Outline

1. Short overview of the last lecture

2. Phrase Structure Grammars and Dependencies
1. Short overview of the last lecture

2. Phrase Structure Grammars and Dependencies
Overview of lecture on Dependency Grammars

- Dependencies and Phrase Structures:
  - basic objectives of syntactic analysis
  - properties of phrase structure grammars

- Basic definitions of Dependencies
  - What are dependencies?
  - Example analyses

- Differences and Relations between Dependencies and Phrase Structures

- Syntactic Theory/CL and Dependencies
  - Meaning to Text Theory
  - Prague Dependency Treebank
Dependencies so far...

- Dependency analyses aim at revealing the syntactic relations between words in the sentence.

- Clear distinction between the syntactic structure of an expression and the means to express this structure:
  - → phrase structure and linear order are means to express a syntactic structure, and can therefore not be part of the syntactic structure itself.

- When A → B, there is a dependency relation between A and B, where A governs B or B depends on A.

- A dependency relation is:
  - Antisymmetric (if A → B, then B ↛ A)
  - Antireflexive (if A → B, then B ≠ A)
  - Antitransitive (if A → B and B → C, then A ↛ C)
  - Labeled: for each dependency relation, it must be specified what kind of syntactic relation it is.
Dependency trees

- A dependency tree is a connected directed labeled graph, which has exactly one root node that does not depend on any other node
  - The nodes are labeled with reduced word forms
  - The branches are labeled with names of syntactic relations
  - In many versions of dependency grammar, a node may not be governed by two or more nodes
- There are three steps to be taken to create a dependency tree:
  1. Determine which items stand in a dependency relation
  2. Determine the direction of the dependency relation (A → B, or B → A)
  3. Determine what the syntactic relation between A and B is
- In most cases, it is easiest to start with identifying the root of the tree
The following guidelines may help to identify the head of a dependency:

- An item always governs its arguments (i.e., the items it subcategorizes for)
- A head may determine concord with another element
- The head carries the inflection that is relevant for the phrase belongs to a category that has the same distribution as the head + dependent
- The head is obligatory
- The head + dependent is a hyponym of the head
Theories differ in the kind of dependencies that they distinguish and the labels they use for specific relations.

A fundamental distinction (found in (almost) all approaches) is the difference between \textbf{arguments} and \textbf{adjuncts}.

- A head subcategorizes for its arguments. They are often (but not always) obligatory.
- An adjunct is an optional element that modifies the head. They are always optional.
- If a dependent is obligatorily present, it is always an argument.
In this lecture, we will use the following dependency relations (adapted from Kahane (2003) and Hudson (2007)).

- **subj(ect):** subject-of
  
  e.g.  
  \[ \text{subj} \quad \text{subj} \quad \text{John ran, John loves Mary} \]

- **obj(ect):** object-of
  
  e.g.  
  \[ \text{obj} \quad \text{obj} \quad \text{John loves Mary, John gave the book to Mary} \]
Dependency relations (3/4)

- **obj(ect)2**: secondary-object-of
  - e.g. John gave Mary the book

- **prep(ositional)**: prepositional-complement-of
  - e.g. John gave the book to Mary, it depends on you

- **comp(lement)**: complement-of
  - e.g. strawberries with sugar, I saw that John ran
Dependency relations (4/4)

- **det(erator):** determiner-of
  
  e.g. the dog

- **ad(junct):** adjunct-of
  
  e.g. strawberries with sugar, smart students, Mary walked home quickly

**NOTE:**

- Dependencies towards prepositions are labeled with *prep* if they are selected for by the verb (i.e. are arguments), but labeled with *ad* if they are adjunct.
- Relations such as ’obj’, ’obj2’, ’prep’ can be seen as “subtypes” of ’comp’
Exercise

Consider the following sentence:

(1) I knew that he knew from the beginning

Provide the Phrase Structure trees and the Dependency trees for this sentence

Make sure the differences between your trees reveal the ambiguity of the sentence
Outline

1. Short overview of the last lecture
2. Phrase Structure Grammars and Dependencies
Short overview of the last lecture
Phrase Structure Grammars and Dependencies

Syntactic relations in Phrase Structures

- Phrase Structures focus on the composition of phrases into chunks, on how words group together to form phrases.
- Is structure then all that matters for grammars that focus on phrase structure?
- Not exactly: phrase structure is what syntactic analysis is mainly about in these approaches, but dependencies can (generally) be derived from phrase structure trees.
When the head of the phrase is well defined, it is straightforward to deduct (unlabeled) dependencies from a phrase structure tree.

E.g.:

```plaintext
S
  /   \
NP   VP
  /     \
N     V
     /   \NP
    /     \
   N     AP
      /   |
     V     N
    /   |
   NP   PP
  /  |
N   P
/  |
A   N
/  |
strawberries  sugar
    |  |
    fresh
```
What is the head of the sentence?

S

NP  VP

N  V  NP

Mary  likes  NP  PP

N  P  NP

AP  N  with  N

A  strawberries  sugar

fresh
Example: converting a PS-tree to dependencies

What is the head of the sentence?

What is the head of the sentence?

