Shared Trees: Representing a Parse Forest with a Single Tree

Philippe Blache
LPL - CNRS
29 Avenue Robert Schuman
F-13621 Aix-en-Provence
pb@lpl.univ-aix.fr

Abstract

We describe in this paper a method allowing to represent several trees with a single one. In a natural language processing perspective, this comes to represent a set of parses (which denotes an ambiguity) with a unique structure. This technique makes use of two notions: *controlled disjunctions* for the encoding of tree nodes and the definition of a *neutral element* for the dominance relation. We obtain a new kind of trees, called *shared trees*, particularly useful for the representation of ambiguous syntactic structures. This paper shows that such an approach is not only a solution for compacting ambiguous structures, but also constitutes the basis of an efficient parsing technique for natural languages.

1 Introduction

The treatment of ambiguity in natural language processing classically relies on mechanisms allowing (*i*) to factorize ambiguous structures and (*ii*) to delay the evaluation of such structures (see for example [Karttunen84], [Dörre90], [Maxwell91], [Chen97] or [Blache98]).

The problem of an efficient treatment of ambiguity is twofold. On the one hand, disambiguating generally requires information coming from all the linguistic levels (lexical, morphological, syntactic and semantic). We need then to maintain during the parse, eventually until the end, ambiguous structures. On the other hand, maintaining ambiguity is generally expensive even using delaying techniques because the evaluation of such structures requires an expansion into a disjunctive normal form. A solution consists in the coexistence of different structures (several trees) represented simultaneously (e.g. shared forests) or implicitly (using a non-deterministic strategy).

We propose in this paper a technique allowing to build a unique ambiguous structure or, in other words, to represent several trees with a single one, called *shared tree*. This kind of tree has two characteristics. First, their nodes contains several values which form a special kind of disjunction called *controlled disjunctions* (a synthesis of different methods such as [Dörre90], [Maxwell91], [Griffith96] and [Blache98]). Controlled disjunctions allow a very compact representation but not powerful enough to take into account complex cases of ambiguity. We overcome this problem by the introduction of a *neutral element* for the dominance relation.

In the first part of this paper, we briefly summarize two factorization techniques: *shared forests* and *underspecification*. We then describe the use of *quasi-trees*, their applications and their limits to represent ambiguity. In a secont part, we present a parsing technique adpated to this kind of ambiguous structure. In the last part (sections 4 to 6) we we present *shared trees* and how they constitute a solution to the problem of compacting, maintaining and treating ambiguity.

2 Towards a Unique Ambiguous Structure

Factorizing an ambiguous structure is useful because instead of maintaining several structures, one have to treat only subparts of smaller ones. However, an approach building up a unique structure can be much more efficient: the parse can carry on until the end even with complex ambiguous structures. This technique comes to a kind of determinization of the parsing process.

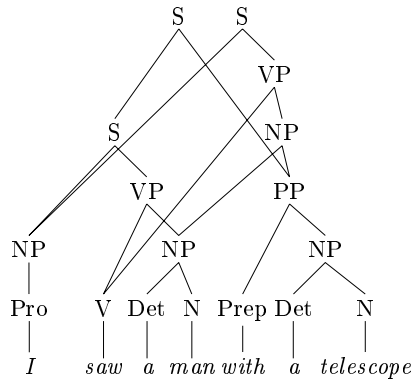
2.1 Shared Parse Forest and Underspecification

One interesting attempt for factorizing information is the *shared parse forest* approach proposed in [Lang91] and used in several works (see for example [Sikkel97]).

The example (1) presents a shared forest of the classical PP-attachment example *"I saw a man with a telescope"*. This forest factorizes the two possible parses of the input string (attachment of the PP to the root S or to the NP object). Such a technique relies on the fact that all the identical subtrees are shared and consequently reduces the space needed for the representation

of a classical set of trees.

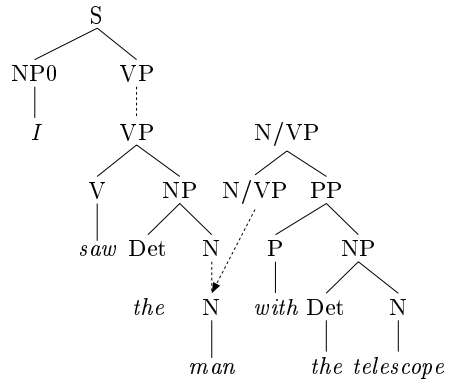
(1)



Shared forests are not only convenient for compacting information, they also constitute an efficient tool for delaying evaluation or representing incomplete information.

