Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

CZ.1.07/2.2.00/28.0041
Centrum interaktivních a multimediálních studijních opor pro inovaci výuky a efektivní učení

evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
UNIVERSITAS MASARYKIANA BRUNENSIS

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Course objective

Introducing

- theoretical backgrounds on parsing
- parsing methods focused on syntax
- practical implementation methods
- possible applications and evaluations

Course syllabus

PART I : Theoretical backgrounds

- Historical overview
- State of the art parsing methods and trends
- Advanced syntactic formalisms

PART II : Practical applications

- Applications & Use Cases
- Practical Implementations
- Parsing Evaluation

Course format

- Weekly lectures (2 hours)
- Final written exam
- Two homework assignments
- Grading
  - Final exam: 60 points
  - Each homework: 20 points
  - For each homework 10 % top scoring individuals receive 5 bonus points
  - Points required for colloquium: 60 points

Introductive and Historical Overview on Natural Languages Parsing

IA161
Syntactic Formalisms for
Parsing Natural Languages

Main points

- Introduction to Natural Language Processing
- Issues in Syntax
- What is a parsing?
- Overview of Parsing methods and trends

Why natural language processing ?

- Huge amounts of data from Internet and Intranet
- Applications for processing large amounts of texts need NLP expertise
  - Classify text into categories
  - Index and search large texts
  - Automatic translation
  - Speech recognition
  - Information extraction
  - Automatic summarization
  - Question answering
  - Knowledge acquisition
  - Text generation/dialogues

History of Natural Language Processing

- 1948 – 1st NLP application?

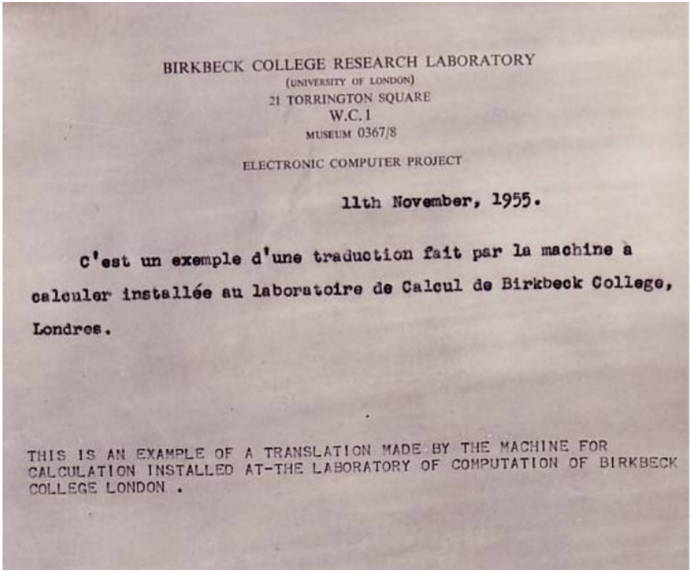
  - dictionary look-up system by Andrew Booth, for machine translation purposes
  - developed at Birkbeck College, London University

- *So far, it turns out, they have not considered at all the problem of multiple meaning (!), and have been concerned only with the mechanics of looking up words in a dictionary. First, you sense the first letter of a word, and then have the machine see whether or not the memory contains precisely the word in question. If so, the machine simply produces the translation (...) of this word. If this exact word is not contained in the memory, then the machine discards the last letter of the word, and tries over. If this fails, it discards another letter, and tries again. After it has found the largest initial combination of letters which is in the dictionary, it "looks up" the whole discarded portion in a special "grammatical annex" of the dictionary. Thus confronted by "running," it might find "run" and then find out what the ending (n)ing does to "run."* (Warren Weaver on Booth's machine)

- This first application shows how closely NLP stands to the origins of computer science.
- Booth was formerly (during WWII) doing research on X-ray crystallography of explosives. This involved lots of arithmetics, hence after WWII he tried to develop electronic computers, the first was an Automatic Relay Calculator (ARC) – 1946.
- In the same year he was funded by Rockefeller Foundation (RF) to visit US researchers, reported that only von Neumann gave him any time. He got in love with (and later married) von Neumann's research assistant Kathleen and redesigned ARC according to von Neumann's architecture.
- In 1947 he visited RF's Natural Sciences Division Director Warren Weaver, who refused to fund a computer for mathematical calculations, but suggested funding a computer for machine translation of natural languages (!).
- Booth developed techniques for parsing text and also for building dictionaries. November 11, 1955 Booth gave an early public demonstration of natural language machine translation (in Figure).

- Later on Booth was very successful in building computers, his wife Kathleen was programming them and wrote one of the first books on programming.
- 1958 Kathleen did research on simulating neural networks to investigate ways in which animals recognise patterns, 1959 then a neural network for character recognition.

History of Natural Language Processing



BIRKBECK COLLEGE RESEARCH LABORATORY
(UNIVERSITY OF LONDON)
21 TORRINGTON SQUARE
W.C.1
MUSEUM 0367/8

ELECTRONIC COMPUTER PROJECT

11th November, 1955.

C'est un exemple d'une traduction fait par la machine a calculer installée au laboratoire de Calcul de Birkbeck College, Londres.

THIS IS AN EXAMPLE OF A TRANSLATION MADE BY THE MACHINE FOR CALCULATION INSTALLED AT-THE LABORATORY OF COMPUTATION OF BIRKBECK COLLEGE LONDON .

History of Natural Language Processing

- 1949 – Warren Weaver

  - Natural Sciences Division Director in the Rockefeller Foundation
  - Mathematician, Science Advocate
  - WWII code breaker
  - He viewed Russian as English in code – the ”Translation” memorandum

Also knowing nothing official about, but having guessed and inferred considerable about powerful new mechanized methods in cryptography – methods which I believe succeed even when one does not know what language has been coded – one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say “This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.”

- Weaver was one of the Machine Translation pioneers and one of the most important science managers at the time. All the time he was meeting scientists, putting them together, organizing funding, and investigating potential research areas; while being a top-scientist – in 1949 he co-authored the The Mathematical Theory of Communication with Claude Shannon.

