Syntactic Theory course

Assignment 1: **Dependency Grammars**

1 Adjacency principle and structure-sharing

1.1 What did Peter say?

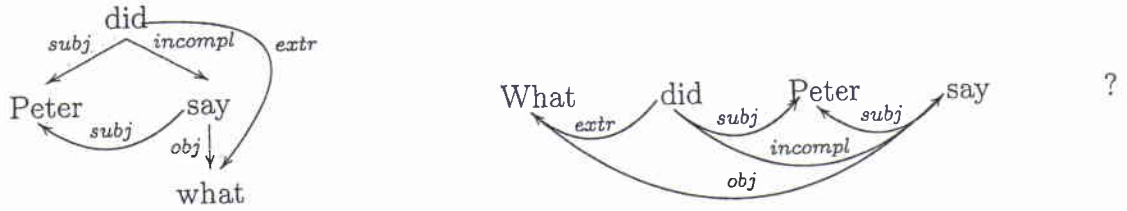

Figure 1: ''What did Peter say ?''

Most of the dependencies here are straightforward, with do structure sharing its subject (Peter) with the subject of say, its incomplement.

$$\texttt{do} \xrightarrow{incompl} \texttt{say}$$
$$\texttt{do} \xrightarrow{subj} \texttt{Peter}$$
$$\texttt{say} \xrightarrow{subj} \texttt{Peter}$$
$$\texttt{say} \xrightarrow{obj} \texttt{what}$$

However, the fourth dependency is a long-distance object dependency and threatens continuity, given the position of what relative to say. For what to be adjacent to say, we require an additional dependency, such that do governs what. Namely:

$$\texttt{do} \xrightarrow{extr} \texttt{what}$$

which involves a local order dependency (*"extractee"*, from [Hudson 00]) that justifies the position of what before do.

How does this dependency support continuity? According to the "revised and final" Adjacency Principle outlined by [Hudson 90], say is adjacent to what if and only if:

- Peter is subordinate to say (true)
- do is subordinate to say (false)

or

- do is subordinate to a mutual head of what and say (true with the extractee dependency) (NB: This test isn't needed for Peter, since we have already shown Peter is subordinate to say)

Therefore, through structure sharing and a local order dependency, we have demonstrated continuity.

1.2 What does Mary think Peter said?

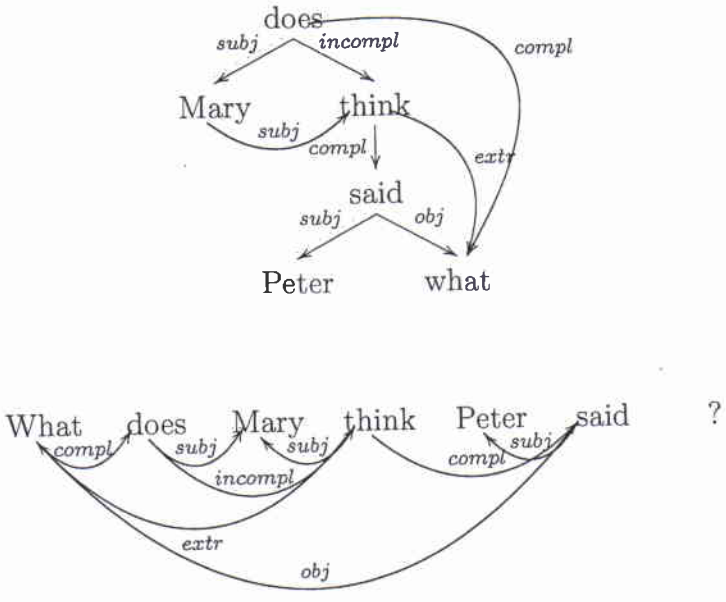

Figure 2: ''What does Mary think Peter said ?''

Again, let's list the more straightforward dependencies, with do structure-sharing its subject Mary with its incomplement think.

$$\texttt{do} \xrightarrow{incompl} \texttt{think}$$
$$\texttt{do} \xrightarrow{subj} \texttt{Mary}$$
$$\texttt{think} \xrightarrow{subj} \texttt{Mary}$$
$$\texttt{think} \xrightarrow{compl} \texttt{say}$$
$$\texttt{say} \xrightarrow{subj} \texttt{Peter}$$
$$\texttt{say} \xrightarrow{obj} \texttt{what}$$

As we did with 1, we can use the Adjacency Principle to demonstrate whether say is adjacent to what. This is true if and only if:

- Peter is subordinate to say (true)
- think, Mary, do are subordinate to say (false - at this point)

or

- {think, Mary, do} are subordinate to a mutual head of what and say (false - at this point)

Since adjacency is not yet satisfied, we have discontinuity. However, if we introduce an extractee dependency from think to what, namely:

$$\texttt{think} \xrightarrow{extr} \texttt{what}$$

then think becomes a mutual head of what and say and, by Hudson's definition, is surbordinate to itself; therefore think is subordinate to a mutual head of what and say. Since think governs Mary (its subject) Mary is also subordinate to this mutual head. However, do is not, so we introduce another dependency:

$$\texttt{what} \xrightarrow{compl} \texttt{do}$$

where do depends on what. Ās a result, do is subordinate to say (since what is subordinate to say through the long-distance dependency $\xrightarrow{obj}$. [Hudson 00] calls this a complement relation, although his rationale for this label is not clear to me. Note that my assumption here is that although dependencies are antitransitive, subordinacy is transitive (i.e., C is subordinate to Ā if C is subordinate to B and B is subordinate to Ā), otherwise subordinacy would be equivalent to dependency.

2 DG Analyses and PS-tree translation algorithm

We propose here two different type of DG analysis, one inspired by MTT [Mel'čuk 98] and one by WG [Hudson 90]. Ās we will see, they offer quite contrasted perspectives.

2.1 Dependency trees

2.1.1 " He kept talking "

This sentence is an example of raising phenomena. The MTT account of such structure, as given by [Mel'čuk 98, p. 71] is quite straightforward: the syntactic structure is made of two dependencies, one subject and one object. For Mel'čuk, there is indeed a dependency between He and talking, but it is a semantic, not a syntactic one.

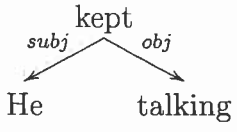

Figure 3: "MTT-like" dependency tree of the sentence

Hudson, on the contrary, explicitly represents a syntactic dependency between the two words, using the structure-sharing rule.

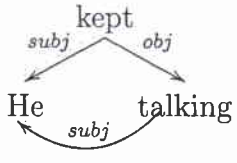

Figure 4: "WG-like" dependency tree of the sentence

2.1.2 " Peter and Mary bought a car "

The treatment of coordination in MTT is explained in [Mel'čuk 98, p. 72 and p. 80]. He argues that coordination phenomenas can be seamlessly treated with "pure" dependencies, without using any other artifact. It thus uses two dependencies, one named coord between the first conjunct and the conjunction, and the other one name conjunct between the conjunction and the second conjunct.

For Hudson, a coordination is a word string, held together by principles other than dependency [Hudson 00]. Is is represented by an "horizontal" set of conjuncts. The conjuncts of a coordination must share the same dependencies to words outside the coordination ("Dependency in coordination Principle").

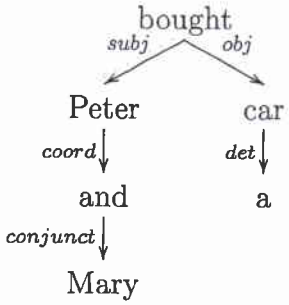

Figure 5: "MTT-like" dependency tree of the sentence

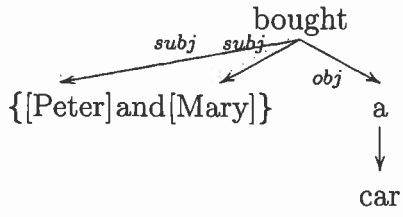

Figure 6: "MTT-like" dependency tree of the sentence

It's also worth noting that, in Mel'čuk's view, the determiner is a dependent of its noun, whereas the opposite holds for Hudson. test: α

2.2 Algorithm converting DG tree relations to phrase structure rules

Before presenting our algorithm, we first have to define what is mathematically meant by the notions of *dependency* and *phrase-structure* trees[1]. Let us recall that a tree can be defined by 2 different ways:

1. As an oriented graph;
2. As a binary relation $\lhd$, called a **tree relation.** We say that $x \lhd y$ iff (y, x) is a link in the corresponding graph, with x and y being 2 distinct nodes.

Let X be an arbitrary set of lexical units, we have:

Definition 1. A **phrase-structure tree** on X is a four-tuple $(X, \mathfrak{B}, \phi, \lhd)$, where $\mathfrak{B}$ is a set of constituents, $\lhd$ a tree relation defined on $\mathfrak{B}$, and ϕ a function (describing the "content" of the constituents) fron $\mathfrak{B}$ to the non-empty subsets of X, so that the three following conditions are satisfied:

1. $\forall \alpha \in \mathfrak{B}$, $\phi(\alpha)$ is a continuous sub-chain of X ;
2. Every subset of X containing only one element is the content of one and only one terminal node ;
3. If $\alpha \lhd \beta$, then $\phi(\alpha) \subseteq \phi(\beta)$.

