and two possible complements.

Generally speaking, a complement can be any of a wide range of phrases:

(4.b)

```
complement([]) -> [].
complement(NP) -> np(NP).
complement(PP) -> pp(PP).
complement(VCOMP) -> vp(VCOMP).
    .....
```

Most of the syntactic rules that we use are, like (4.a), based on the simple context-free skeleton of definite clause grammars, with the same augmentation mechanisms: non-terminals have arguments and additional PROLOG goals (enclosed in braces) can be stated. The non-terminals in our rules are uniformly assigned a single argument, whose content is a feature structure, and the PROLOG goals are used to state mutual constraints between these feature structures.

In example (4.a), the two complements of the verb are unified with the second and third elements of the *subcat* list of the verb, thereby enforcing its lexical subcategorization requirements. The *head_of* predicate is defined so as to unify the head features of the lexical head with those of the larger verb phrase. Since *sem_form* is a head feature, the lexical value of *sem_form* for the verb will be assigned to the verb phrase. In the process, arguments of the semantic predicate associated with the verb will become instantiated to the semantic objects associated with the complements of the verb.

In order to deal with certain more complex syntactic phenomena, such as unbounded dependencies, we take advantage of the special facilities built into the extraposition grammar formalism.

4.2 Syntactic processing

Although the format of our grammatical rules closely resembles the format of definite clause grammars (DCGs) and extraposition grammars (XGs), there are some important differences.

Because of their direct relationship with clausal logic, DCGs and XGs have two distinct interpretations: on the one hand, a declarative interpretation in which they can be viewed as defining a relation between strings and structural descriptions; and on the other hand, a procedural interpretation in which they may be viewed indifferently as parsers or synthesizers.

However, given the standard compilers for these formalisms, the procedural interpretation of any given set of rules can rarely be used for both analysis and synthesis tasks. For example, any DCG containing left-recursive rules will produce infinite loops when applied to analysis tasks, although the same grammar may well be suitable for synthesis tasks. Moreover, in order to obtain reasonably efficient parsers and synthesizers, it is necessary to control the order in which goals are called in each mode.

Our solution to these problems is to retain the use of DCG-like rules which have a well-defined declarative semantics; however, we enrich these rules with control annotations which, while not affecting their semantics, provide a rule compiler with the information needed to produce both an analysis-oriented and a synthesis-oriented version of the rule. Left-recursion is eliminated in the analysis version, and both versions typically display a different ordering of the goals. The result is that we can actually derive fairly efficient parsers and synthesizers from one and the same grammar.

For further details on this *double-compilation* approach, see Dymetman & Isabelle (1988).

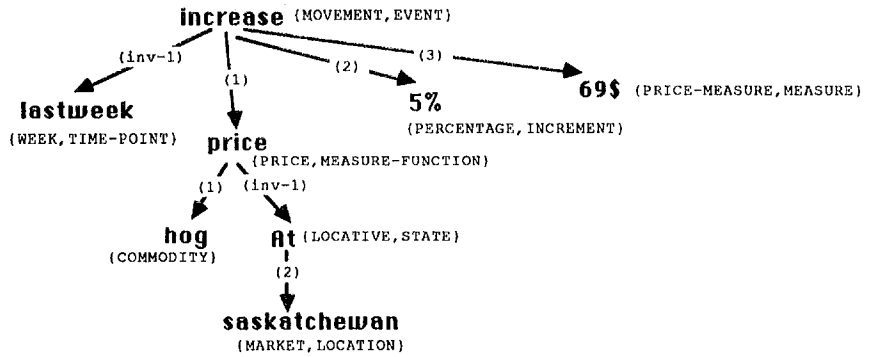
4.3 Checking of semantic well-formedness

In order for a semantic structure built by this compositional process to be accepted as valid, it must pass a *semantic well-formedness* check, which involves *semantic constraints* and the *type subsumption hierarchy*.

This check can be briefly described as follows: for each predicative (or functional) node *pn* in the semantic structure, having *an1*, *an2*... as argument nodes, one tries to find a *validating schema* (see §3.3) *PT(AT1,AT2,....)* such that *PT* is a type subsuming *pn*, *AT1* a type subsuming *an1*, *AT2* a type subsuming *an2*,...

For instance, given the semantic structure of section 2.2, partially annotating it with the types of each node yields (4.c).