History of Natural Language Processing

- 1966 – Over-promised under-delivered
  - Machine Translation worked only word by word
  - NLP brought the first hostility of research funding agencies
    - NLP gave AI a bad name before AI had a name.
  - All funding of NLP came to a grinding halt due to the infamous ALPAC report.
    - Public spent 20 million with very limited outcomes.
- 1966–1976 – “A lost decade”
- Revival in 1980’s
  - Martin Kay: The Proper Place of Men and Machines in Language Translation

- ALPAC (Automatic Language Processing Advisory Committee) was a committee of seven scientists led by John R. Pierce, established in 1964 by the U. S. Government in order to evaluate the progress in computational linguistics in general and machine translation in particular. Its report, issued in 1966, gained notoriety for being very skeptical of research done in machine translation so far, and emphasizing the need for basic research in computational linguistics; this eventually caused the U. S. Government to reduce its funding of the topic dramatically.

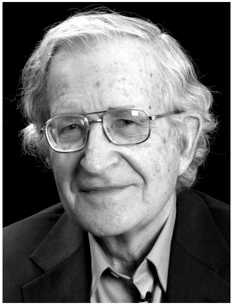
- ALPAC's final recommendations were that research should be supported on:
- 1. practical methods for evaluation of translations;
- 2. means for speeding up the human translation process;
- 3. evaluation of quality and cost of various sources of translations;
- 4. investigation of the utilization of translations, to guard against production of translations that are never read;
- 5. study of delays in the over-all translation process, and means for eliminating them, both in journals and in individual items;
- 6. evaluation of the relative speed and cost of various sorts of machine-aided translation;
- 7. adaptation of existing mechanized editing and production processes in translation;
- 8. the over-all translation process; and
- 9. production of adequate reference works for the translator, including the adaptation of glossaries that now exist primarily for automatic dictionary look-up in machine translation

- Kay’s counterargument: “The goal of MT should not be the fully automatic high quality translation (FAHQT) that can replace human translators. Instead, MT should adopt less ambitious goals, e.g. more cost-effective human-machine interaction and aim at enhancement of human translation productivity.”

NLP looked to Linguistics

Linguistics is language described, not prescribed.
Linguistics had few applicable theories for Machine Translation

- 1957 – Noam Chomsky's Syntactic Structures revolutionized Linguistics as it applies to Machine Translation.



  - Rule based system of syntactic structures.
  - Believed there are features common to all languages that enable people to speak creatively and freely.
  - Hypothesized all children go through the same stages of language development regardless of the language they are learning – a concept of an innate Universal Grammar (never proven)
  - One of the most prominent persons of NLP in 20th century, though very controversial.

- Avram Noam Chomsky (born December 7, 1928) is an American linguist, philosopher, cognitive scientist, logician, and political commentator and activist. Working for most of his life at the Massachusetts Institute of Technology (MIT), where he is currently Professor Emeritus, he has authored over 100 books on various subjects.
- He is credited as the creator or co-creator of the Chomsky hierarchy, the universal grammar theory, and the Chomsky–Schützenberger theorem. Chomsky is also well known as a political activist, and a leading critic of U.S. foreign policy, state capitalism, and the mainstream news media. Ideologically, he aligns himself with anarcho-syndicalism and libertarian socialism.
- Highly influential, between 1980 and 1992, Chomsky was cited within the field of Arts and Humanities more often than any other living scholar, and eighth overall within the Arts and Humanities Citation Index during the same period. He has been described as a prominent cultural figure, and was voted the ”world’s top public intellectual” in a 2005 poll.

NLP looked to Linguistics

- 1958 – Bar-Hillel report
  - Concluded Fully-Automatic High-Quality Translation (FAHQT) could not be accomplished without human knowledge.
- 1968 – Case Grammar (Fillmore)
  - “The case for case” paper
  - Later evolved into Frame Semantics
- 1970 – Augmented Transition Networks (Woods)
  - Procedural Semantics – Theory of the “meaning” of sentence.
  - Augmented Transition Network (ATN) parser

- Yehoshua Bar-Hillel was charged by the US Office of Naval Research to make a critical assessment of current MT activity in the United States and Great Britain. Agencies of the US government were funding research on a large and increasing scale.
- Bar-Hillel had been the first person appointed full-time to work on MT at the Massachusetts Institute of Technology in 1951.
- His basic contention was that MT research – now a “multimillion dollar affair”, as he pointed out – was, with few exceptions, set on a mistaken and unattainable goal, namely, fully automatic translation of a quality equal to that of a good human translator. This he held to be utterly unrealistic, and in his view resources were being wasted which could be more fruitfully be devoted to the development of less ambitious and more practical computer aids for translators.
- Woods: 2010 ACL Lifetime Achievement Award
- all within the LUNAR project of NASA Apollo 11 Program
- ATN parser was the first properly formalized hand-built parser that could produce Chomsky-style deep structure analyses for a large, fluent subset of natural English.

Successful NLP Systems

- 1970 – Shrdlu (Terry Winograd)

Human: *Pick up a big red block.*
Computer: OK.
Human: *Grasp the pyramid.*
Computer: I don't understand which pyramid you mean.
Human (changing their mind): *Find a block which is taller than the one you are holding and put it into the box.*
Computer: By ”it”, I assume you mean the block which is taller than the one I am holding.
Computer: OK.
Human: *What does the box contain?*
Computer: The blue pyramid and the blue block.
Human: *What is the pyramid supported by?*
Computer: The box.
Human: *How many blocks are not in the box?*
Computer: Four of them.
Human: *Is at least one of them narrower than the one which I told you to pick up?*
Computer: Yes, the red cube.