Another approach has been proposed by [Chen97] for TAGs. It consists in constructing a unique structure by introducing a new kind of relation. The figure (2)[1] presents this particular dominance relation (indicated by a dashed arrow) between a node and the lowest possible attachment site. This relation simply indicates that there can exist another attachment site between the root and this lowest site.

(2)



This representation is obviously more compact even if less precise (underspecification is at the basis of this approach) than shared forests. The problem here is that such structure requires the use of two kind of dominance relations. Moreover, as pointed out by the authors, the result is not a tree.

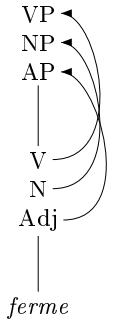
The technique proposed in this paper, *shared trees*, improves significantly a shared forest or an underspecified representation.

2.2 Unary Quasi-Trees

Unary Quasi-Trees (noted UQT), proposed in [Blache98], rely on (*i*) the encoding of dominance relations independently from the dependency ones and (*ii*) a representation of covariation between node values with *controlled disjunctions*.

[1]We don't describe here the notion of quasi-nodes indicated by dashed lines.

(3)



The first characteristic of the UQTs, associated to a lexical entry, concerns the encoding of several level of dominance (the example (3) presents three levels). It is important to notice that dominance is the only kind of information encoded in UQTs.

The second characteristics concerns the representation of ambiguity. At the difference of most of other approaches, we represent ambiguity with a unique structure for the different entries associated to a same lexical item. Each possible categorization corresponds to a node value and nodes represent sets of values (encoded by a disjunction).

A *covariation* relation between the node values is then defined: each value of a node at a level *i* must covary with a value at a level *i+1* and vice-versa. This mechanism is implemented by controlled disjunctions (see [Blache98]) that indicate explicitly the covariation for each value.

The example (3) shows the information associated to the French word *"ferme"* which can be a verb (*"to close"*), a noun (*"farm"*) or an adjective (*"firm"*). The covariation is represented by the arrows between the different values of the nodes.

Other approaches exist that represent dominance relations at the lexical level, for example *supertagging* (see [Srinivas97]). But they don't address the question of ambiguity at the representation level. This is the main difference with UQT[2].

3 Fusion: A Parsing Technique Using UQTs

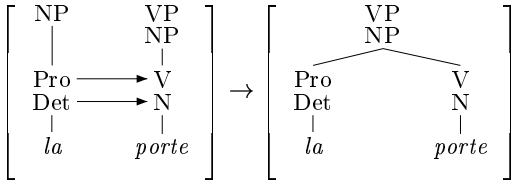
Using rich structures associated to lexical entries simplifies the development of shallow parsing techniques (see for example the *Lightweight Dependency Analyzer* using supertags presented in [Srinivas97]). We propose in this paper a parsing technique, called *fusion*, suited to the use of UQTs. This technique is interesting because the same approach can be used for shallow parsing or fine-grained syntactic analysis.

The fusion mechanism consists in selecting two juxtaposed UQTs and merging them under some conditions (which can be more or less precise according to the granularity level of the parse). The result is a *shared tree* with the fused node as root. The same process is applied recursively until no more trees can be created.

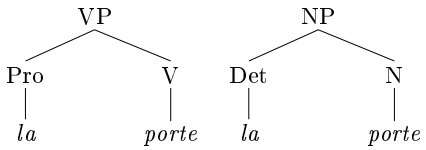
The fusion operation relies on the knowledge of dominance and dependency relations. The minimal fusion condition can be formulated as follows:

Constraint 1 (Fusion Condition) *Two juxtaposed trees can be fused if they are linked by a dependency relation and their root node values are of the same type.*

[2]Another difference with supertagging is that UQTs only represent direct dominance from the bottom to the top of the tree which means one daughter per mother.