→ we’ll let ’likes’ percolate up in the tree
Example: converting a PS-tree to dependencies

What are the dependents of 

likes

<table>
<thead>
<tr>
<th>NP</th>
<th>likes</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>likes</td>
</tr>
<tr>
<td></td>
<td>NP</td>
</tr>
<tr>
<td>Mary</td>
<td>likes</td>
</tr>
<tr>
<td>NP</td>
<td>PP</td>
</tr>
<tr>
<td>N</td>
<td>P</td>
</tr>
<tr>
<td>AP</td>
<td>N</td>
</tr>
<tr>
<td>A</td>
<td>strawberries</td>
</tr>
<tr>
<td>fresh</td>
<td></td>
</tr>
</tbody>
</table>

| sugar |
What are the dependents of likes?

\[
\text{likes} \\
\text{NP} \\
\text{N} \\
\text{Mary} \\
\text{likes} \\
\text{NP} \\
\text{NP} \\
\text{PP} \\
\text{N} \\
\text{AP} \\
\text{A} \\
\text{strawberries} \\
\text{N} \\
\text{sugar} \\
\text{fresh} \\
\]

→ Let’s look at the daughters of ’likes’
What are the heads of the daughters of ’likes’?

likes

 NP
  | NP
  | NP
  | N
  | NP
  | PP
  | N
  | P
  | NP
  | AP
  | N
  | with
  | N
  | A
  | strawberries
  | sugar
  | fresh
 CSC 1960 11/20

Example: converting a PS-tree to dependencies

What are the heads of the daughters of 'likes'? 

likes

NP    NP
|      |
N      N
|      |
Mary   NP    PP
|      |
AP    N    P    NP
|      |
A  strawberries    sugar
|      |
fresh

→ We’ll take the same steps as for ‘likes’
To identify the last dependencies, we will take the same steps as before:

1. Label mother nodes with their lexical heads
2. Remove redundant nodes

Example: converting a PS-tree to dependencies

```
likes
  /   
/     
Mary   strawberries
      /   
     /    
    AP    PP
   /  
  A   P
    /  
   fresh with
      /  
       N
         /  
          sugar
```
Example: converting a PS-tree to dependencies

To identify the last dependencies, we will take the same steps as before:

1. Label mother nodes with their lexical heads
2. Remove redundant nodes

```
likes
  
Mary - strawberries
  
AP PP

A P NP

fresh with

sugar
```
Converting a PS-tree to dependencies

Steps to take:

1. Start at the root of the tree
2. Identify lexical head of the phrase
3. Percolate the lexical head up to its maximal projection
4. Remove redundant nodes from the tree
5. Repeat steps 2-4 for all maximal projections in the tree
Dependency relations

- Since X-bar, heads are easily identifiable in phrase structures.
- So we can easily identify heads and their dependents.
- But what about their labels?
- They can be defined with respect to the tree: Recall:
  - subject-of [NP, S]
  - object-of [NP, VP]
  - etc.
- Naturally, this means that we need to integrate labels before removing redundant nodes from the tree.
Deriving a Dependency Tree from a PS Tree

S
  NP  VP
    N  V  NP
      Mary  likes  NP
            P  PP
               N  NP
                  with  N
                                  sugar
                                 N
                                 with  N
                                   strawberries
                                  AP
                                    A  N
                                      fresh
Deriving a Dependency Tree from a PS Tree

S

subj

NP

subj

N

Mary

V

likes

NP

obj

N

strawberries

V

with

NP

ad

N

fresh

with

NP

Ad

AP

strawberries

with

NP

comp

N

sugar
Deriving a Dependency Tree from a PS Tree

```
likes
  
subj
  Mary
  
likes
  
obj
  strawb.
  
with
  comp
  sugar
  
strawb.
  
strawb. with
  ad
  
strawb.
  
with
  comp
  sugar
  
strawberries
  
strawb. with
  ad
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
  
strawb. with
  ad
  fresh
  
strawb.
Deriving a Dependency Tree from a PS Tree

likes

 Mary

strawberries

fresh

with

sugar

subj

obj

ad

comp
Dependencies can be derived from phrase structures, because phrases consist of a head and its dependents (if it has any).

Similarly, you can derive phrase structures from dependencies by grouping heads and their dependents together.

Just like we needed definitions on structures to derive the labels for our dependencies, some additional information is necessary to derive a well-formed PS-tree.
To derive a PS-tree from a dependency representation it is necessary to define

1. how constituents of a phrase are ordered relative to each other (if linear order is not registered somehow in the dependency representation)
2. how to map relations to the correct $\bar{X}$-level formation
To a certain extend, phrases and dependencies present the same information:
A set of principles allows you to map from one to the other

This points to an interesting property of language:
- A head and its dependents tend to group together in the surface string (i.e. they form a continuous phrase)

Phrase Structures seem to reflect this fact in their approach to syntax, but what about dependency grammars?
Both Mel’čuk (1988) and Hudson (2007) mention the tendency of words to form continuous phrases as an important property of language.

It seems to hold cross-linguistically; there are exceptions in most languages, but they generally concern ‘marked’ structures (except maybe Dutch and Swiss German).

According to Mel’čuk (1988) this observation was first made by Hays and LeCref (around 1960), but note that it was already (implicitly) used in transformational syntax.

In Dependency Grammars this property of word order is captured by the **Projectivity** or the **Adjacency** principle.
A sentence is projective if and only if among the arcs of dependency linking its wordforms:

(i) No arc crosses another arc:

\[ [*w1 \rightarrow w2, w3, w4] \]

(ii) No arc crosses the top node:

\[ [*w1, w2 \leftarrow w3, w4] \]

Mel’čuk (1988; p.35-36)
A sentence is projective if and only if we can draw a dependency tree from which each node can be connected by a vertical line to its corresponding form in the surface string without crossing another line.

```
Mary likes strawberries with sugar
```

```
Mary likes fresh Strawberries with sugar
```
Adjacency Principle

’If A depends directly on B [...], and some other element C intervenes between them (in linear order of strings), then C directly depends on A or on B or on some other intervening element.’

Hudson (1984: p.98-99)
Word Grammar assumes **strict projectivity** (Hudson 2003)

In other words: all well-formed expressions must be projected

Word Grammar must thus find a way to deal with discontinuous phrases (see final slides of the next lecture, if you are interested)
Bibliography I