- SHRDLU was an early natural language understanding computer program, developed by Terry Winograd at MIT in 1968–1970. In it, the user carries on a conversation with the computer, moving objects, naming collections and querying the state of a simplified ”blocks world”, essentially a virtual box filled with different blocks. SHRDLU was written in the Micro Planner and Lisp programming language on the DEC PDP-6 computer and a DEC graphics terminal. Later additions were made at the computer graphics labs at the University of Utah, adding a full 3D rendering of SHRDLU's ”world”.
- The name SHRDLU was derived from ETAOIN SHRDLU, the arrangement of the alpha keys on a Linotype machine, arranged in descending order of usage frequency in English.

Successful NLP Systems II

- 1973 – Lunar question answering system (Woods)

```
WHAT IS THE AVERAGE CONCENTRATION OF ALUMINUM IN HIGH
ALKALI ROCKS?
WHAT SAMPLES CONTAIN P200?
GIVE ME THE MODAL ANALYSES OF P200 IN THOSE SAMPLES
GIVE ME EU DETERMINATIONS IN SAMPLES WHICH CONTAIN ILM
```

- LUNAR is an experimental natural language, information retrieval system. It was designed to help geologists access, compare, and evaluate chemical-analysis data on moon rock and soil composition obtained from the Apollo-11 mission. The primary goal of the designers was research on the problems involved in building a man-machine interface that would allow communicate in ordinary English, A "real world" application was chosen for two reasons: First, it tends to focus effort on the problems really in need of solution (sometimes this is implicitly avoided in "toy" problems) and second, the possibility of producing a system capable of performing a worthwhile task.
- LUNAR system operates by translating a question entered in English into an expression in a formal query language (Codd, 1974). The translation is done with an augmented transition network (ATN) parser coupled with a rule-driven semantic interpretation procedure, which guides the analysis of the question.

- The "query" that results from this analysis is then applied to the database to produce the answer to the request,The query language is a generalization of the predicate calculus. Its central feature is a quantifier function that is able to express, in a simple manner, the restrictions placed on a database-retrieval request by the user. The function is used in concert with special enumeration functions for classes of database objects, freeing the quantifier function from explicit dependence on the structure of the database. LUNAR also served as a foundation for the early work on speech understanding at BBN.
- The formal query language used by LUNAR system contains three types of objects: designators, which name classes of objects in the database (including functionally defined objects); propositions, which are formed from predicates with designators as arguments; and commands, which initiates actions.
- Request: (DO MY SAMPLES HAVE GREATER THAN 13 PERCENT ALUMINIUM
  Query Language Translation (after parsing):
  (TEST (FOR SOME X1 / (SEQ SAMPLES) : T ; (CONTAIN X1 NPR* X2 / 'AL203) (GREATERTHAN 13 PCT))))
  Response :
  YES

- LUNAR processes these request in the following three steps:
  1. Syntactic analysis using an augmented transition network parser and heuristic information (including semantics) to produce the most likely derivation tree for the request;
  2. Semantic interpretation to produce a representation of the meaning of the request in a formal query language;
  3. Execution of the query language expression on the database to produce the answer to the request.
- LUNAR's language processor contains an ATN grammar for a large subset of English, the semantic rules for interpreting database requests, and a dictionary of approximately 3,500 words. As an indication of the capabilities of the processor, it is able to deal with tense and modality, some anaphoric references and comparatives, restrictive relative clauses, certain adjective modifiers and embedded complement constructions.

Successful NLP Systems III

- 1976 – TAUM-METEO (University of Montreal)
  - prototype MT system for translating weather forecasts between English and French
- 1985 – METEO (John Chandioux)
  - successor of TAUM-METEO
  - in operational use at Environnement Canada forecasts until 30th of September 2001
- 1970 – SYSTRAN
  - provided translations for US Air Force’s Foreign Technology Division
  - adopted by XEROX (1978)
  - still developed, present in wide range of systems
- Google language tools
- Microsoft spell check

- The METEO System is a Very High Quality Machine Translation system for weather bulletins that has been in operational use at Environnement Canada from 1982 to 2001. It stems from a prototype developed in 1975-76 by the TAUM Group, known as TAUM-METEO. As many authors confuse the prototype with the actual system, a bit of history is in order.
- The initial motivation to develop that prototype was that a junior translator came to TAUM to ask for help in doing the extremely boring (and at the same time difficult) job of translating weather bulletins at Environment Canada he had to do at the moment.
- Indeed, since all official communications emanating from the Canadian government must be available in French and English, because of the official bilingual services act of 1968, and weather bulletins represent a large amount of translation in real time, junior translators had to spend several months of purgatory producing first draft translations, then revised by seniors. That was in fact a quite difficult job, because of the specificities of the English and French sublanguages used, and not very motivating, as the lifetime of a bulletin is only 4 hours.

Major Issues in NLP

Ambiguity in Language:

- Syntactic (structural)
- Semantic (word sense)
- Referential

Ambiguity Makes NLP difficult

- Structural/Syntactic ambiguity
  - I saw the Grand Canyon flying to New York.
  - I saw the sheep grazing in the field.
- Word Sense ambiguity
  - The man went to the bank to get some cash.
  - The man went to the bank and jumped in the river.
- Referential ambiguity
  - Steve hated Paul. He hit him.
  - He = Steve ? or he = Paul ?

Linguistics levels of analysis

- Speech
- Written language
  - Phonetics
  - Phonology
  - Morphology
  - Syntax
  - Semantics
  - Beyond: pragmatic, cognitive, logic...

Each level has an input and output representation, output from one level is the input to the next, sometimes levels might be skipped (merged) or split.

Issues in syntax

- Propagation of errors from lower levels – mainly morphology, need to correctly identify the part of speech (POS)
  "The man did his homework"
  - <u>Who</u> did <u>what</u>?
    man=noun; *did*=verb; *his*=genitive; *homework*=noun
- Identify collocations
  - *Mother in law, hot dog, ...*

More issues in Syntax

- Anaphora resolution
  "The <u>son</u> of <u>my professor</u> entered my class. <u>He</u> scared me."
- Preposition attachment
  "I saw the man <u>in the park</u> <u>with a telescope</u>."

