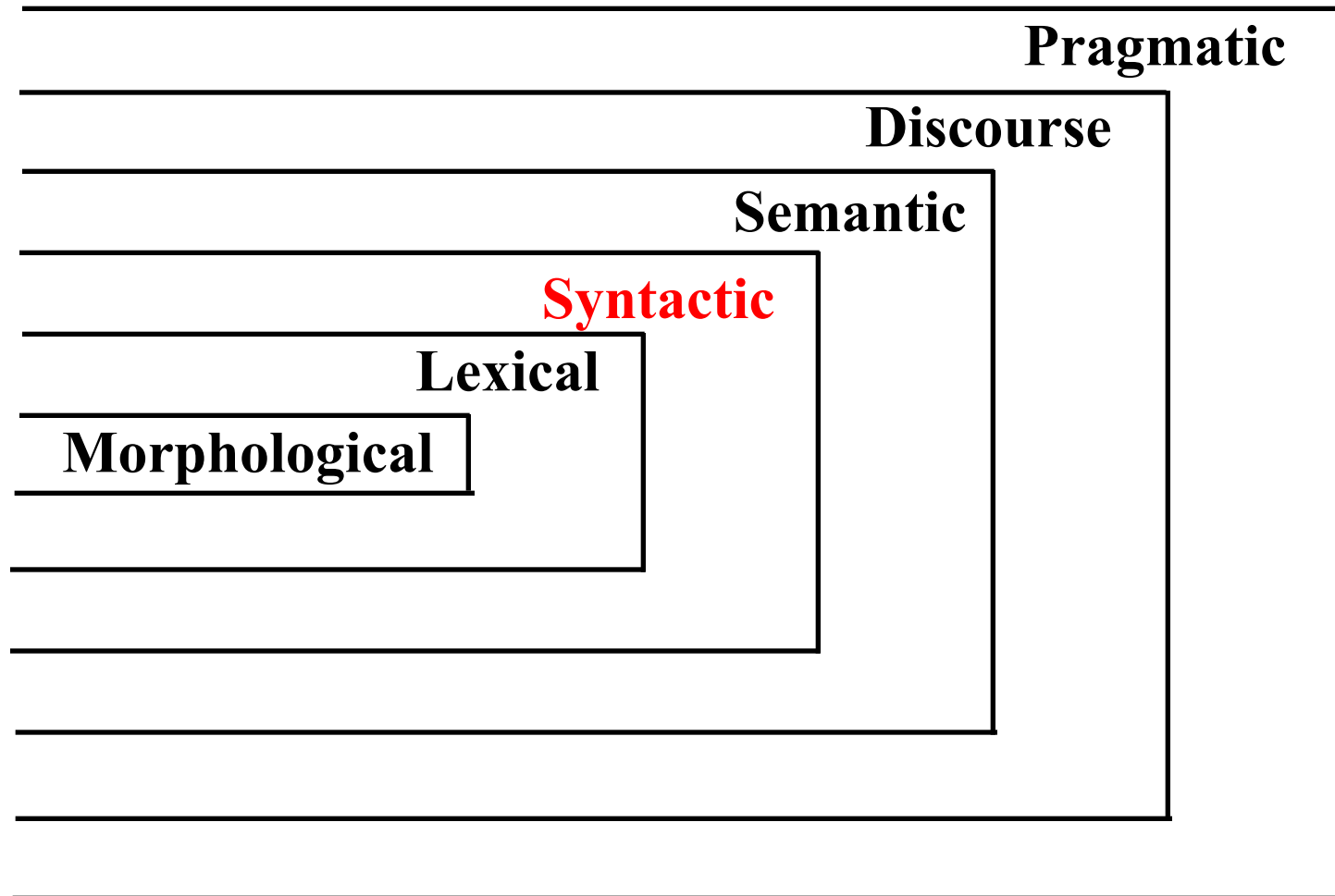

Context Free Grammars

Synchronic Model of Language



Syntactic Analysis

- Syntax expresses the way in which words are arranged together.
- The kind of implicit knowledge of your native language that you had mastered by the time you were 3 or 4 years old without explicit instruction
 - Do these word sequences fit together?
I saw you yesterday
you yesterday I year

