Dependency Grammars
Lecture 2

Syntactic Theory
Winter Semester 2009/2010

Antske Fokkens

Department of Computational Linguistics
Saarland University

27 October 2009

Outline

Outline

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:

  $\rightarrow$ 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 $\rightarrow$ B, there is a dependency relations between A and B, where A **governs** B or B **depends on** A
- A dependency relations is:
  - Antisymmetric (if A $\rightarrow$ B, then B $\nrightarrow$ A)
  - Antireflexive (if A $\rightarrow$ B, then B $\neq$ A)
  - Antitransitive (if A $\rightarrow$ B and B $\rightarrow$ C, then A $\nrightarrow$ 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 $\rightarrow$ B, or B $\rightarrow$ 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 direction of a dependency

- 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

Dependency relations (1/4)

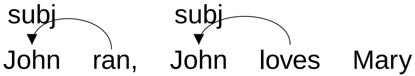
- Theories differ in the kind of dependencies that they distinguish and the labels they use for specific relations
- A fundamental distinction (found in (almost) all approaches) is the difference between **arguments** and **adjuncts**
  - A head subcategorizes for its arguments. They are often (but not 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

Dependency relations (2/4)

In this lecture, we will use the following dependency relations (adapted from Kahane (2003) and Hudson (2007)).

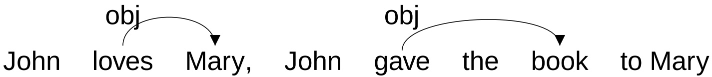
- **subj(ect)**: subject-of

  e.g.



- **obj(ect)**: object-of

  e.g.



Dependency relations (3/4)

- **obj(ect)2**: secondary-object-of

  obj2

  e.g. John gave Mary the book

- **prep(ositional)**: prepositional-complement-of

  e.g.

  prep prep

  John gave the book to Mary, it depends on you

- **comp(lement)**: complement-of

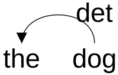
  e.g.

  comp comp comp

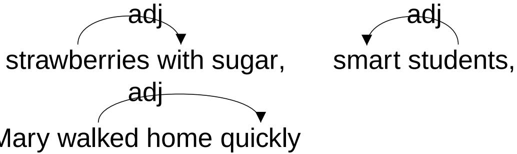
  strawberries with sugar, I saw that John ran

Dependency relations (4/4)

- **det(erminer)**: determiner-of

e.g. 

- **ad(junct)**: adjunct-of

e.g. 

- 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

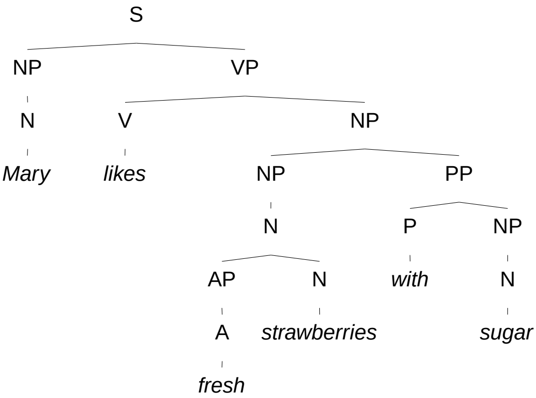
Syntactic relations in Phrase Structures

- 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

From Phrase Structures to Dependencies

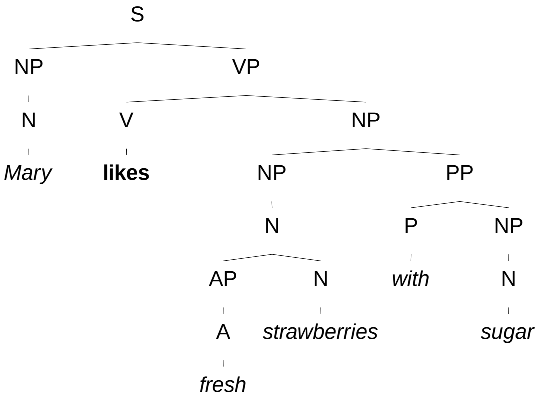
- When the head of the phrase is well defined, it is straight-forward to deduct (unlabeled) dependencies from a phrase structure tree