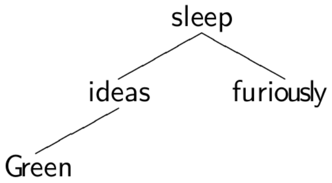
Syntax input and output

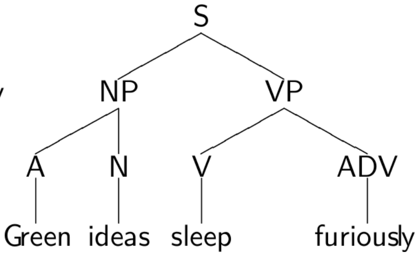
- **Input:** sequence of pairs (lemma, (morphological) tag)
- **Output:** sentence structure (tree) with annotated nodes (all lemmas, (morpho-syntactic tags, functions) of various forms
- Deals with:
  - The relation between lemmas & morphological categories and the sentence structure use syntactic categories such as subject, verb, object,...

Syntactic representation

- Tree structure
- Two main ideas for the tree
  - **Phrase structure** (derivation tree)
    - Using bracketed grouping
    - Brackets annotated by phrase type
    - Heads (often) explicitly marked
  - **Dependency structure**
    - Basic relation: head (governor) – dependent
    - Links annotated by syntactic functions
    - Phrase structure implicitly present

Dependency Tree vs. PS Tree





Shallow parsing

“the man chased the bear”

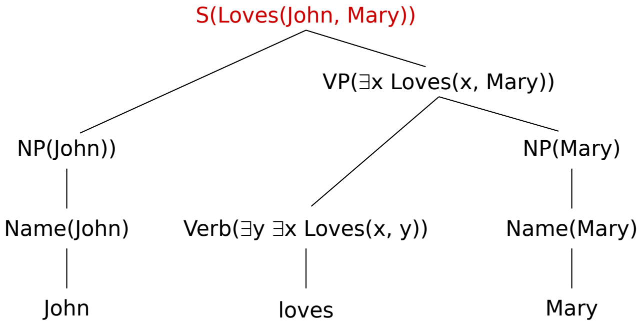
“the man” “chased the bear”

Subject -- Predicate

- Identify basic structures
  - NP-[the man] VP-[chased the bear]

Full parsing

“John loves Mary“



Help figuring out automatically questions *like who did what and when?*

What is a natural language parsing ?

- One of the most commonly researched tasks in Natural Language Processing (NLP)
    - Parsing, in traditional sense, is what happens when a student takes the words of a sentences one by one, assigns each to a part of speech, specifies its grammatical categories, and lists the grammatical relations between words (identifying subject and various types of object for a verb, specifying the word with which some other word agrees, and so on).

Characteristics of parsing

Much of the history of parsing until a few decades ago can be understood as the direct consequence of the history of theories of grammar:

- Parsing is done by human beings, rather than by physical machines or abstract machine
- What is parsed is a bit of natural language, rather than a bit of some language-like symbolic system
- Parsing is heuristic rather than algorithmic

New notions of parsing

In the second half of 20^{th} century the parsing has come to be extended to a large collection of operations in relation with theoretical linguistics, formal language theory, computer science, artificial intelligence and psycholinguistics:

- Parsing is the syntactic analysis of languages.
- The objective of Natural Language Parsing is
  - to determine parts of sentences (such as verbs, noun phrases, or relative clauses), and the relationships between them (such as subject or object).
- Unlike parsing of formally defined artificial languages (such as Java or predicate logic), parsing of natural languages presents problems due to ambiguity, and the productive and creative use of language.

Parsing

- The grammar for Natural Language is ambiguous and typical sentences have multiple possible analyses (syntactically and semantically).
- Some parsing tools (i.e. grammatical, morphologic, syntactic, statistic, probabilistic, heuristic, ...) help to find the most plausible parse tree of a given sentence.

Practical function of a parsing

- Parsing can tell us when a sentence is in a language defined by a grammar
- Parsing makes the extraction of the information possible by identifying relations between words, or phrases in sentences.

Practical function of a parsing

- Parsers are being used in a number of disciplines:
  - In computer science
    - Compiler construction, database interfaces, self-describing databases, artificial intelligence...
  - In linguistics
    - Text analysis, corpora analysis, machine translation...
  - In document preparation and conversion
  - In typesetting chemical formulae
  - In chromosome recognition

Practical function of a parsing

- However,
  - Many different possible syntactic formalisms:
    - Regular expressions, Context-free grammars, Context-sensitive grammars, ...
  - Many different ways of representing the results of parsing:
    - Parse tree, Chart, Graph, ...

Historical overview of parsing methods

- Basically two ways to parse a sentence
    - **Top-down** vs. **Bottom-up**

      We can characterize the search strategy of parsing algorithms in terms of the **direction** in which a structure is built:
      from the words upwards (bottom-up) or
      from the root node downwards (top-down)

Historical overview of parsing methods

- **Directionality** in these two ways

Directional vs. Non-directional

- Non-directional top-down methods by S. Unger (1968)
- Non-directional bottom-up methods by CYK
- Directional top-down methods:
  - The predict/match automaton, Depth-first search (backtrack), Breadth-first search (Greibach), Recursive descent, Definite Clause grammars
- Directional bottom-up methods:
  - The shift/reduce automaton, Depth-first search (backtrack), Breadth-first search, restricted by Earley(1970)

Historical overview of parsing methods

- Methods originating at parsing of formal languages
  - Linear directional top-down methods:
    - LL(K)
  - Linear directional bottom-up methods:
    - Precedence, bounded-context, LR (k), LALR(1), SLR(1)
- Methods specifically devised for parsing of natural languages
  - Generalized LR (Masaru Tomita)
  - Chart parsing (Martin Kay)

Summary

- Natural language parsing as one of the NLP domain
- Extended notion of parsing in relation with different fields
- Ambiguity of language
- What is it to “parse”?
- Overview of basic parsing methods