(4) 

The figure (4) presents a fusion between two UQTs. All their values belong to a dependency relation (indicated by an arrow). The question is then to find if the roots can be fused. In this example, we can verify that the resulting shared tree can be rooted either by *VP* (with the interpretation *"carry her/it"*) or *NP* (with the interpretation *"the door"*). The covariation (implicit in this figure) between the values allows to generate from this shared tree the two trees of the figure (5).

(5) 

However, the process described above is not totally correct for two reasons : (*i*) the fusion between the root values *NP* and *VP* of the UQTs is impossible and (*ii*) the tree rooted by *VP* in the example (5.1) presents an immediate dominance between *VP* and *Pro* which is also incorrect (*Pro* has to be projected into a *NP*). We need then to find a more adequate representation of the dominance relation in order to tackle these problems.

4 Neutral Element

Let's treat first the question of the *VP* dominating incorrectly *Pro* in the example (5). The problem here comes to factorize two trees with different depths. We propose for this to introduce a *neutral element* for the dominance relation.

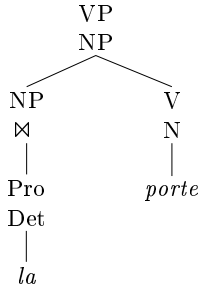
4.1 Definition

The neutral element, noted $\bowtie$, is defined as follows :

Definition 1 (Neutral Element) *Let's note $\alpha \rightsquigarrow \beta$ the dominance relation between two values α and β, let's note $\bowtie$ the neutral element for the relation $\rightsquigarrow$. We have :*

$$\alpha \rightsquigarrow \bowtie \rightsquigarrow \beta \equiv \alpha \rightsquigarrow \beta$$

In terms of graphs, a path containing a neutral element can be contracted by eliminating it.

(6) 

The use of $\bowtie$ allows the representation of two different trees into a same structure as shown in the figure (6). In this example, the value *Pro* covaries with the value *NP* and *Det* with $\bowtie$.

In the case where the interpretation *Det* is chosen, then this value is directly dominated (by contraction) by the root value *NP* whereas the value *Pro* is dominated by *NP* which is to its turn dominated by *VP*. So, a same branch of this tree represents two possible paths with different number of nodes.

4.2 Creating Shared Trees

A *shared tree* is a tree in which (*i*) nodes can be complex (i.e. constituted by a disjunction of values), (*ii*) values are constrained by covariation relations and (*iii*) nodes can contain a neutral element. In this perspective, UQTs are a particular case of shared trees.

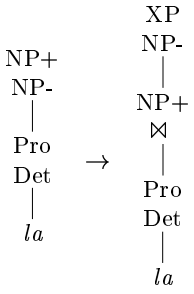
The definition of some explicit conditions is then necessary before describing the creation of shared trees. Let's introduce first the notion of *primary dominance*[3].

Definition 2 (Primary Dominance) *We call primary dominance the relation between a head and its projection.*
A primary dominance is indexed with the sign + (a secondary one with -).

In the example (7), the values *Pro* and *Det* are both projected into a *NP*, the first using a primary dominance and the other a secondary one (noted respectively *NP+* and *NP-*). The distinction between different kinds of dominance relations allows to precise a new constraint:

Constraint 2 (Fusion restriction) *No two primary dominance values can be fused.*

This constraint stipulates that a syntactic unit has a single head[4]: two secondary projections can be fused, but a primary projection can only be fused with a secondary one.

(7) 

In the example (7), the only possibility to fuse a primary dominance (here *NP+*) is a secondary one of the same type. If the primary dominance value is complete (or saturated, in other words no more values depends from it) as in our example (in which *Pro* is the only possible daughter of *NP+*), then this value cannot be fused with any other one. The only solution is to project a new level. At the opposite, a secondary dominance has to be fused with a primary one and it is not necessary to project it. We can generalize this observation by describing the first case of the introduction of a neutral element (called $\bowtie$-projection). This case arises when two node

[3] A dominance relation which is not primary is necessarily secondary.

[4] This constraint can be relaxed for multi-heads constructions.

values only differ by the dominance type. In the following definitions, a node is noted by a tuple $\langle x_1, ..., x_n \rangle$, each x_i representing a value.