E.g. :



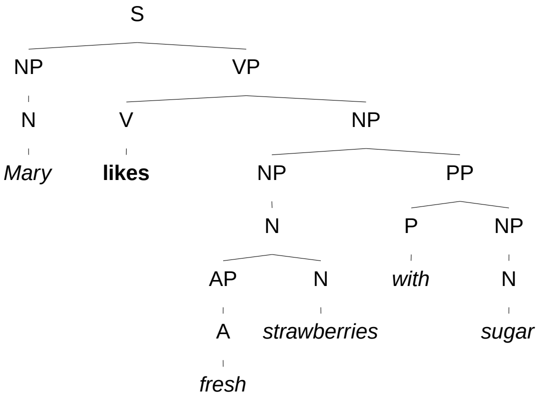
Example: converting a PS-tree to dependencies

- What is the head of the sentence?



Example: converting a PS-tree to dependencies

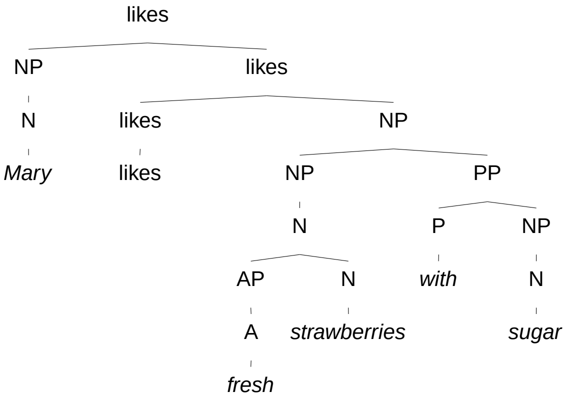
- What is the head of the sentence?



→ we'll let 'likes' percolate up in the tree

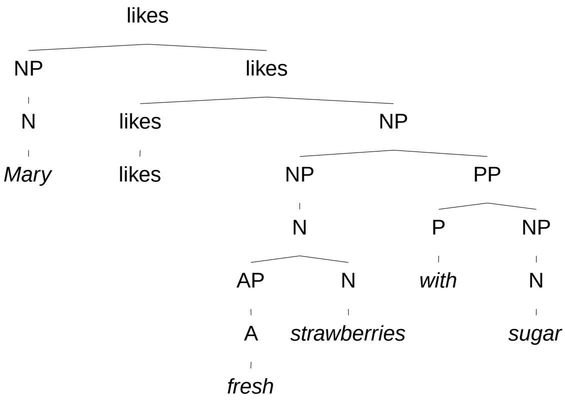
Example: converting a PS-tree to dependencies

- What are the dependents of likes?



Example: converting a PS-tree to dependencies

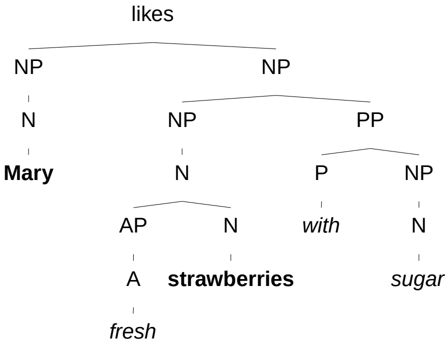
- What are the dependents of likes?



→ Let's look at the daughters of 'likes'

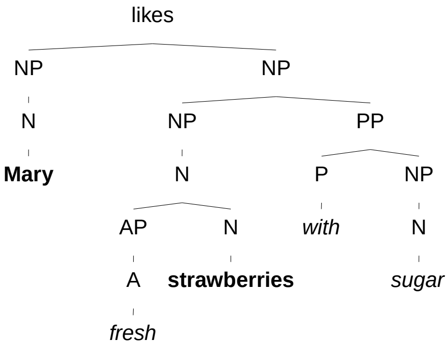
Example: converting a PS-tree to dependencies

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



Example: converting a PS-tree to dependencies

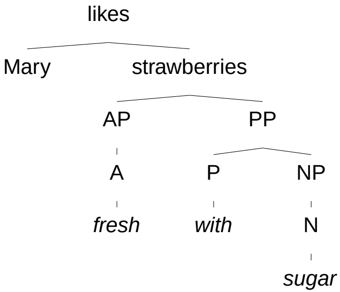
- What are the heads of the daughters of ’likes’?