References I

- H. Bunt, J. Carroll & G. Satta (eds.): *New Developments in Parsing Technology*, Kluwer, Dordrecht/Boston/London 2004
- H. Bunt, P. Merlo, & J. Nivre (eds.): *Trends in Parsing Technology: Dependency Parsing, Domain Adaptation, and Deep Parsing*, Springer Dordrecht, Heidelberg/London/New York 2010
- H. Bunt, M. Tamita (eds.): *Recent advances in parsing technology*, Kluwer, Boston, 1996
- G. Dick: *Parsing techniques: a practical guide*, Springer, 2008
- Roger G. Johnson: *Andrew D. Booth – Britain's Other "Fourth Man". In: History of Computing. Learning from the Past*, Springer Berlin Heidelberg, 2010.
- J. Hutchins: *From First Conception to First Demonstration: the Nascent Years of Machine Translation, 1947–1954. A Chronology. In: Machine Translation, Volume 12, Issue 3*, Kluwer, 1997.

References II

- J. Hutchins: *Milestones no.6: Bar-Hillel and the nonfeasibility of FAHQT. In: International Journal of Language and Documentation no.1*, 1999.
- M. Kay: *The proper place of men and machines in language translation. In: Machine Translation, Volume 12, Issue 1–2*, Kluwer 1997 (reprint of 1980).
- More on history of MT: `http://www.hutchinsweb.me.uk/history.htm`

Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

Basic parsing methods

Main points

- Context-free grammar
- Parsing methods
  - Top-down or bottom-up
  - Directional or non-directional
- Basic parsing algorithms
  - Unger
  - CKY (or CYK)
  - Left-corner parsing
  - Earley

Ambiguity in Natural Language

- Notion of ambiguity
  - Essential ambiguity: same syntactic structure but the semantics differ
  - Spurious ambiguity: different syntactic structure but no change in semantics

There is no unambiguous languages!

- An input may have exponentially many parses
- Should identify the “correct” parse

Ambiguity in Natural Language

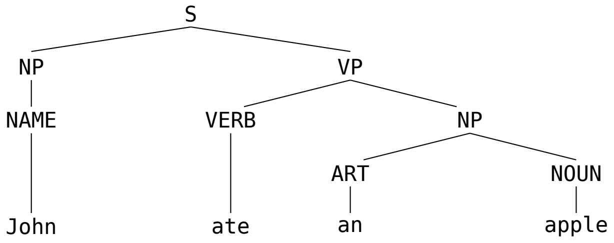
- Main idea of parsing

Parsing (syntactic structure)

Input: sequence of tokens

`John ate an apple`

Output: `parse tree`



Ambiguity in Natural Language

- Basic connection between a sentence and the grammar it derives from is the “parse tree”, which describes how the grammar was used to produce the sentences.
- For the reconstruction of this connection we need a “parsing techniques”

Ambiguity in Natural Language

- **Word categories**: Traditional parts of speech

Noun	Names of things	boy, cat, truth
Verb	Action or state	become, hit
Pronoun	Used for noun	I, you, we
Adverb	Modifies V, Adj, Adv	sadly, very
Adjective	Modifies noun	happy, clever
Conjunction	Joins things	and, but, while
Preposition	Relation of N	to, from, into
Interjection	An outcry	ouch, oh, alas, psst

Formal language

- Symbolic string set which describe infinitely unlimited language as mathematical tool for recognizing and generating languages.
- Topic of formal language: finding finitely infinite languages using rewriting system.
- Three basic components of formal language: finite symbol set, finite string set, finite formal rule set

Constituency

- Sentences have parts, some of which appear to have subparts. These groupings of words that go together we will call constituents.

(How do we know they go together?)

- I hit the man with a cleaver
  I hit [the man with a cleaver]
  I hit [the man] with a cleaver

- You could not go to her party
  You [could not] go to her party
  You could [not go] to her party

The Chomsky hierarchy

- Type 0 Languages / Grammars (LRE: Recursively enumerable grammar)
  Rewrite rules $\alpha \rightarrow \beta$
  where α and β are any string of terminals and non-terminals

- Type 1 Context-sensitive Languages / Grammars (LCS)
  Rewrite rules $\alpha X \beta \rightarrow \alpha \Upsilon \beta$
  where X is a non-terminal, and α, Υ, β are any string of terminals and non-terminals, (Υ must be non-empty but strings α and β can be empty).

- Type 2 Context-free Languages / Grammars (LCF)
  Rewrite rules $X \rightarrow \Upsilon$
  where X is a non-terminal and Υ is any string of terminals and non-terminals

- Type 3 Regular Languages / Grammars (LREG)
  Rewrite rules $X \rightarrow \alpha Y$
  where X, Y are single non-terminals, and α is a string of terminals; Y might be missing.

The Chomsky hierarchy

Type 0 > 1 > 2 > 3

according to generative power

- Superior language can generate inferior language but superior language is more inefficient and slow than inferior language.

The Chomsky hierarchy

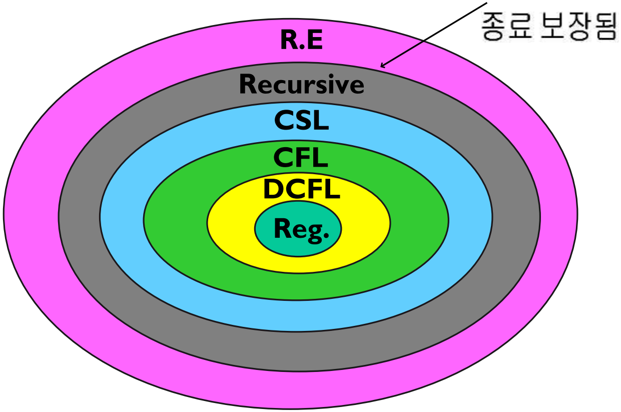


Figure : Chomsky hierarchy

Context-free grammar (Type 2)

The most common way of modeling constituency.

The idea of basing a grammar on constituent structure dates back to Wilhem Wundt (1890), but not formalized until Chomsky (1956), and, independently, by Backus (1959).

