Dependency Grammars

Petr Horáček, Eva Zámečníková and Ivana Burgetová

Department of Information Systems
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, 612 00 Brno, CZ

FRVŠ MŠMT FR97/2011/G1

Outline

- **Introduction**
- **Dependency Grammars vs. PSG**
- **Dependency Formalism**

Topic

- **Introduction**
- **Dependency Grammars vs. PSG**
- **Dependency Formalism**

Dependency Grammars

Dependency Grammars

- Alternative to phrase structure grammars (PSG).
- Capture direct relations between words in a sentence.
  - No phrasal nodes.

- The term dependency grammar actually covers many particular formalisms.
  - Theory of Structural Syntax (Tesnière, 1959) – considered the starting point of modern dependency grammar theory
  - Word Grammar (WG) (Hudson, 1984)
  - Functional Generative Description (FGD) (Sgall et al., 1986)
  - Meaning-Text Theory (MTT) (Mel’čuk, 1988)
  - Extensible Dependency Grammar (XDG) (Debusmann et al., 2004)
  - . . .
- Here we will discuss the common core points of these theories, and compare dependency grammars and PSG.

Topic

- Introduction
- **Dependency Grammars vs. PSG**
- Dependency Formalism

Phrase Structure Grammar

Definition

A phrase structure grammar (PSG) G is a quadruple
$G = (N, T, P, S)$, where

- N is a finite set of *nonterminals*,
- T is a finite set of *terminals*, $N \cap T = \emptyset$
- $P \subseteq (N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$ is a finite relation – we call each $(x, y) \in P$ a *rule* (or *production*) and usually write it as
$$x \to y,$$
- $S \in N$ is the *start symbol*.

Phrase Structure Grammar

Derivation in PSG

Let G be a PSG. Let $u, v \in (N \cup T)^*$ and $p = x \rightarrow y \in P$. Then, we say that uxv directly derives uyv according to p in G, written as $uxv \Rightarrow_G uyv\,[p]$ or simply

$$uxv \Rightarrow uyv$$

We further define $\Rightarrow^+$ as the transitive closure of $\Rightarrow$ and $\Rightarrow^*$ as the transitive and reflexive closure of $\Rightarrow$.

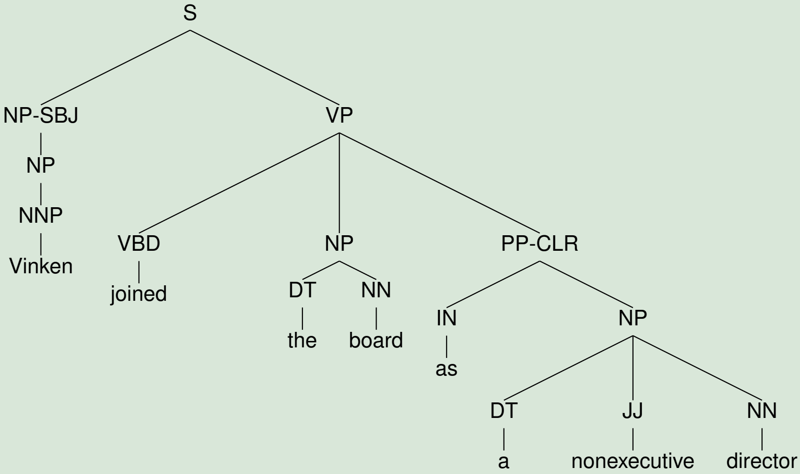
Generated Language

Let G be a PSG. The language generated by G is defined as

$$L(G) = \{w : w \in T^*, S \Rightarrow^* w\}$$

PSG – Derivation Tree

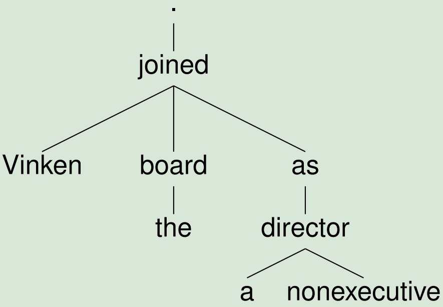
Example



(Adapted from Penn Treebank)

Dependency Tree

Example



.

joined

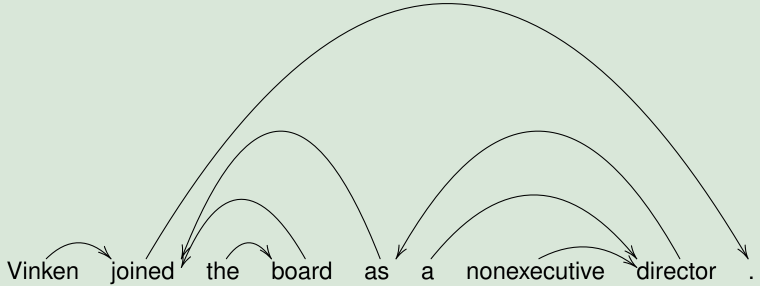
Vinken board as

the director

a nonexecutive

Dependency Tree

Example



Dependency Grammars vs. PSG

Advantages

- Simplicity
  - Easy to understand.
  - Faster manual annotation of sentences in corpora (in PSG, the trees are generally much more complicated, and we also need some base set of grammar rules).
  - Efficient parsing.
- Robustness and portability
  - Can parse any sentence.
  - Uniformly applicable to many languages.
- Permutations of words without affecting syntactic structure are possible.
  - Useful for free word order languages (such as Czech).