Definition 2. A **dependency tree** on X is simply a plain tree on X, defined by the couple $(X, \lhd)$

[1]The definition and a sketch of our algorithm is outlined in [Kahane 97]

Of course, the tree relations $\lhd$ don't have the same signification in the two formalisms:

1. For phrase-structure trees, $\alpha \lhd \beta$ means that the constituent α (for example `NP`) is included in the constituent β (for ex. `S`) ;
2. For dependency trees, $\alpha \lhd \beta$ means that the word α depends on the word β - for ex. `car` depends on `bought`.

The algorithm allowing us to translate a dependency tree into a phrase-structure tree is detailed on page 6.

We give below an example of a translation generated by our algorithm:

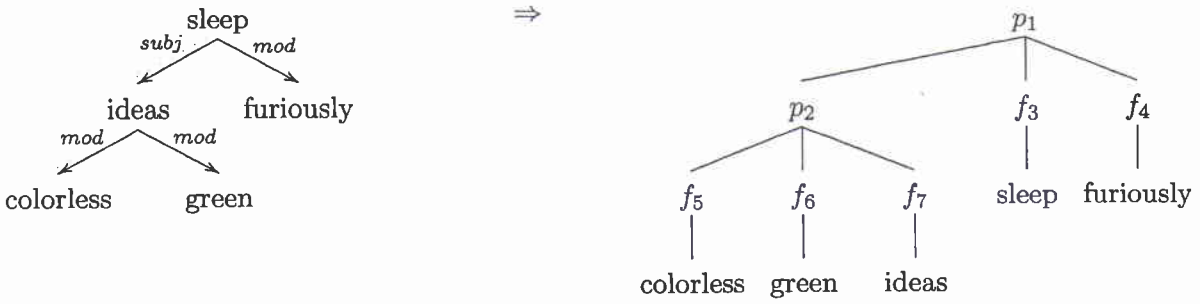

Figure 7: Example of a translation of DG into PS tree

Now, if we rename our constituents by their "usual" names (ie `NP`, `VP`, etc.) and refactor our rewriting rules so as to use only binary ones (by adding intermediate levels), we end up with a structure like this :

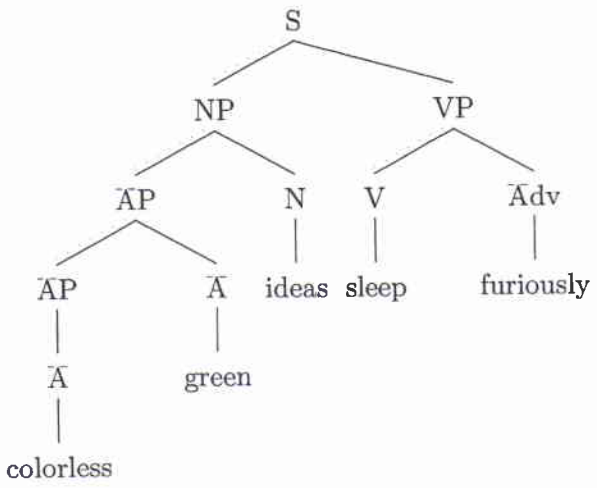

Figure 8: modified PS tree of the figure 7 (renaming of constituents and use of binary rules)

... which is exactly the kind a phrase-structure tree we are used to.

Algorithm 1 Translation of a D-tree into a PS-tree

Require: A dependency tree $(X, \lhd_1)$

% *Initialization of variables*
$\mathfrak{B} = \emptyset$ % *Set of constituents*
$c_\phi = \emptyset$ % *Set of couples describing the function* ϕ
$c_{\lhd_2} = \emptyset$ % *Set of couples describing the relation* $\lhd_2$

for every node x of the D-tree **do**
% *STEP 1: We create two constituents, arbitrarily named* f_x *and* p_x*, for each node of the D-tree*
Add a new constituent f_x to $\mathfrak{B}$
Add a new constituent p_x to $\mathfrak{B}$

% *STEP 2: We describe the contents of the two constituents by specifying the value of their* ϕ *function*

% *1) We define* $\phi(f_x) = \{x\}$
Add a new couple $(f_x, \{x\})$ to c_ϕ

% *2) We define* $\phi(p_x) =$ *projection of* x *(but we first need to calculate it)*
$proj_x = \emptyset$
for every node $y \neq x$ of the D-tree **do**
if $y \preceq x$ **then**
Add y to pro_x
end if
end for
Add a new couple $(p_x, proj_x)$ to c_ϕ