CFG = Context-Free Grammar = Phrase Structure Grammar= BNF = Backus-Naur Form

Context-free grammar (Type 2)

- CFG rewriting rule

$$X \rightarrow \Upsilon$$

where X is a non-terminal symbol and ϒ is string consisting of terminals/non-terminals.

The term “Context-free” expresses the fact that the non-terminal *v* can always be replaced by *w*, regardless of the context in which it occurs.

Context-free grammar (Type 2)

G = < T, N, S, R>

T is set of terminals (lexicon)

N is set of non-terminals (written in capital letter). S is start symbol (one of the non-terminals)

R is rules/productions of the form X $\rightarrow \Upsilon$, where X is a non-terminal and Υ is a sequence of terminals and non-terminals (may be empty).

A grammar G generates a language L

Example1 of Context-Free Grammar

G = < T, N, S, R>

T = { that, this, a, the, man, book, flight, meal, include, read, does }

N = { S, NP, NOM, VP, DET ,N, V, AUX }

S = S

R = {

S → NP VP
S → Aux NP VP
S → VP
NP → Det NOM
NP → N
VP → V
VP → V NP
}

Det → that | this | a | the
N → book | flight | meal | man
V → book | include | read
AUX → does

Example2 of Context-Free Grammar

R1: S -> NP VP
R2: NP -> DET N
R3: NP -> NP PNP
R4: NP -> PN
R5: VP -> V
R6: VP -> V NP
R7: VP -> V PNP
R8: VP -> V NP PNP
R9: VP -> V PNP PNP
R10: PNP -> PP NP
R11: PP-> to|from|of
R12: DET -> an|a
R13: DET -> his|her
R14: DET -> the
R15: V -> eat|serve
R16: V -> give
R17: V -> speak|speaks
R18: V -> discuss
R19: PN -> John|Mark
R20: PN -> Mary|Juliette
R21: N -> daugther|mother
R22: N -> son|boy
R23: N -> salad|soup|meat
R24: N -> desert|cheese|bread
R25: ADJ -> small|kind

Simplified example of $CFG = G_D$

Example2 of Context-Free Grammar

Using the presented grammar, we make a first derivation for the sentence *"John speaks"*,

S -> G_D NP VP (by R1)
S -> G_D PN VP (by **R4**)
-> G_D John VP (by R19)
-> G_D John V (by **R5**)
-> G_D John speaks (by R17)

Example2 of Context-Free Grammar

Another derivation of *"John speaks"* from G_D using rule 5 before rule 4

```
S   ->  G_D NP VP
S   ->  G_D NP V
    ->  G_D NP speaks
    ->  G_D PN speaks
    ->  G_D John speaks
```

Production Rule 3

NP -> NP PNP

Because it contains the same symbol in his left and his right, we say that the production having this property is recursive.

Production Rule 3

This property of $R3$ involves that the language generated by the grammar G_D is infinite, because we can create the sentences of arbitrary length by iterative application of $R3$.

Test

NP -> G_D NP PNP -> G_D NP PNP PNP -> G_D NP PNP PNP PNP....

- The son of John speaks
- The son of the mother of John speaks
- The son of the daughter of the daughter ....of John speaks.

Production Rule 3

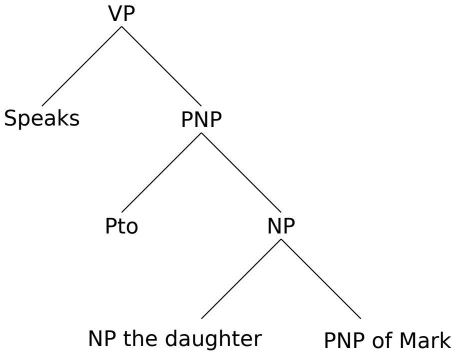
- Last remark concerning this grammar (G_D)

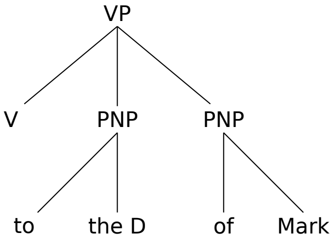
This grammar can generate sentences which are **ambiguous**.
"John speaks to the daughter of Mark"

Example

1. A conversation between **John** and **the daughter of Mark** (R7)
2. A conversation **about Mark** between **John** and **the daughter** (R9)

Production Rule 3





Commonly used non-terminal abbreviations

S	sentence
NP	noun phrase
PP	prepositional phrase
VP	verb phrase
*X*P	*X* phrase
N	noun
PREP	preposition
V	verb
DET/ART	determiner / article
ADJ	adjective
ADV	adverb
AUX	auxiliary verb
PN	proper noun

Parsing methods

- Classification of parsing methods

  Top-down parsing vs. **Bottom-up** parsing

  - Directional vs. non-directional parsing

Top-down or bottom-up

- Top-down parsing
  - The sentence from the start symbol, the production tree is reconstructed from the top downwards
  - Identify the production rules in prefix order
  - Never explores a tree that cannot result in an S
  - BUT Wastes time generating trees inconsistent with the input
- Bottom-up parsing
  - The sentence back to the start symbol
  - Identify the production rules in postfix order
  - Never generates trees that are not grounded in the input
  - BUT Wastes time generating trees that do not lead to an S

Top-down parsing

- Top-down parsing is goal-directed.
    - A top-down parser starts with a list of constituents to be built.
    - It rewrites the goals in the goal list by matching one against the LHS of the grammar rules,
    - and expanding it with the RHS,
    - ...attempting to match the sentence to be derived.
- If a goal can be rewritten in several ways, then there is a choice of which rule to apply (search problem)
- Can use depth-first or breadth-first search, and goal ordering.

Top-down parsing

Simulation of the operation of parser in top-down methods

The son speaks

1. S
2. NP VP
3. DET N VP
4. 4. a N VP. Fail: input begin by the. We return to DET N VP
5. the N VP
6. the daughter VP. New fail α=le N VP

   ......
7. the son VP
8. the son V
9. the son speaks.