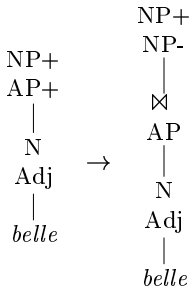
Definition 3 (⋈-projection by dominance type) *If two values $\alpha+$ and $\alpha-$ of the same type belong to a same node, then a new tree can be created as follows :*

- *replace the node $\langle \alpha+, \alpha- \rangle$ by $\langle \alpha+, \bowtie \rangle$*
- *create the new root $\langle \beta, \alpha- \rangle$ (β being the projection of α) dominating the previous node.*

The example (7) describe this first case of ⋈-projection: the saturated *NP+* is projected into the variable value *XP* whereas the secondary projection *NP-* has been replaced by ⋈ and pushed up to the root.

The second case of ⋈-projection occurs when two values α and β of a same node have the following property: α is a primary projection and β has a secondary projection of the same type as α. This example is presented in the figure (8): *NP+* is primary and *AP* can be projected into *NP-*.

(8)



In this case, we can define the ⋈-projection as follows:

Definition 4 (⋈-projection by inclusion) *If two values $\alpha+$ and $\beta+$ (β having a projection of type α) belong to a same node, then a new tree can be created as follows :*

- *replace the node $\langle \alpha+, \beta+ \rangle$ by $\langle \bowtie, \beta+ \rangle$*
- *create the new root $\langle \gamma, \alpha- \rangle$ (γ being the projection of α) dominating the previous node.*

The example (8) illustrates this second case of ⋈-projection: the value *AP* can be a constituent of *NP*. So, *AP* is projected into a *NP-* whereas *NP+* is replaced by ⋈ and pushed to the root.

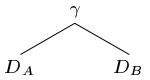
5 Fusion with Shared Trees

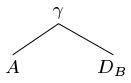
We can now present the fusion mechanism applied to shared trees. This process takes as input two trees and returns a new shared tree. The input is a set of UQTs at the first step of the parsing process. These trees are filtered in the sense that we eliminate the values not connected by dependency relations.

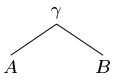
The general mechanism relies on the unifiability of the roots. When the roots of two trees to be fused are not unifiable, a ⋈-projection is applied to one or both of these trees. The fusion is then applied to these new trees.

The fusion can be described more precisely as follows :

- *Let A and B be two filtered trees with α and β their respective roots, D_A and D_B the subtrees of A and B under α and β.*
- *Let insert_ne be a function implementing the ⋈-projection.*
- *Let $\sqcup$ be a unification operation between sets of categories.*
- *Let ST be the shared tree to be constructed.*

1. $\gamma = \alpha \sqcup \beta$
2. **if** $\gamma \neq \emptyset$ **then** ST ← 

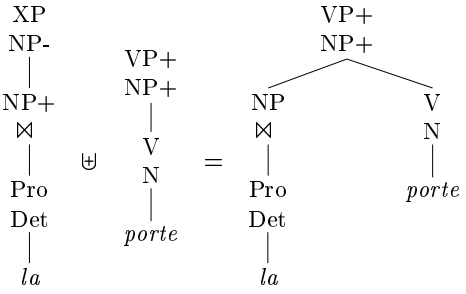
3. **else** α'=insert_ne(α)
4. $\gamma = \alpha' \sqcup \beta$
5. **if** $\gamma \neq \emptyset$ **then** ST ← 

6. **else** β'=insert_ne(β)
7. $\gamma = \alpha' \sqcup \beta'$
8. **if** $\gamma \neq \emptyset$ **then** ST ← 

9. **endif**
10. **endif**

The example of the figure (9) illustrates this fusion process. For clarity's sake, we only present here the trees to be fused and the result. The two roots obtained after insertion of a neutral element in the first tree can be unified: *XP*, which is a variable, unifies with *VP* and *NP-* with *NP+* (according to the constraint 1). The result is a shared tree (dependency relations are not indicated in the example, but are the same as represented in the figure (4)).

