Mathematics for linguists

Gerhard Jäger
gerhard.jaeger@uni-tuebingen.de

Uni Tübingen, WS 2009/2010

November 17, 2009

Trees

Tree diagrams

A tree diagram of a sentence represents three kins of information:

- the constituent structure of the sentence,
- the grammatical category of each constituent, and
- the linear order of the constituents.

Trees

Conventions

- A tree consists of *nodes*, which are connected by
- **edges**
- By convention, edges are **directed** downward.
- Every node has a **label**.

Trees

Dominance

- A node x **dominates** a node y if there is a connected sequence of directed edges that start with x and end with y.
- For a given treen T,

$$D_T := \{\langle x, y\rangle | x \text{ dominates } y \text{ in } T\}$$

  is the corresponding **dominance relation**
- D_T is a weak ordering, i.e. it is reflexive, transitive and anti-symmetric.

Trees

Conventions

- If x is the immediagte predecessor of y in D_T, then x **immediately dominates** y.
- The immediate predecessor of x according to D_T is called the **mother node** of x.
- The immediate successors of x are called the **daughter nodes** of x.
- If two nodes are not identical but have the same mother node, then they are called **sister nodes**.
- Every tree has finitely many trees.
- Every tree has a least element. The least element is called **root** or **root node**of the tree.
- The maximal elements of a tree are called **leaves**.

Precedence

- Tree diagrams contain information on the linear order of nodes.
- Node x **precedes** node y iff x is to the left of y and neither of the two nodes dominates the other one.
- For a tree T,

  $$P_T := \{\langle x, y\rangle | x \text{ precedes } y\}$$

  is the corresponding **precedence relation**.
- P_T is a strict ordering, i.e. it is irreflexive, transitive and asymmetric.

Exclusivity

In a tree T, any two nodes x and y are related by precedence (i.e. $P_T(x, y)$ or $P_T(y, x)$) iff they are not related by dominance (i.e. neither $D_T(x, y)$ nor $D_T(y, x)$).

No crossing

If in a tree T, node x precedes node y, then every node x' that is dominated by x precedes every node y' that is dominated by y.

This condition prevents that

- One node has several mother nodes, and that
- edges cross.

Trees

Labeling

For every tree T there is a labeling function L_T which assigns a label to each node.

- L_T need not be injective (several nodes may have the same label).
- In derivation trees, leaves (also called **terminal nodes**) are mapped to terminal symbols, and all other nodes to non-terminal symbols.

Using these properties of trees, we can prove **theorems**, i.e. facts that hold for all trees. For instance

Theorem

If x and y are sister nodes, than either $P(x, y)$ or $P(y, x)$.

Theorem

The set of leaves of a tree are linearly ordered by P.

Grammars and trees

- Trees represent the relevant aspects of a derivation.
- Connection between derivaton and tree is most transparent if all rules of the grammar have the form

$$A \to \alpha$$

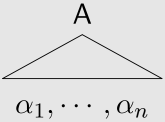
(with $A \in V_N$ and $\alpha \in (V_T \cup V_N)^*$)

Grammars and trees

Definition

A grammar $G = \langle V_T, V_N, S, R \rangle$ where all rules have exactly one non-terminal symbol as left hand side **generates** a tree T iff

- the root of T is labeled with S,
- the leaves are labeled either with terminal symbols or with ϵ, and
- for each sub-tree



  in T, there is a rule $A \to \alpha_1, \cdots, \alpha_n$ in R.

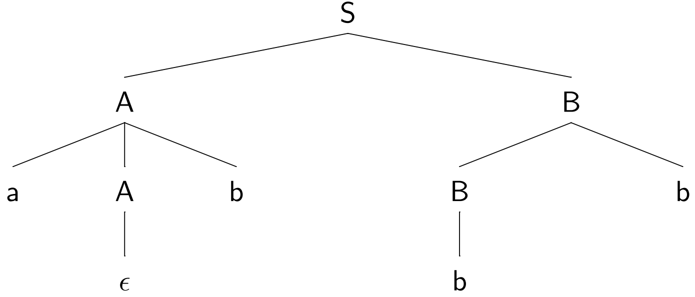
Grammars and trees

Example grammar

$$
\begin{array}{rcl}
G & = & \langle \{a, b\}, \{S, A, B\}, S, R \rangle \\
R & = & \left\{ \begin{array}{ll} S \rightarrow AB & B \rightarrow Bb \\ A \rightarrow aAb & B \rightarrow b \\ A \rightarrow \epsilon & \end{array} \right\}
\end{array}
$$

Grammars and trees

This grammar generates for instance the following tree:



Question: Which language is generated by this grammar?