   ......

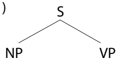
Top-down parsing

Top-down parsing example

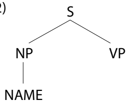
$$
\begin{array}{lll}
\text{S} & \rightarrow & \text{NP VP} \\
 & \rightarrow & \text{NAME VP} \\
 & \rightarrow & \text{“John” VP} \\
 & \rightarrow & \text{“John” VERV NP} \\
 & \rightarrow & \text{“John” “ate” NP} \\
 & \rightarrow & \text{“John” “ate” DET NOUN} \\
 & \rightarrow & \text{“John” “ate” “an” NOUN} \\
 & \rightarrow & \text{“John” “ate” “an” “apple”}
\end{array}
$$

Top-down parsing

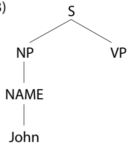
(1)



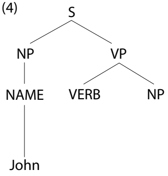
(2)



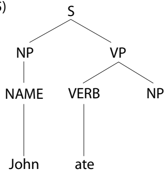
(3)



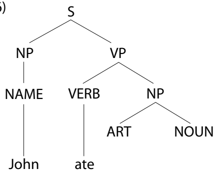
(4)



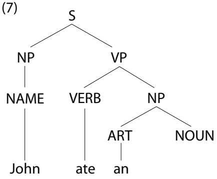
(5)



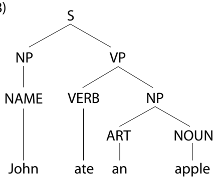
(6)



(7)



(8)



Top-down parsing

Algorithm of top-down left-right (LR) parsing

α is a primal current word, u input to be recognized.

tdlrp = main function
tdlrp (α, u)

```
begin
    if $(\alpha = u)$ then return (true) fi
        A $= u_1 \ldots\ldots u_k A \Upsilon$
    while $(\exists A -> \beta)$ do
        $(\beta = u_{k+1} \ldots\ldots\ldots u_{k+1}{}^{\delta})$ with $\delta = \epsilon$ ou $\delta = A\ldots$
    if $(tdlrp(u_1 \ldots\ldots u_{k+1}{}^{\delta} \Upsilon) = true)$ then return(true) fi
    od
        return (false)
end
```

Top-down parsing

Problems in top-down parsing

- Left recursive rules... e.g. NP → NP PP... lead to infinite recursion
- Will do badly if there are many different rules for the same LHS. Consider if there are 600 rules for S, 599 of which start with NP, but one of which starts with a V, and the sentence starts with a V.
- Top-down parsers do well if there is useful grammar-driven control: search is directed by the grammar.
- Top-down is hopeless for rewriting parts of speech (preterminals) with words (terminals).

Bottom-up parsing

- Bottom-up parsing is data-directed.
  - The initial goal list of a bottom-up parser is the string to be parsed.
  - If a sequence in the goal list matches the RHS of a rule, then this sequence may be replaced by the LHS of the rule.
  - Parsing is finished when the goal list contains just the start symbol.
- If the RHS of several rules match the goal list, then there is a choice of which rule to apply (search problem)
- Can use depth-first or breadth-first search, and goal ordering.

Bottom-up parsing

Let’s suppose that we have a sentence *“the son eats his soup”* in the grammar G_D.

Question

How we can do to verify that the word belong to the language generated by the grammar G_D and if the answer is positive to assign a tree?

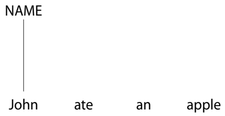
→ The first idea can be given in the following algorithms:

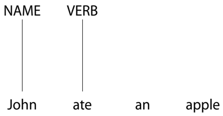
Bottom-up parsing

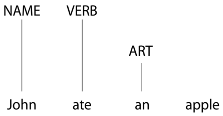
Bottom-up parsing example

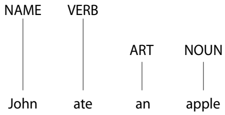
“John”	“ate”	“an”	“apple”
→ NAME	“ate”	“an”	“apple”
→ NAME	VERV	“an”	“apple”
→ NAME	VERV	DET	“apple”
→ NAME	VERV	DET	NOUN
→ NP	VERV	DET	NOUN
→ NP	VERV	NP	
→ NP	VP		
→ S			

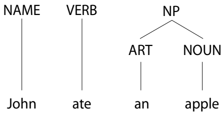
Bottom-up parsing

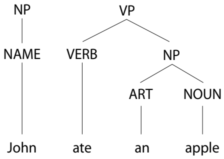
(1)


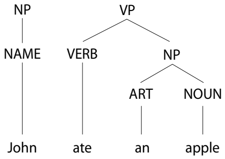
(2)


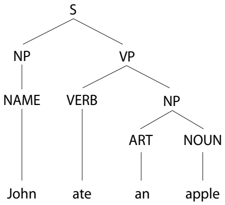
(3)


(4)


(5)


(6)


(7)


(8)


Bottom-up parsing

Problems with bottom-up parsing

- Unable to deal with empty categories: termination problem, unless rewriting empties as constituents is somehow restricted (but then it's generally incomplete)
- Inefficient when there is great lexical ambiguity (grammar-driven control might help here). Conversely, it is data-directed: it attempts to parse the words that are there.
- Both Top-down (LL) and Bottom-up (LR) parsers can (and frequently do) do work exponential in the sentence length on NLP problems.

Left-corner parsing

Left-corner parsing

- Bottom-up with top-down filtering:
    - combine top-down processing with bottom-up processing in order to avoid going wrong in the ways that we are prone to go wrong with pure top-down and pure bottom-up techniques

Left-corner parsing

Going wrong with top-down parsing

```
S -> NP VP
NP -> DET N
NP -> PN
VP -> IV
DET -> the
N -> robber
PN -> Vincent
IV -> died
```

Vincent died.