→ We’ll take the same steps as for ’likes’

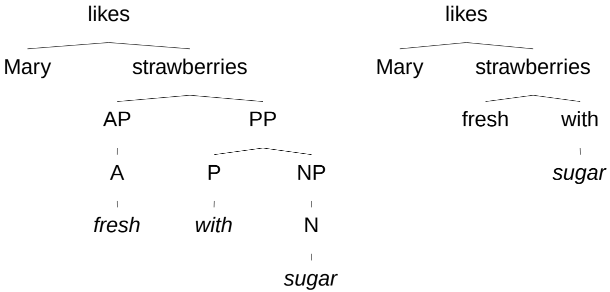
Example: converting a PS-tree to dependencies

- 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



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



Converting a PS-tree to dependencies

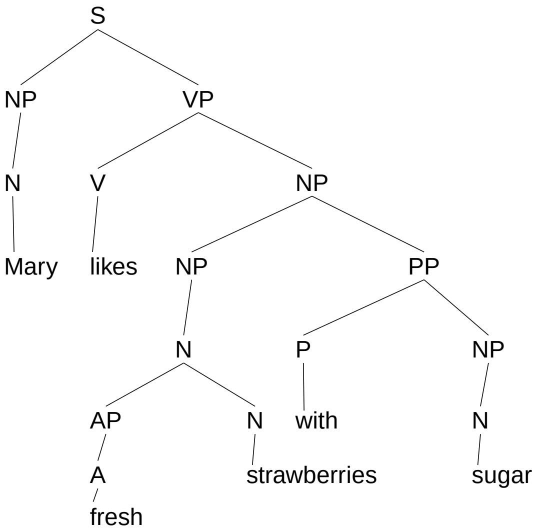
Steps to take:

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

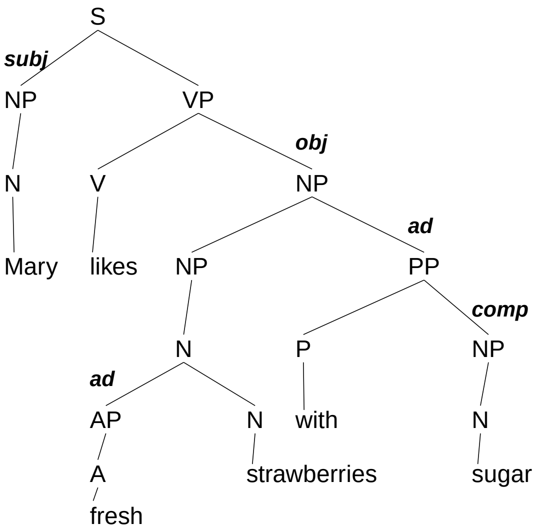
Dependency relations

- Since $\overline{\text{X}}$-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

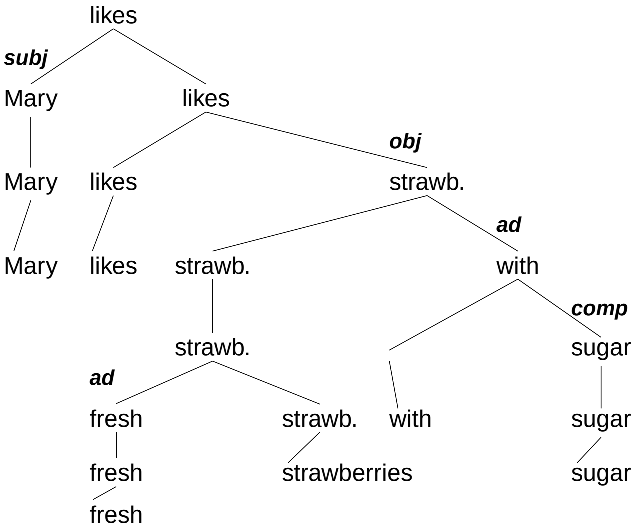
Deriving a Dependency Tree from a PS Tree