Context-sensitive rules

Sometimes it is desirable to restrict the applicability of a certain rule to specific contexts. For instance:

- $D \rightarrow$ *des* only if the following noun is masculin or neuter singular genitive
- $/d/ \rightarrow [d]$ only if this segment is not at the end of a word
- [past, 1.pers] $\rightarrow -t-$ only if it is preceded by the stem of a weak verb
- ...

Question: Can you think of more examples for context-sensitive rules?

Context-sensitive rules

- usual format for context-sensitive rules:

$$A \rightarrow \gamma / \alpha \_ \beta$$

- A: non-terminal symbol
- α, β, γ: string of terminal and non-terminal symbols
- $\gamma \neq \epsilon$
- $\alpha \_ \beta$ is the context in which the rule $A \rightarrow \gamma$ can be applied
- "official" notation:

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

The Chomsky hierarchy

Different restrictions for the format of rules of a grammar lead to the following hierarchy of grammar types:

Chomsky hierarchy

Typ 0 no restrictions

Typ 1 rules of the form *context-sensitive grammar*
$S \rightarrow \epsilon$ or $\alpha A \beta \rightarrow \alpha \gamma \beta$
$A, S \in V_N$ (S start symbol), $\alpha, \beta, \gamma \in (V_T \cup V_N)^*$, $\gamma \neq \epsilon$

If $S \rightarrow \epsilon$ is a rule, then S never occurs as the right hand side of a rule.

Typ 2 Rules of the form $A \rightarrow \gamma$ *context-free grammar*
$A \in V_N$, $\gamma \in (V_T \cup V_N)^*$

Typ 3 Rules of the form $A \rightarrow \vec{x} B$ *regular grammar*
or $A \rightarrow \vec{x}$
$A, B \in V_N$, $\vec{x} \in V_T^*$

The Chomsky hierarchy

- no strict hierarchy, because ϵ may occur as right hand side in context-free gramamrs, but no (in the general case) in context-free grammars

$$\textbf{Typ 3} \subset \textbf{Typ 2} \not\subseteq \textbf{Typ 1} \subset \textbf{Typ 0}$$

The Chomsky hierarchy

Grammar hierarchy corresponds to hierarchy of formal languages:

- *Type-0 languages* ("recursively enumerable languages"): languages that are generated by type-0 grammars
- *Type-1 languages* ("context-sensitive languages"): languages that are generated by type-1 grammars
- *Type-2 languages* ("context-free languages"): languages that are generated by type-0 grammars
- *Type-3 languages* ("regular languages"): languages that are generated by type-0 grammars

Theorem

If L is a context-free language, than it is also a context-sensitive language.

The Chomsky hierarchy

- All context-sensitive languages are **decidable** — for each of these languages, there is a computer program that can decide in finite time whether or not a given string belongs to that language.
- Recursively enumerable languages are not always decidable. For instance, the set of all provable mathematical statements is a recursively enumerable language that is not decidable.
- Context-free languages can be processed efficiently by a computer (time complexity is maximally cubic).
- Regular languages can be processed very efficiently by a computer (time complexity is maximally linear).
- Context-sensitive languages can not alway be processed efficiently by a computer.

The Chomsky hierarchy

- 1957 (Chomsky): proof that English is not a regular language
- 1957 (Chomsky): conjecture that natural languages are generally not context-free, but context-sensitivel
- 1982 (Pullum & Gazdar): „Natural Languages and Context-Free Languages“ — arguments that neither English nor any other natural language has so far clearly proven to be not context-free.
- 1984 (Huybregts), 1985 (Shieber): proof that Swiss German is not context-free
- Most phonological and morphological processes in natural languages can be captured by regular grammars.

The Chomsky hierarchy