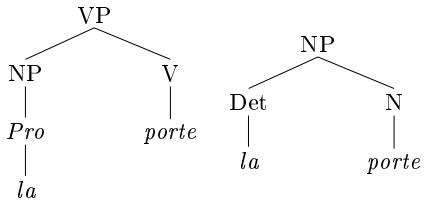
(9)



The covariation relations between the values, the definition of the neutral element and the dependency relations (the relations between *Pro/V* and *Det/N*) allows to make an equivalence between the shared tree in (9)

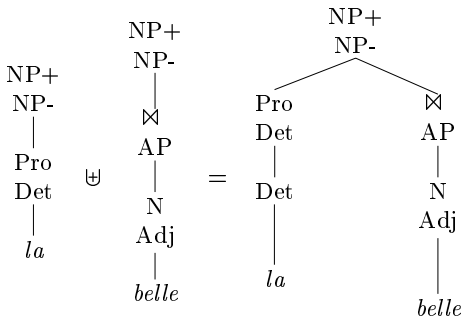
and the trees in (10).

(10)



Let's take another example. The figure (11) is similar to the previous one: the fusion requires the insertion of a neutral element (in the second tree), the resulting shared tree corresponds to three regular trees corresponding to the three valid categorizations (*Pro/Adj, Det/N, Det/Adj*).

(11)



However, this shared tree presents a problem. Indeed, the covariation between the root and its two immediate daughters entails a covariation between *Pro* and *N* which is impossible. More precisely, if we keep the original mechanism of covariation, then the propagation of this constraint must be bi-directional (from daughter to mother and vice-versa). In this case, if the categorization *Pro* in the leftmost branch is chosen, then this value covaries with *NP+* which to its turn covaries with the value *N* of its right daughter. This result is inconsistent with the constraint which forbids two primary projections in a same unit.

We need a supplementary constraint restricting the scope of covariancy within shared trees. This constraint applies to a particular kind of nodes described as follows:

Definition 5 (Exclusion node) *An exclusion node is a node containing a value being a primary projection of two of its daughters.*

As shown in the example, The covariation can only be applied to this kind of nodes. This is expressed by the constraint:

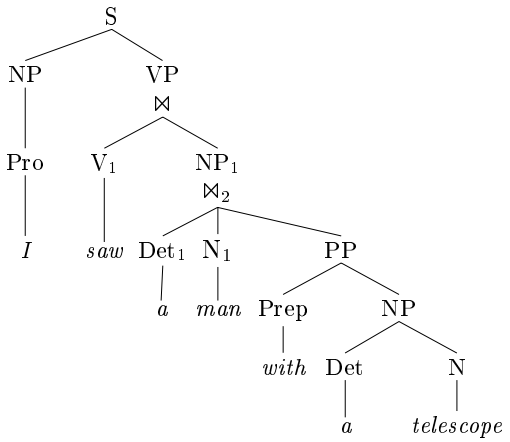
Constraint 3 (Covariancy restriction) *Covariancy cannot be propagated down to an exclusive node.*

6 Examples

The figure (12) shows the representation of the same example as (1) using *shared trees*. We can easily verify that the shared tree (12) is equivalent to the shared forest (1) in that they both represent the two possible trees. Let's describe more precisely this structure.

The covariation[5] between the two nodes containing a neutral element allows to attach the values *Det* and *N* to a *NP* root dominated by the *VP* (i.e. in an object position). But what is interesting is the attachment of the *PP*. Indeed, this value can covary with $\bowtie_2$ or the NP_1 value. In the first case, the node *PP* will be directly attached to the root *S*. In the second case, *PP* covaries with NP_1 which covaries with *VP*.

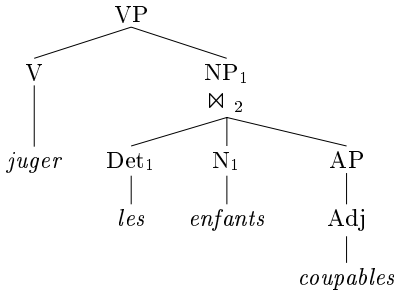
(12)



The comparison between the two representations is direct: they have the same expressivity power, but shared trees are obviously more compact than shared forests. The first advantage of shared trees is then a lowering of space complexity. We describe in the next section other interests of shared trees.

The figure (13) shows another example of rattachement ambiguity for an adjective phrase. The mechanism is the same as with the previous example: the AP can be attached to the *NP* node or the *VP* one.

(13)



The interpretations of this example are (1) *to judge guilty children* or (2) *to consider children guilty.*

7 Discussion

Representing ambiguity using a unique structure presents several advantages. We list in this section some properties that seem to us important.