Disadvantages

- Less informative (but still useful in practice)
  - There is less explicit information about the constituents of the sentence (nonterminals in PSG).

Topic

- Introduction
- Dependency Grammars vs. PSG
- **Dependency Formalism**

Dependency

Idea

Syntactic structure of a sentence consists of binary asymmetrical relations between the words of the sentence.

- Words in dependency relation – various names in different formalisms:
    - Parent – Child
    - Head – Modifier
    - Governor – Dependent
    - . . .
- Arrows from child to parent.
    - May also be drawn in opposite direction, depending on authors.

Dependency

Notation

- If w is child and v is its parent, we write

$$w \rightarrow v$$

- If there is a path from w to v, we write

$$w \rightarrow^{*} v$$

  (transitive closure)

Dependency Tree – Properties

1. Single head – each word has one and only one parent (except for the root node).

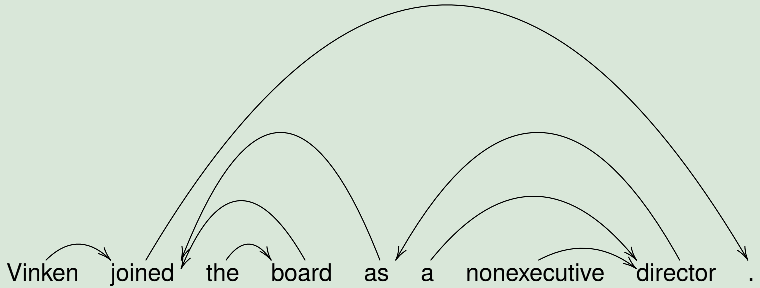
2. Connected – all words form a connected graph.

3. Acyclic – if $w_i \rightarrow w_j$, $w_j \rightarrow^* w_i$ never holds.
   - The graph does not contain cycles.
   - Note: w_i denotes i-th word in sentence.

4. Projective – if $w_i \rightarrow w_j$, then for all w_k, where $i < k < j$, either $w_k \rightarrow^* w_i$ or $w_k \rightarrow^* w_j$ holds.
   - Non-crossing between dependencies.
   - Some dependency formalisms allow non-projectivity.

Projective Dependency Tree

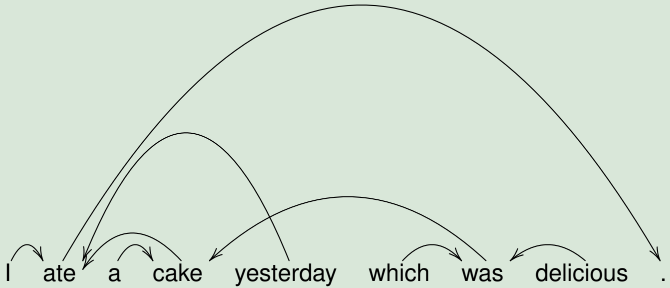
Example



- There is no crossing of dependencies.
- For example, all the words between “joined” and “.” finally depend on either “joined” or “.”
  - nonexecutive $\rightarrow^*$ joined

Non-projective Dependency Tree

Example

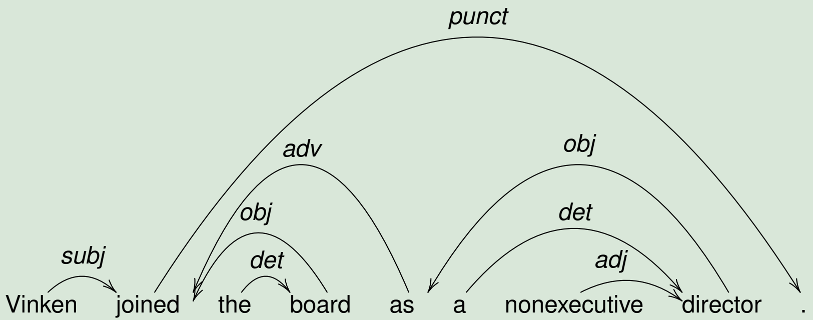


- There are crossing dependencies.
    - yesterday → ate
    - was → cake

Dependency Tree with Labels

- We may want to know not only which word depends on which, but also how.
- We can assign labels to dependencies.

Example



Root Node

- In PSG, the root node of derivation tree is given by the starting nonterminal of the grammar.
  - Usually corresponds to the whole sentence.

- What should be the root of dependency tree?
  - There is nothing like nonterminal symbols in dependency grammars.

- Different authors use different notations.
- For example, the root node can be:
  - Punctuation mark (“.”) – we use this notation
  - Verb
  - Some abstract root symbol

References

- Ralph Debusmann, Denys Duchier, Geert-Jan Kruijff:
  Extensible Dependency Grammar: A New Methodology,
  Proceedings of the Workshop on Recent Advances in Dependency Grammar, p. 78-85, 2004
- Richard Hudson:
  Word Grammar,
  Blackwell, 1984
- Igor Mel'čuk:
  Dependency Syntax: Theory and Practice,
  State University of New York Press, 1988
- Lucien Tesnière:
  Éléments de syntaxe structurale,
  Editions Klincksieck, 1959
- Petr Sgall, Eva Hajičová, Jarmila Panevová, Jacob Mey:
  The meaning of the sentence in its semantic and pragmatic aspects,
  Springer, 1986