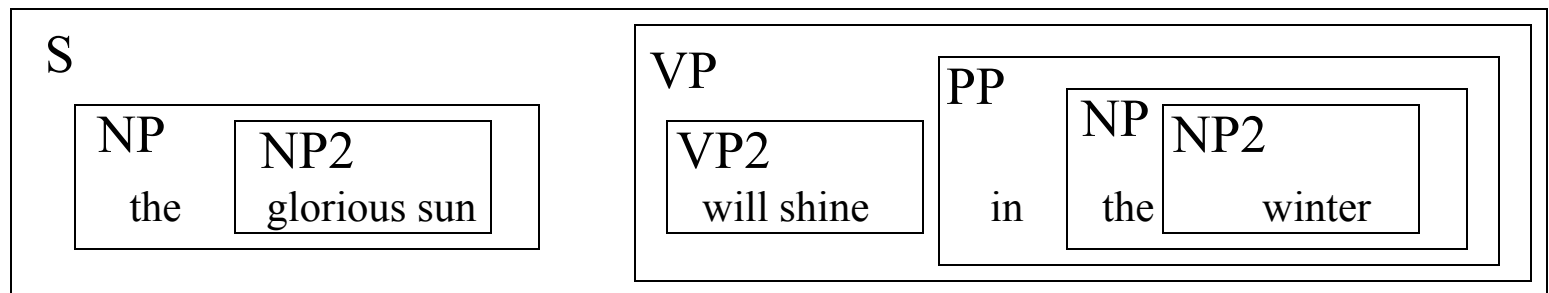
that and after year last
colorless green ideas sleep furiously (Chomsky)
- NLP uses syntax to produce a structural analysis of the input sentence

Notations for syntactic structure

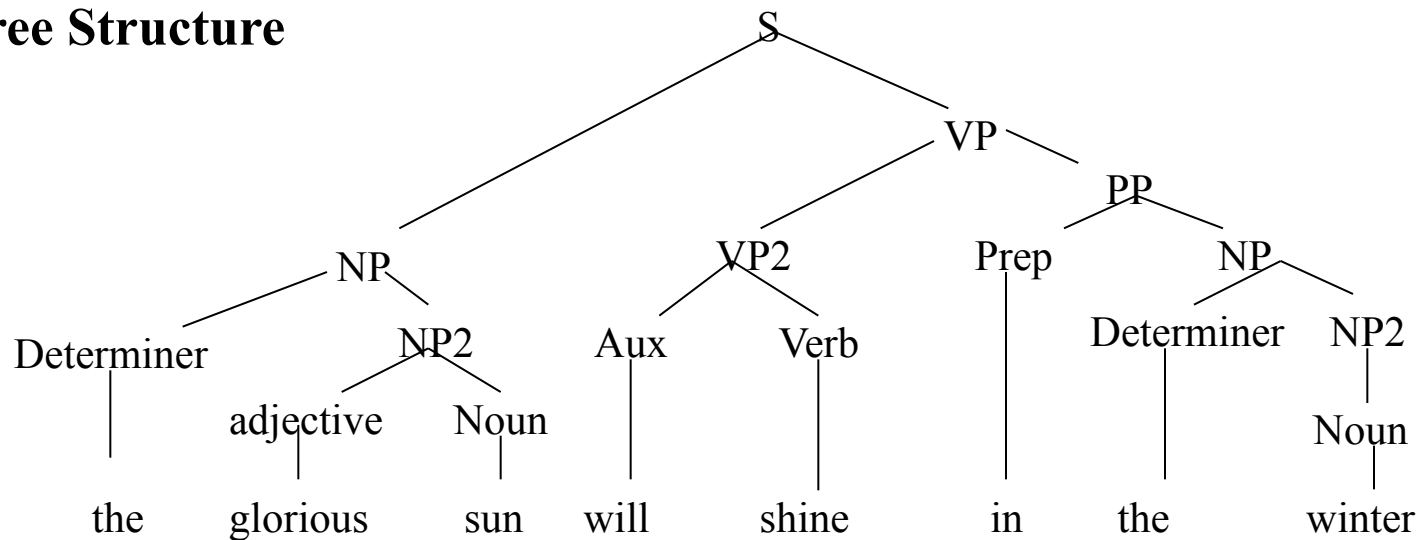
Bracketed text

[_S [_{NP} the [_{NP2} glorious sun]] [_{VP} [_{VP2} will shine] [_{PP} in [_{NP} the [_{NP2} winter]]]]]