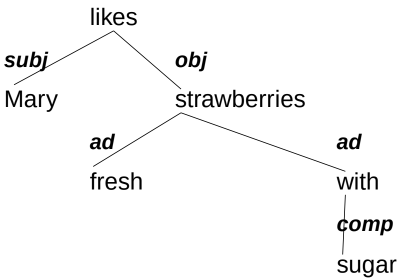
Deriving a Dependency Tree from a PS Tree



Deriving a Dependency Tree from a PS Tree



Deriving a Dependency Tree from a PS Tree



Going from Dependencies to Phrase Structure

- 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

From Dependencies to Phrases

- 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 $\overline{\text{X}}$-level formation

Phrase Structures and Dependencies

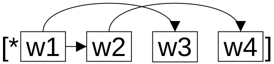
- To a certain extend, phrases and dependencies present the same information:
  A set of principles allows you to map from one to the other
- This points to an interesting property of language:
  - a head and its dependents tend to group together in the surface string (i.e. they form a continuous phrase)
- Phrase Structures seem to reflect this fact in their approach to syntax, but what about dependency grammars?

Projectivity or Adjacency

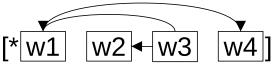
- Both Mel'čuk (1988) and Hudson (2007) mention the tendency of words to form continuous phrases as an important property of language
- It seems to hold cross-linguistically; there are exceptions in most languages, but they generally concern 'marked' structures (except maybe Dutch and Swiss German)
- 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.

Projectivity/Adjacency (1)

- A sentence is projective if and only if among the arcs of dependency linking its wordforms:
  - (i) No arc crosses another arc:



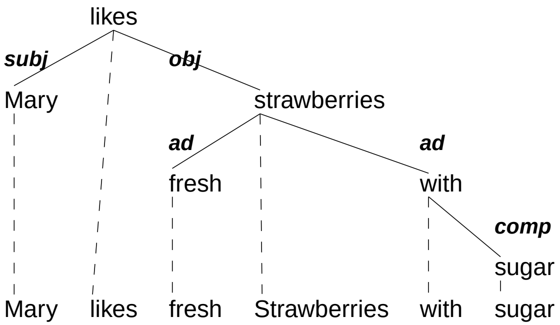
  - (ii) No arc crosses the top node:



Mel'čuk (1988; p.35-36)

Projectivity/Adjacency (2)

- 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



Projectivity/Adjacency (3)

- 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)

Projectivity as principle

- 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

- Haegeman, Liliane (1991). *Introduction to Government and Binding Theory*. Oxford, UK: Blackwell Publishers.
- Hudson, Richard A. (1984). *Word Grammar*. New York, USA: Blackwell.
- Hudson, Richard A. (1990). *English Word Grammar*. Oxford, UK: Blackwell. http://www.phon.ucl.ac.uk/home/dick/ewg.htm.
- Hudson, Richard A. (2007). *Language Networks - The new word grammar*. New York, USA: Oxford Press.
- Kahane, Sylvain (2003). The Meaning-Text Theory. *Dependency and Valency. Handbooks of Linguistics and Communication Sciences 25 1-2*. Berlin, Germany: De Gruyter.
- Kordoni, Valia (2008a). Syntactic Theory Lectures 1 and 2. Course slides.
- Kordoni, Valia (2008b). Syntactic Theory Lectures 3 and 4. Course slides.

Bibliography II

- Mel'čuk, Igor (1988) *Dependency Syntax: Theory and Practice*. Albany, USA: State University of New York Press.
- Ouhalla, Jamal (1994). *Introducing Transformational Grammar*. New York, USA: Oxford University Press.
- Sag, Ivan A., Thomas Wasow and Emily M. Bender (2003). *Syntactic Theory*. A Formal Introduction. Palo Alto, USA: CSLI Publications.
- Schneider, Gerold (1998). *A Linguistic Comparison of Constituency, Dependency and Link Grammar. Lizentiatsarbeit, Institut für Informatik der Universität Zürich*. http://www.ifi.unizh.ch/cl/study/lizarbeiten/lizgerold.pdf.
- Zwicky, Arnold (1985). Heads. *Journal of Linguistics* 12, 1-30.