(4.c)



The checking of this structure then involves looking for:

- a validating schema for 'lastweek', in this case the schema
  TIME-POINT(EVENT)
- a validating schema for 'increase', in this case the schema
  MOVEMENT(MEASURE-FUNCTION,INCREMENT,MEASURE)
- a validating schema for 'At', in this case the schema
  AT(PRICE,MARKET)

5. Transfer

As we have seen, the transfer component implements a relation between two language-dependent semantic structures. The decision to restrict the input and output of transfer to such semantic structures is motivated by a number of considerations. Pre-theoretically, the very notion of translation implies a linguistic reformulation which preserves essential *meaning*. Such abstract intermediate structures also have the practical advantage of simplifying the transfer component. That is to say, we assume that the analysis component is powerful enough to neutralize certain source language transformations and that a full-fledged synthesis component can take care of such details of target language realization as governed prepositions.

The transfer component itself is essentially lexical, with all relevant knowledge expressed in a *transfer lexicon*, a sample of which appears in (5.a) :

(5.a)

(i) eat <-> manger.

(ii) miss(1: X, 2: Y) <-> manquer(1: Y', 2: X').

(iii) walk(inv-1: across(2: X))
<-> traverser(2: X', inv-1: $manner (2: apied)).

entry (i) is straightforward;
entry (ii) expresses an "argument conversion" : *john misses mary <=> mary manque à john;*
entry (iii) expresses a more complex correspondance: *john walks across the street <=> john traverse la rue à pied*

This lexicon is compiled into a set of Prolog clauses. The transfer algorithm then performs a simultaneous recursive root-to-leaves traversal of source and target semantic structures, making use of these clauses to maintain translational equivalence of the source and target structures. Practically speaking, the result is that when translating from English to French, for example, as the transfer algorithm traverses the English semantic structure, the French semantic structure is constructed in parallel, by progressive instantiations.

In this way, the transfer process may effect certain restructurings, but these are lexically triggered: we do not foresee the need for an independent structural transfer component, as in ARIANE-78 for example <c.f. Boitet & Nedobejkine, 1981>.

6. Conclusion

CRITTER is currently being implemented in QUINTUS Prolog on SUN-3 workstations. At the time of writing, the status of the prototype is as follows:

- morphological descriptions for both English and French are running, including exhaustive descriptions of the inflectional systems; the dictionaries include approximately 500 lexemes in each language, and are expected to go beyond the 1000 mark;

- syntactic descriptions already cover a significant part of the constructions found in the sublanguage, although much work remains to be done to deal adequately with ellipsis, complex coordination, etc.; furthermore, the grammars of English and French are actually used in a reversible manner, although at the time of writing, they tend to overgenerate in the synthesis mode;

- a simple version of type-checking has been implemented, but work remains to be done on defining an adequate hierarchy of semantic types;

- an initial implementation of the transfer component (dictionary and programs) is under way and the first translations should be produced within a few weeks.

7. Acknowledgments

The authors wish to thank Lucie Cusson, Bruno Godbout, François Perreault and Michel Simard, who have made important contributions to the work reported here.

REFERENCES

Boitet, C. & Nedobejkine, N. (1981) "Recent developments in Russian-French machine translation at Grenoble". In: Linguistics 19, 199-271.

Bourbeau, L. & Pinard, F. (1986). Dictionnaire Micro-informatisé du francais. Montreal, Canada.

Dymetman, M. & Isabelle, P. (1988). "Reversible Logic Grammars for Machine translation". In Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages, Carnegie-Mellon University.

Gazdar, G., Klein, E., Pullum, G., Sag I. (1985) Generalized Phrase Structure Grammar. Oxford: Blackwell.

Isabelle, P. & Macklovitch, E. (1986) "Transfer and MT Modularity". In Proceedings of Coling 86, Bonn, FRG.

Kittredge, R. & Lehrberger, J. (1982) eds.: Sublanguage: studies of language in restricted semantic domains. Berlin: de Gruyter.

Landsbergen, J. (1987) Montague Grammar and Machine Translation. Philips Research M.S. 14.026, Eindhoven, The Netherlands.

Pereira, F. (1981) "Extraposition Grammars". In: Computational Linguistics 7.4, 243-256.

Pollard, C., Sag, I. (1988) Information-Based Syntax and Semantics, vol. 1: Fundamentals, CSLI lecture Notes no 13, CSLI, Stanford University.

Sag I., Kaplan R., Karttunen L., Kay M., Pollard C., Shieber S., Zaenen A. (1986) "Unification and Grammatical Theory", Proceedings of the West Coast Conference on Formal Linguistics, Stanford Linguistics Association, Stanford University.

Sowa, J. (1983) Conceptual Structures: Information Processing in Mind and Machine. Reading, Mass: Addison-Wesley.