Nested Boxes



Tree Structure



Formal Grammar

- Rules that embody generalizations that hold for the symbols and combinations of symbols in a language for constructing acceptable sentences
 - grammar is most closely identified with syntax, but may contain elements of all levels of language
- Theoretical linguists use grammar
 - To indicate which are the well-formed sentences, defining that language
 - To show how variations of a ‘deep-structure’ are derived through ‘transformations’ on this ‘deep-structure’
 - ‘Competence’ based – an ideal speaker’s internalized ability to create and understand all sentences
- Applied uses
 - To assign a structural description to the linguistic elements of which an utterance is comprised
 - “...typically not consciously modeled after any particular linguistic theory, but as descriptions of phenomena that appeared in input text.”
 - ‘Performance’ based – person’s actual use of language

Context-Free Grammars

- Capture constituency and ordering
 - Ordering is
 - What are the rules that govern the ordering of words and bigger units in the language
 - Constituency is
 - How do words group into units and what we say about how the various kinds of units behave
 - A constituent is a sequence of words that behave as a unit
 - Examples are noun phrases, verb phrases, etc.

Context-Free Grammar consists of:

- **Non-terminal symbols**

 - S, NP, VP, etc. representing the categories of phrases

- **Terminal symbols**

 - car, man, house*, representing words in the lexicon

 - The rewrite rules will include lexical insertion rules

 - (e.g. $N = car \mid man \mid house$)

