Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Bosanski
Čeština
Deutsch
فارسی
Français
한국어
Hrvatski
Italiano
Latina
Nederlands
日本語
Polski
Português
Српски / srpski
Srpskohrvatski / српскохрватски
Svenska
Українська
Tiếng Việt
中文
Edit links

Parse tree

From Wikipedia, the free encyclopedia

A **parse tree** or **parsing tree**[1] or **derivation tree** or **(concrete) syntax tree** is an ordered, rooted tree that represents the syntactic structure of a string according to some context-free grammar. The term *parse tree* itself is used primarily in computational linguistics; in theoretical syntax the term *syntax tree* is more common. Parse trees are distinct from the abstract syntax trees used in computer programming, in that their structure and elements more concretely reflect the syntax of the input language. They are also distinct from (although based on similar principles to) the sentence diagrams (such as Reed-Kellogg diagrams) sometimes used for grammar teaching in schools.

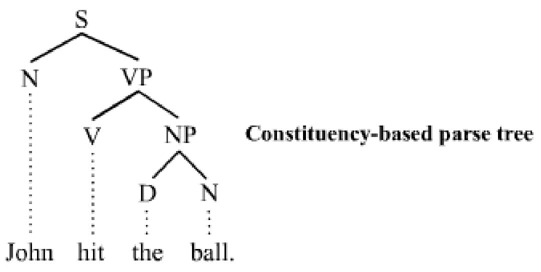
Parse trees are usually constructed based on either the constituency relation of constituency grammars (phrase structure grammars) or the dependency relation of dependency grammars. Parse trees may be generated for sentences in natural languages (see natural language processing), as well as during processing of computer languages, such as programming languages.

A related concept is that of **phrase marker** or **P-marker**, as used in transformational generative grammar. A phrase marker is a linguistic expression marked as to its phrase structure. This may be presented in the form of a tree, or as a bracketed expression. Phrase markers are generated by applying phrase structure rules, and themselves are subject to further transformational rules.

Contents [hide]

Constituency-based parse trees [edit]

The constituency-based parse trees of constituency grammars (= phrase structure grammars) distinguish between terminal and non-terminal nodes. The interior nodes are labeled by non-terminal categories of the grammar, while the leaf nodes are labeled by terminal categories. The image below represents a constituency-based parse tree; it shows the syntactic structure of the English sentence *John hit the ball*:



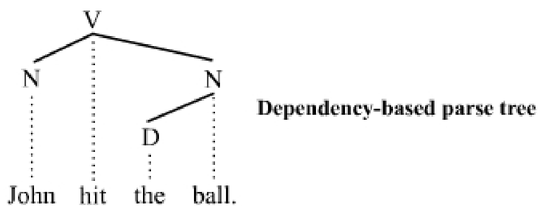
The parse tree is the entire structure, starting from S and ending in each of the leaf nodes (*John*, *hit*, *the*, *ball*). The following abbreviations are used in the tree:

- S for sentence, the top-level structure in this example
- NP for noun phrase. The first (leftmost) NP, a single noun "John", serves as the subject of the sentence. The second one is the object of the sentence.
- VP for verb phrase, which serves as the predicate
- V for verb. In this case, it's a transitive verb *hit*.
- D for determiner, in this instance the definite article "the"
- N for noun

Each node in the tree is either a *root* node, a *branch* node, or a *leaf* node.[2] A root node is a node that doesn't have any branches on top of it. Within a sentence, there is only ever one root node. A branch node is a mother node that connects to two or more daughter nodes. A leaf node, however; is a terminal node that does not dominate other nodes in the tree. S is the root node, NP and VP are branch nodes, and *John* (N), *hit* (V), *the* (D), and *ball* (N) are all leaf nodes. The leaves are the lexical tokens of the sentence.[3] A node can also be referred to as *parent* node or a *child* node. A parent node is one that has at least one other node linked by a branch under it. In the example, S is a parent of both N and VP. A child node is one that has at least one node directly above it to which it is linked by a branch of a tree. From the example, *hit* is a child node of V. The terms *mother* and *daughter* are also sometimes used for this relationship.

Dependency-based parse trees [edit]

The dependency-based parse trees of dependency grammars[4] see all nodes as terminal, which means they do not acknowledge the distinction between terminal and non-terminal categories. They are simpler on average than constituency-based parse trees because they contain fewer nodes. The dependency-based parse tree for the example sentence above is as follows:



This parse tree lacks the phrasal categories (S, VP, and NP) seen in the constituency-based counterpart above. Like the constituency-based tree, constituent structure is acknowledged. Any complete sub-tree of the tree is a constituent. Thus this dependency-based parse tree acknowledges the subject noun *John* and the object noun phrase *the ball* as constituents just like the constituency-based parse tree does.

The constituency vs. dependency distinction is far-reaching. Whether the additional syntactic structure associated with constituency-based parse trees is necessary or beneficial is a matter of debate.

Phrase markers [edit]

Phrase markers, or P-markers, were introduced in early transformational generative grammar, as developed by Noam Chomsky and others. A phrase marker representing the deep structure of a sentence is generated by applying phrase structure rules; this may then be undergo further transformations. Phrase markers may be presented in the form of trees (as in the above section on constituency-based parse trees), but are often given instead in the form of bracketed expressions, which occupy less space. For example, a bracketed expression corresponding to the constituency-based tree given above may be something like:

$$[_S\ [_{NP}\ John]\ [_{VP}\ [_V\ hit]\ [_{NP}\ the\ [_N\ ball]]]]$$

As with trees, the precise construction of such expressions and the amount of detail shown can depend on the theory being applied and on the points that the author wishes to illustrate.

Notes [edit]

1. ^ See Chiswell and Hodges 2007: 34.
2. ^ See Carnie (2013:118ff.) for an introduction to the basic concepts of syntax trees (e.g. root node, terminal node, non-terminal node, etc.).
3. ^ See Alfred et al. 2007.
4. ^ See for example Ágel et al. 2003/2006.

See also [edit]

- Constituent (linguistics)
- Dependency grammar
- Computational linguistics
- Terminal and non-terminal functions
- Parsing
- Phrase structure grammar
- Sentence diagram
- Verb phrase

- Parse Thicket

References [edit]

- Ágel, V., Ludwig Eichinger, Hans-Werner Eroms, Peter Hellwig, Hans Heringer, and Hennig Lobin (eds.) 2003/6. Dependency and valency: An international handbook of contemporary research. Berlin: Walter de Gruyter.
- Carnie, A. 2013. Syntax: A generative introduction, 3rd edition. Malden, MA: Wiley-Blackwell.
- Chiswell, Ian and Wilfrid Hodges 2007. Mathematical logic. Oxford: Oxford University Press.
- Aho, Alfred et al. 2007. Compilers: Principles, techniques, & tools. Boston: Pearson/Addison Wesley.

External links [edit]

- Syntax Tree Editor
- Linguistic Tree Constructor
- phpSyntaxTree – Online parse tree drawing site
- phpSyntaxTree (Unicode) – Online parse tree drawing site (improved version that supports Unicode)
- Qtree – LaTeX package for drawing parse trees
- TreeForm Syntax Tree Drawing Software
- rSyntaxTree Enhanced version of phpSyntaxTree in Ruby with Unicode and Vectorized graphics
- Visual Introduction to Parse Trees Introduction and Transformation
- OpenCourseOnline Dependency Parse Introduction (Christoper Manning)

Categories: Syntax | Trees (data structures)