Parsing: The possibility to create and compute over a unique structure determinises parsing. In the fusion mechanism, each step of the parsing process consists in completing a set of shared trees, without any expansion

[5] An index stipulates the position of the covariating value of the dominating node.

of these ambiguous structures into an equivalent disjunctive normal form. This result is important because it allows the maximum delay for the disambiguation process which is apply only if an actual disambiguating information arises. In this strategy, no "blind" disambiguation is used, at the difference of most of the other approaches, in particular that relying on probabilities.

Maintaining ambiguity: The other important result is that shared trees allow to maintain the ambiguity of a given sentence. This is of deep interest in particular for machine translation (see for example [Wedekind97]) in which ambiguity often needs to be translated as such. Shared trees are not only a way to compact information, it is also a way to represent explicitly ambiguity.

Control: Shared trees allow an efficient control of the evaluation in several ways. First of all, the search space is considerably reduced: a unique tree replaces a forest. Moreover, the research of a model within this space is highly controlled by the covariation process. Concretely, covariation reduces again the search space to different subsets of consistent values.

Decision trees: It seems interesting to draw a parallel with semantic evaluation techniques used for boolean constraint solvers. In particular, binary decision diagrams (see [Bryant86]) allow a compact representation of disjunctive formulae. BDDs are a semantic representation of such formulae whereas shared trees are a syntactic one. In both cases, there is a representation of the dependencies between the values (covariation for shared trees and if-then-else formulae with BDDs). It should be interesting to apply some techniques used in this kind of semantic evaluation in our parsing process.

8 Conclusion

The shared tree technique proposed in this paper allows an efficient representation of ambiguous structures. The efficiency comes from several properties: space complexity reduction, control mechanisms between values, maintenance of the ambiguity. Moreover, shared trees can be used whatever the linguistic formalism. We presented in this paper a general technique allowing to generate automatically such trees.

A parser implementing shared trees has been developped in Java. What is interesting in this approach is that it allows to reuse the same system for different granularity parses. In other words, this parsing system can be used for shallow parsing purposes (the current level of development already allows text chunking) or at the opposite for fine-grained analysis. This seems to us an important result in the sense that such an approach allows the development of shallow parsing techniques taking into account the ambiguity, so with several applications such as information retrieval or speech synthesis.

References

[Blache98] Philippe Blache. 1998. "Parsing Ambiguous Structures using Controlled Disjunctions and Unary Quasi-Trees." in proceedings of *ACL-COLING'98*.

[Bryant86] R. Bryant. 1986. "Graph-Based Algorithms for Boolean Function Manipulation." in *IEEE Transactions on Computers*, 35:8.

[Chen97] John Chen & K. Vijay-Shanker. 1997. "Towards a Reduces Commitment, D-Theory Style TAG Parser." in proceedings of the *International Workshop on Parsing Technologies*.

[Dörre90] Jochen Dörre & Andreas Eisele. 1990. "Feature Logic with Disjunctive Unification" in proceedings of *COLING'90*.

[Griffith96] John Griffith. 1996. "Modularizing Contexted Constraints." In *Proceedings of COLING'96*.

[Karttunen84] Lauri Karttunen. 1984. "Features and Values" in proceedings of *COLING'84*.

[Lang91] Bernard Lang. 1991. "Towards a Uniform Formal Framework for Parsing." In M. Tomita (ed.), *Current Issues in Parsing Technology*. Kluwer Academic Publishers.

[Maxwell91] John T. Maxwell III & Ronald M. Kaplan. 1991. "A Method for Disjunctive Constraints Satisfaction." In M. Tomita (ed.), *Current Issues in Parsing Technology*. Kluwer Academic Publishers.

[Sikkel97] Klaas Sikkel. 1997 *Parsing Schemata* Springer.

[Srinivas97] Bangalore Srinivas. 1997 "Performance Evaluation of Supertagging for Partial Parsing" in proceedings of *IWPT'97*.

[Wedekind97] Jürgen Wedekind & Ronald Kaplan. 1997 "Ambiguity-Preserving Generation with LFG- and PATR-style Grammars" in *Computational Linguistics*, 22:4.