% *STEP 3: We describe the tree relation* $\lhd_2$ *between all the constituents, i.e we build the tree*

% *1) We have* $f_x \lhd_2 p_x$ *(a node is always included in his projection)*
Add a new couple (f_x, p_x) to $c_{\lhd_2}$

% *2) We also have the property* $y \lhd_1 x \Rightarrow p_y \lhd_2 p_x$
for every node $y \neq x$ of the D-tree **do**
if $y \lhd_1 x$ **then**
Add a new couple (p_y, p_x) to $c_{\lhd_2}$
end if
end for
end for

return the phrase-structure tree $(X, \mathfrak{B}, \phi, \lhd_2)$

2.3 Phrase-structure trees

We now present the two PS-trees generated by our algorithm. We only use the MTT trees (the WG trees are much more complex and "irregular").

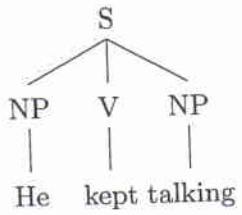

Figure 9: PS tree generated for the first DG-tree

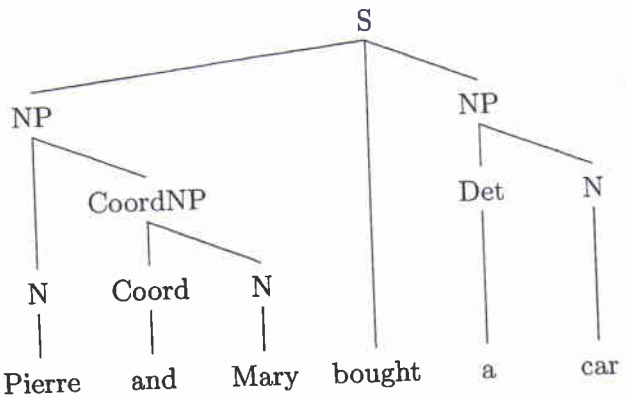

Figure 10: PS tree generated for the second DG-tree

3 Collins's Statistical Parser

3.1 Dependency Model

3.1.1 What Collins wanted

Before going into any detail regarding the dependency model described in Collins' paper, it seems appropriate to comment on the motivation behind his Bigram Lexical Dependency concept. Unlike, say, Hudson's work on Dependency Grammars, Collins never intended to propose a full-fledged system for describing dependency relations which is powerful enough to function as a stand-alone grammar. Instead, Collins' objective was to improve performance in traditional phrase-structure parsing by calculating probabilities phrase-structure-trees based on the probabilities of head-word dependencies.

In short, dependency relation model in the Collins paper are a means to an end, and as such the only requirement for it is to help achieve this end. This should be kept in mind when comparing it to dependency models with more ambitious aims (such as providing a proper grammar).

3.1.2 Collins' Dependency Model

Collins' only considers words in reduced sentences. This means that, for any non-recursive ("base") NP, only the syntactic head is considered, non-head words (such as determiners, adjectives etc.) of non-recursive NPs do not even enter the dependency model.

For the words that are considered in the model, it is their role in syntactic constituents that is modelled. For Collins, every constituent (of a phrase structure tree) which has n child constituents has one head-child and n-1 dependency relations of non-head-children with this head. Head-words propagate up through the constituent system, that is, a constituent's head word is the head word of it's head-child constituent.

The dependencies Collins models are between the head-child of a constituent and the non-head-children of the same constituent. More specifically, Collins defines a dependency as the relation between a constituent's head word H and the head C of one of the constituent's other children, with a relationship R. This relationship R is also specified fully in terms of the syntactic constituent structure, namely in terms of the non-terminal symbol of the head-child H, the non-terminal symbol of the constituent of which both H and C are children of, and the non-terminal symbol of the child constituent C. This is best illustrated with an example:

In the sentence `"John loves Mary"`, the VP with the head word `"loves"` is the head-child-constituent of the top-level S-constituent, which also has an NP with head word `"John"` as its child constituent. In Collins' terms, we have

$$\texttt{loves} \xrightarrow{R} \texttt{John}$$

i.e. `"John"` is a dependent of "loves" where the relation R between them is specified as $R =< \mathrm{NP}, \mathrm{S}, \mathrm{VP} >$ which specifies that the head of an NP (`"John"`) is a dependent of the head of a VP (`"loves"`) in a situation where both are child constituents of S.