- **Rewrite rules / productions**

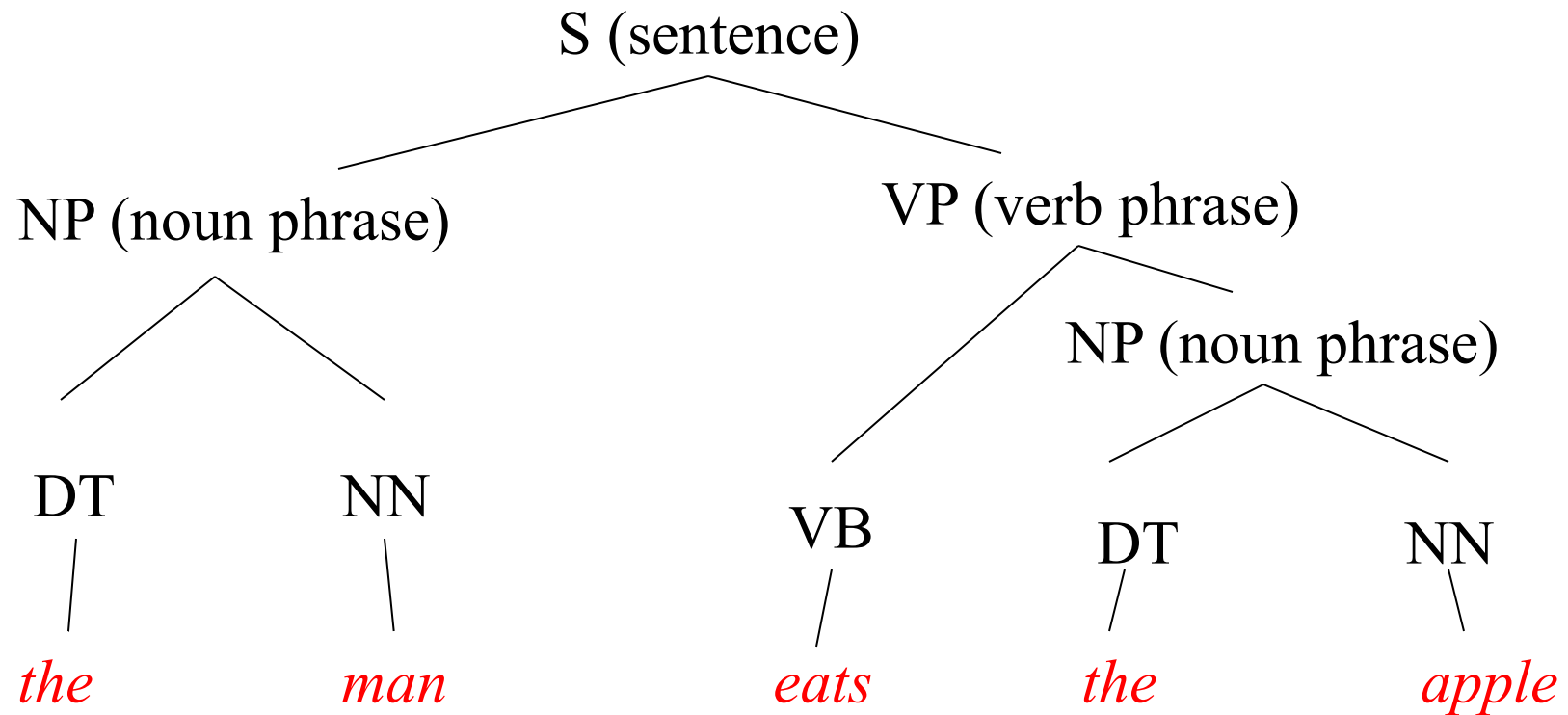
 - $S \rightarrow NP VP \mid NP VP NP$

 - (note use of \mid symbol to give alternate rhs of rules)

- A designated start symbol S

- A **derivation** is a sequence of rewrite rules applied to a string that exactly covers the items in that string

Derivation of syntax from grammar rules



Context Free Grammar Rules:

$S \rightarrow NP VP$

$NP \rightarrow DT NN$

$VP \rightarrow VB NP$

$VP \rightarrow VB$

$DT \rightarrow the$

$NN \rightarrow man$ |

$apple$

$VB \rightarrow eat$

Generativity vs. Parsing

- As with FSAs and FSTs you can view these rules as either synthesis or analysis machines
 - Generate strings in the language
 - Reject strings not in the language
 - Impose structures (trees) on strings in the language
- The latter two are the analysis tasks of parsing
 - Parsing is the process of finding a derivation (i. e. sequence of productions) leading from the START symbol to a TERMINAL symbol (or TERMINALS to START symbol)
 - Shows how a particular sentence *could be* generated by the rules of the grammar
 - If sentence is structurally ambiguous, more than one possible derivation is produced

Context-Free Grammars

- Why Context-Free?
 - The notion of **context** in CFGs has nothing to do with the ordinary meaning of the word context in language.
 - All it really means is that the non-terminal on the left-hand side of a rule can be replaced regardless of context
 - Context-sensitive grammars allow context to be placed on the left-hand side of the rewrite rule
- In programming languages, and other uses of CFGs in Computer Science, notably XML, CGS are
 - Unambiguous
 - Assign at most, 1 structural description to a string
 - Parsable in time linearly proportional to the length of the string

Key Constituents for English

- Sentences (and clauses)
 - Declaratives: A plane left
S -> NP VP
 - Imperatives: Leave!
S -> VP
 - Yes-No Questions: Did the plane leave?
S -> Aux NP VP
 - WH Questions: When did the plane leave?
S -> WH Aux NP VP
- Verb Phrases
- Noun Phrases
- Prepositional Phrases

Noun Phrases

- Noun phrases have a **head noun** with pre and post-modifiers
 - Determiners, Cardinals, Ordinals and Quantifiers are all optional, indicated here with parentheses
 - NP -> (DT) (Card) (Ord) (Quan) **Noun**
 - Noun -> NN | NP | NPS | NNS
 - Post-modifiers include prepositional phrases, gerundive phrases, and relative clauses
 - the **man** [from Moscow]
 - any **flights** [arriving after 11pm] (gerundive)
 - the **spy**[who came in from the cold] (relative clause)

Some examples on these slides are from the Jurafsky and Martin text and from Jim Martin's online course materials.

Recursive Rules

- One type of Noun phrase is a Noun Phrase followed by a Prepositional phrase

NP → NP PP

PP → Prep NP

- Of course, this is what makes syntax interesting

flights from Denver

flights from Denver to Miami

flights from Denver to Miami in February

flights from Denver to Miami in February on a Friday

flights from Denver to Miami in February on a Friday under \$300

*flights from Denver to Miami in February on a Friday under \$300 with
lunch*

Verb Phrases

- Simple Verb phrases

VP -> Verb	<i>leave</i>
Verb NP	<i>leave Boston</i>
Verb NP PP	<i>leave Boston in the morning</i>
Verb PP	<i>leave in the morning</i>

- Verbs may also be followed by a clause

VP -> Verb S
I think I would like to take a 9:30 flight

- The phrase or clause following a verb is sometimes called the complementizer

Conjunctive Constructions

- $S \rightarrow S \text{ and } S$
 - John went to NY and Mary followed him
- $NP \rightarrow NP \text{ and } NP$
- $VP \rightarrow VP \text{ and } VP$
- ...
- In fact the right rule for English is
 $X \rightarrow X \text{ and } X$

