Named Disjunctions and Lazy Evaluation for Syntactic Ambiguities

Philippe Blache
2LC - CNRS
1361 route des Lucioles, 06560 Sophia Antipolis, France
pb@llaor.unice.fr

Abstract

Ambiguity is one of the main sources of complexity in natural language processing. We propose in this paper an original solution relying on the use of named disjunctions which are set of ordered formulae linked by a name. This representation is implemented within a constraint logic programming paradigm (allowing the use of underspecified structures) by means of lazy evaluation techniques. This approach avoid the expansion of the disjunctions into a normal form and allows in some cases to compute a partial syntactic structure without disambiguating.

1. Introduction

Ambiguity remains one of the main problem in natural language processing. We generally represent it as disjunction (see [Kasper90] and [Döre90] for formal aspects) and different solutions are proposed for its implementation. The classical one consists in developing the normal form and delaying as much as possible the interpretation (see [Karttunen84]). Another approach relies on the use of canonical structures and represents disjunctions as a conjunction of negations (see [Nakazawa88], [Ramsay90], [Dawar90] or [Johnson90]). More recently, [Döre90] proposed the notion of named disjunctions. Unfortunately, these approaches present drawbacks: untractability, inadaptation to the current linguistic formalisms or impossibility of a direct interpretation.

We propose a solution avoiding these drawbacks. It relies on named disjunctions and is implemented by means of active constraints and lazy evaluation within the framework of the constraint logic programming paradigm. Moreover, this solution is dynamic and makes use of lexical and phrase-structure constraints provided by the context.

2. Disambiguation

Our disambiguation approach mainly relies on relations between the features characterizing a linguistic object (i.e. a lexical item, a phrase, etc.). Indeed, the disambiguation of a particular feature generally entails that of the entire structure. We will call these disambiguating features trigger features, the disambiguated one being the target features.

Disambiguating a trigger feature requires external informations provided by the syntactic structure (what syntax call selection relations). In the example (1), the 2 words are ambiguous. But there exist selection relations between Det/N and Pro/V which reduces the combinatorial to two grammatical solutions.

(1) la ferme (the farm/close it)

These relations are expressed by means of a named disjunction (noted by an indexed V). A named disjunction (hereafter ND) binds several disjunctive formulae with a same name. The position of the disjunct within the formula is relevant for its interpretation: if one of the disjunct in a ND is interpreted to true, then all the disjuncts occuring at the same position in the other formulae of the ND must also be true. Representing NDs consists in defining the features bound by a trigger/target relation. This operation names a disjunction and defines an order on the disjuncts so as into a ND, two values with the same rank describe the same sign.

3. Implementation

We describe in this section a system implementing named disjunctions. In this approach, only ambiguous informations are represented in the disjunction (the other features form the common substructure). Moreover, the disjunctions are not developed into a normal form. Ambiguity in maintained until sufficient information allows the disambiguation. This mechanism is implemented as a delayed process.

Ambiguities are represented at the lexical level. Indeed, we know at this level the kind of ambiguity, the relations between the ambiguous features and the consequences this ambiguity has on the rest of the structure.

The predicates described in the example (2) are used in our HPSG parser developed in LIFE (a Prolog-based language integrating several paradigms such as functional, object oriented, constraint and logic programming). For the predicate trigger/3, the variable I is
the rank of the element A in the list L and in target/3, the element X is of rank I in the list L.

\[
\begin{align*}
\text{trigger}(I,A,L) & :- \\
\text{trigger\_followed}(I,A,L). \\
\text{trigger\_followed}(I,J,A,\{X,L\}) & :- \\
\text{trigger\_followed}(I,J,A,L).
\end{align*}
\]

(2)

The basic structures in LIFE are ψ-terms; they are typed feature structures and implement the HPSG signs. In this language, the function "=" selects a feature of a ψ-term; it is used for path representation. The predicate "=" unifies two ψ-terms.

The example (3) describes the lexical entry (represented as a predicate) of the French determiner la. It implements a categorial ambiguity between a pronoun and a determiner. Practically, this predicate consists in defining several feature types: for example, the predicate \( \text{A.synsem.loc.cat.head.maj=S} \) indicates that S is the type of the feature maj.

\[
\begin{align*}
\text{word}(la,A) & :- \\
A\text{-phon}=la, \\
A\text{-synsem.loc.cat.head.maj}=S, \\
A\text{-synsem.loc.cat.head}=Cl, \\
A\text{-synsem.loc.content}=C2, \\
\text{Di=tdet. Di.spec=sn}, \\
\text{D2=tnom}, \\
\text{D3=quant}, \text{D3.det=exist}, \\
\text{D4=nom.obj}, \\
\text{D4.index.gen=masc}, \\
\text{D4.index.num=masc}, \\
\text{residuate}(I,\text{trigger}(I,S,\{det,n\})), \\
\text{residuate}(I,\text{target}(I,Cl,\{D1,D2\})), \\
\text{residuate}(I,\text{target}(I,C2,\{D3,D4\})).
\end{align*}
\]

(3)

A disjunctive formula is represented as a list and a named disjunction is a set of lists of same length. The relation between these lists (i.e. the name of the disjunction) is defined by an integer variable representing the position in the lists (i.e. the different disjunct values). The control of this relation is implemented by the trigger and target predicates. More precisely, the instantiation of a trigger feature comes to select a position in the corresponding list. This position must be propagated to the entire ND for its disambiguation.

Practically, in the example (3), the evaluation of trigger depends on the knowledge of S (the type of the Maj feature) and target depends on I (the rank of the disjunct). So, this evaluation is delayed (using the built-in predicate residuate which execute a predicate if a control term is touched) until those variables become known.

When the trigger feature is disambiguated, the trigger predicate is evaluated and instantiates the rank I of the trigger value in the disjunction. To its turn, target is evaluated and calculates the value of rank I in the corresponding list.

In the example (3), Maj is the trigger feature. Its disambiguation involves a disambiguation of the rest of the structure. This comes to instantiate the corresponding types for the Head and the Content value.

4. Conclusion

The use of named disjunctions for the representation of lexical or phrase-level ambiguities allows the factorisation of common knowledge and the description of relations between features within a linguistic object. Such a high-level mechanism can be implemented without expanding the corresponding disjunctions into a normal form. We propose for that to take into account the contextual information and to rely the disambiguation process on a distinction between trigger and target features. Such properties together with delaying mechanisms (implementing lazy evaluation) and active constraints (allowing the use of underspecified structures, cf [Blache96]) allows an efficient implementation. We have experimented this approach for an HPSG parser implemented in LIFE.

References