It is for dependency relations of this type that Collins calculates probabilities by counting their occurrences in the training corpus.

3.1.3 Comparison to other Dependency Models

Differences between Collins' model and the Dependency Grammar approaches discussed in class occur on various levels. For one, Collins is only modelling a subset of the dependency relations considered in Dependency Grammar approaches. Dependencies between constituents of non-recursive NPs (e.g. between the head noun and its modifiers) are left out.

More fundamentally, the level of description for the relation between a head and its dependent is completely different. Where Dependency Grammars describe this relation in semantically relevant terms (describing, for example, that "John" fulfils a subject role for `"loves"` in the example above), Collins simply describes the structural path from dependent to head in a phrase constituent structure. Knowing the syntactic rules of a specific language, it is sometimes possible to infer functional information from Collins' relations - in the example above, the $R =< \mathrm{NP}, \mathrm{S}, \mathrm{V} >$ relation in the example above indicates, in English, that `"John"` is in a subject role.

But this is diametrically opposed to the Dependency Grammar philosophy of representing functional, semantically relevant relations and making the specific syntactic structures the part that has to be inferred (as in 2.2).

The lack of explicit representation of roles in Collins' representation ties in with another difference, the absence of a valency concept in his model. Many Dependency Grammar models (for example the one used for the Prague Dependency Treebank (PDT) have a clear notion of "slots" for heads, that is, they define a frame describing the different dependency relations that are expected for a specific head (e.g. that `"loves"` has a subject and an object). There is no comparable notion in Collins' paper, where the dependencies of one head word with different dependants are treated as unrelated bigram dependencies.

3.1.4 What's better, then?

From the discussion above, it is clear that a Dependency Grammar model along the lines described by Hudson, or used in the PDT, is much more powerful than Collins' model when it comes to describing language fully in terms of dependencies. For example, the concept of valency in the PDT-VALLEX framework [Hajič 03] allows us to detect cases where dependents are not expressed on the surface level.

But all this is outside the scope of what Collins wanted to do. For evaluating the relative plausibility of competing phrase structure trees, it is not necessary. Collins' structure-bound dependency relations have the advantage that they can easily be converted from and to phrase structures, and the lexical information contained in the bigram dependencies seems to be specific enough to make Collins' parser perform as well as the best known parsers at the time of publication.

3.2 Mapping from PS trees to dependencies

It is not problematic to map from PS trees to the dependencies used in Collins' paper. As Collins describes, the decision which child-constituent is the head-child of a constituent (and which word is the head word of a non-recursive NP) is decided using "a simple set of rules". In Collins' model, being able to identify the head constituents of phrase constituents in the PS tree is all that is required to find heads and dependants for all dependency relations, and the relations themselves are fully specified by the structural relationship of head and dependant, as outlined above. This means that all aspects of the dependencies are specified by the PS structure, without any need to make use of additional information such as valency etc.

For "true" dependency relations, it is a different matter. More detailed, semantic-related descriptions of dependencies, such as the functors used in the PDT-Vallex are not specified by PS structures and require understanding of a sentence on a semantic as well as syntactic level. On the other hand, mapping from PS trees to dependencies can also be problematic in cases where the dependency formalism is not adequate to describe a certain syntactic phenomenon fully - the coordination detailed in figure 6 is an example for this.

References

[Collins 96] M. Collins. *A New Statistical Parser Based on Bigram Lexical Dependencies.* In Proc. of the 34th Annual Meeting of the ACL, 1996.

[Hajič 03] Jan Hajič, Jarmila Panevová, Zdeňka Urešová, Alevtina Bémová, Veronika Kolářová-Řezníčková' & Petr Pajas. *PDT-VALLEX: Creating a Large-coverage Valency Lexicon for Treebank Annotation.* In J. Nivre & E. Hinrichs, editeurs, Proceedings of The Second Workshop on Treebanks and Linguistic Theories, pages 57–68, Vaxjo, Sweden, 2003. Vaxjo University Press.

[Hudson 90] Richard A. Hudson. English word grammar. Blackwell, Oxford, 1990.

[Hudson 00] Richard A. Hudson. Dependency grammar. Course notes from ESSLLI2000, Birmingham, 2000.

[Kahane 97] S. Kahane. *Bubble trees and syntactic representations.* In Proceedings of the 5th Meeting of the Mathematics of the Language, 1997.

[Mel'čuk 98] Igor Mel'čuk. *Dependency in Linguistic Description.* 1998. (unpublished), URL: "www.olst.umontreal.ca/FrEng/Dependency.pdf".