Problems

- Context-Free Grammars can represent many parts of natural language adequately
- Here are some of the problems that are difficult to represent in a CFG:
 - Agreement
 - Subcategorization
 - Movement (for want of a better term)

Agreement

- This dog
 - Those dogs
 - This dog eats
 - Those dogs eat
 - In English,
 - subjects and verbs have to agree in person and number
 - Determiners and nouns have to agree in number
 - Many languages have agreement systems that are far more complex than this.
 - Solution can be either to add rules with agreement or to have a layer on the grammar called the features
- *This dogs
 - *Those dog
 - *This dog eat
 - *Those dogs eats

Subcategorization

- Subcategorization expresses the constraints that a particular verb (sometimes called the predicate) places on the number and syntactic types of arguments it wants to take (occur with).
 - Sneeze: John sneezed
 - Find: Please find [a flight to NY]_{NP}
 - Give: Give [me]_{NP}[a cheaper fare]_{NP}
 - Help: Can you help [me]_{NP}[with a flight]_{PP}
 - Prefer: I prefer [to leave earlier]_{TO-VP}
 - Told: I was told [United has a flight]_S
 - ...

Subcategorization

- Should these be correct?
 - John sneezed the book
 - I prefer United has a flight
 - Give with a flight
- The various rules for VPs *overgenerate*.
 - They permit the presence of strings containing verbs and arguments that don't go together
 - For example $VP \rightarrow V NP$ therefore
Sneezed the book is a VP since “sneeze” is a verb and “the book” is a valid NP
- Now *overgeneration* is a problem for a generative approach.
 - The grammar should represent **all and only** the strings in a language
- From a practical point of view... Not so clear that there's a problem

Movement

- Consider the verb “booked” in the following example:

– [[My travel agent]_{NP} [booked [the flight]_{NP}]_{VP}]_S



- i.e. “book” is a straightforward transitive verb. It expects a single NP arg within the VP as an argument, and a single NP arg as the subject.

Movement

- But what about?
 - Which flight do you want me to have the travel agent book?
- The direct object argument to “book” isn’t appearing in the right place. It is in fact a long way from where its supposed to appear.
- And note that its separated from its verb by 2 other verbs.
- In Penn Treebank, these types of movement are annotated by have an empty Trace constituent appear in the right place.

The Point about CFGs

- CFGs appear to be just about what we need to account for a lot of basic syntactic structure in English.
- But there are problems
 - That can be dealt with adequately, although not elegantly, by staying within the CFG framework.
- There are simpler, more elegant, solutions that take us out of the CFG framework (beyond its formal power)
 - For example, Feature Structures for CFGs place constraints on how the rules can be applied

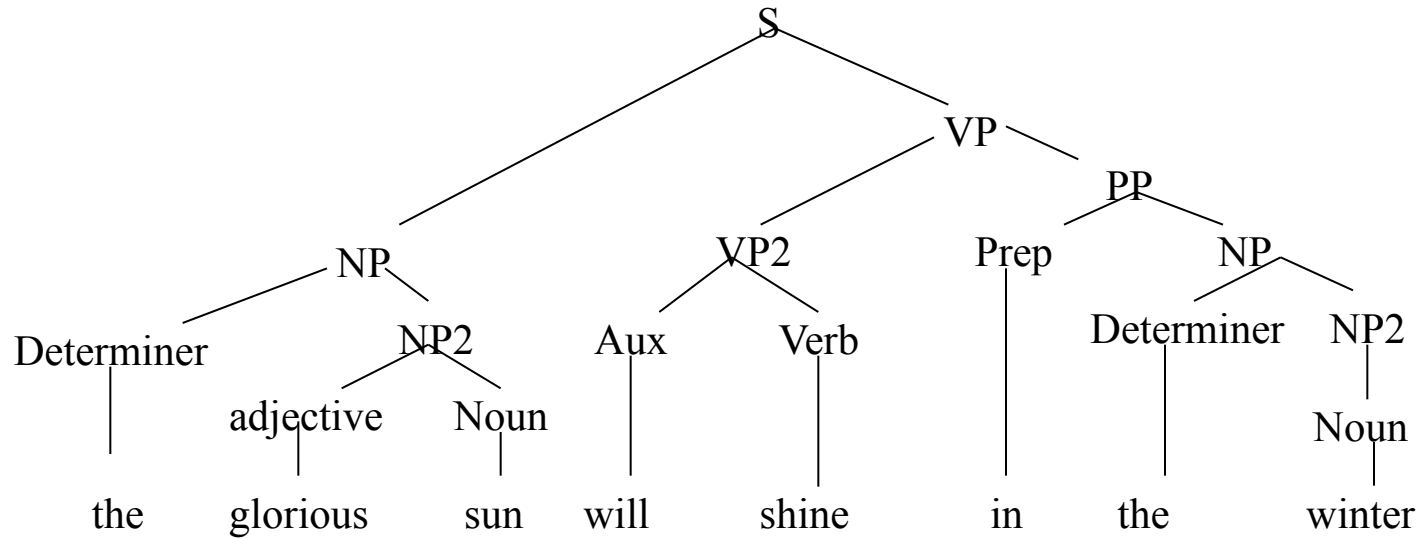
In-class exercise

Dependency Grammars

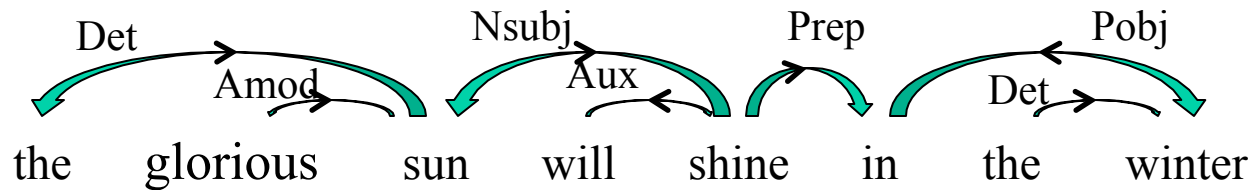
- Dependency grammars offer a different way to represent syntactic structure
 - CGS represent constituents in a parse tree that can derive the words of a sentence
 - Dependency grammars represent syntactic dependency relations between words that show the syntactic structure
 - Typed dependency grammars label those relations as to what the syntactic structure is
- Syntactic structure is the set of relations between a word (aka the head word) and its dependents.

Examples

- Context Free Grammar Tree Structure



- Dependency Relation Structure

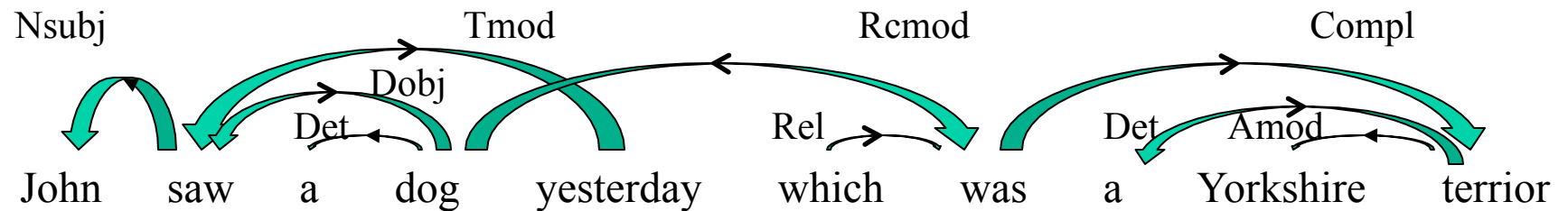


Dependency Relations

- The set of Grammar Relations has varied in number
 - 48 in the Stanford dependency parser
 - 59 in Minipar, a dependency parser from Dekang Lin
 - 106 in Link, a related link grammar parser from CMU
- The examples on the previous page used those from the Stanford dependency parser
 - De Marneffe, MacCartney and Manning, Generating Typed Dependency Parses from Phrase Structure Parses, LREC (Language Resources and Evaluation Conference), 2006.

Projective vs. Non-Projective

- In the dependency graph as depicted in the previous example, with the words in sentence order, if no arcs cross, then it is a projective tree
- If there are crossing arcs, then it is a non-projective tree



- CoNLL (Conference on Natural Language Learning) 2006 had dependency parsing as the shared task on 13 languages, not including English. Out of the languages which had non-projective sentences in the treebanks, from .5% to 5% of the sentences were non-projective.