Left-corner parsing

Going wrong with bottom-up parsing

S -> NP VP
NP -> DET N
VP -> IV
VP -> TV NP
TV -> plant
IV -> died
DET-> the
N -> plant

The plant died.

1	DET plant died	
2	DET TV IV	Fail
3	DET N IV	OK
4	NP VP	OK
5	S	

Left-corner parsing

Combining Top-down and Bottom-up Information

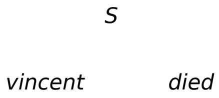
```
S -> NP VP
NP -> DET N
NP -> PN
VP -> IV
DET -> the
N -> robber
PN -> Vincent
IV -> died
```

Vincent died.

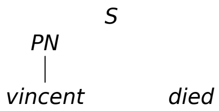
Left-corner parsing

Now, let’s look at how a left-corner recognizer would proceed to recognize Vincent died.

1. Input: Vincent died. Recognize an S. (Top-down prediction.)

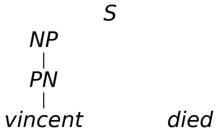


2. The category of the first word of the input is PN. (Bottom-up step using a lexical rule.)

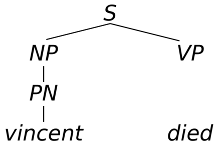


Left-corner parsing

3. Select a rule that has at its left corner : NP-> PN. (Bottom-up step using a phrase structure rule.)

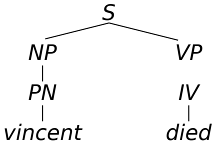


4. Select a rule that has at its left corner: S->NP VP. (Bottom-up step.)
5. Match! The left hand side of the rule matches with S, the category we are trying to recognize.

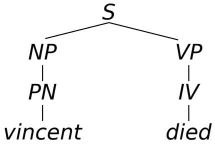


Left-corner parsing

6. Input: died. Recognize a VP. (Top-down prediction.)
7. The category of the first word of the input is IV. (Bottom-up step.)



8. Select a rule that has at its left corner: VP->IV. (Bottom-up step.)
9. Match! The left hand side of the rule matches with VP, the category we are trying to recognize.



Left-corner parsing

- What is a left-corner of a rule:
  - the first symbol on the right hand side. For example, *NP* is the left corner of the rule $S \rightarrow NP\,VP$, and *IV* is the left corner of the rule $VP \rightarrow IV$. Similarly, we can say that *Vincent* is the left corner of the lexical rule $PN \rightarrow$ *Vincent*.

Left-corner parsing

- What is a left-corner of a rule:
  - "Predictive" parser : it uses grammatical knowledge to predict what should come next, given what it has found already.
  - 4 operations creating new items from old:
    "Shift", "Predict", "Match" and "Reduce"

Left-corner parsing

- Definition (Corner relation)
  The relation $\angle$ between non-terminals A and B such that $B \angle A$ if and only if there is a rule $A \rightarrow B\alpha$, where α denotes some sequence of grammar symbols

- Definition (Left corner relation)
  The transitive and reflexive closure of $\angle$ is denoted by $\angle^*$, which is called left-corner relation

Left-corner parsing

Left-corner table

Non Terminal	Left-corners
S	S NP time an VorN files
NP	NP time an VorN files
VP	VP VorN files VorP like
PP	PP VorP like
VorN	VorN files
VorP	VorP like

Grammar

S → NP VP
S → S PP
NP → time
NP → an arrow
NP → VorN
VP → VorN
VP → VorP NP
PP → VorP NP
VorN → files
VorP → like

How to deal with ambiguity?

- Backtracking
  - Try all variants subsequently.
- Determinism
  - Just choose one variant and keep it (i.,e. greedy).
- Parallelism
  - Try all variants in parallel.
- Underspecification
  - Do not desambiguate, keep ambiguity.

Summary

- One view on parsing: parsing as a phrase-structure formal grammar recognition task
- Parsing approaches: top-down, bottom-up, left-corner

Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

`ia161@nlp.fi.muni.cz`

Autumn 2013

Chart parsing

Main points

- CKY algorithm
- Earley parsing
- General chart parsing methods

Directional or non-directional

$\left\{ \begin{array}{l} \text{Directional top-down} \\ \text{Directional bottom-up} \end{array} \right.$

$\left\{ \begin{array}{l} \text{Non-directional top-down method} \\ \quad \text{– firstly by } \textbf{Unger} \\ \text{Non-directional bottom-up} \\ \quad \text{– by Cocke, Younger and Kasami (}\textbf{CYK, also CKY}\text{)} \end{array} \right.$

$\rightarrow$ They access the input in an seemingly arbitrary order, so they require the entire input to be in memory before parsing can start

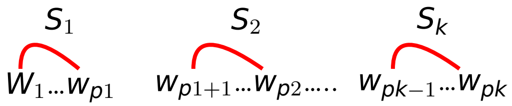
Non-directional top-down methods by Unger

- Capable of working with the entire class of CFG
- Expects as input a sentence and a CFG
- It works by searching for **partitionings of the input** which match the right hand side(RHS) of production rules.

Non-directional top-down methods by Unger

- Let G denote a CF grammar and w be an input sentence.
- **Principle**: if the input sentence w belongs to the language $L(G)$ it must be derivable from the start symbol S of the grammar G.

Let S be defined as: $S \rightarrow S_1 \, S_2 ... Sk$
The input sentence w must be obtainable from the sequence of symbols $S_1 \, S_2 ... Sk$ in a way that S_1 must derive a first part of the input, S_2 a second part, and so on.



Non-directional bottom-up methods as CYK

CYK is an example of chart parsing

- discovered independently by Cocke, Younger and kasami
- Consider which non-terminals can be used to derive substrings of the input, beginning with shorter strings and moving up to longer strings

1. Start with strings of length one, matching the single character in the input strings against unit productions in the grammar
2. Then considers all substrings of length two, looking for production with right-hand side elements that match the two characters of the substring.
3. Continues up to longer strings

Non-directional bottom-up methods as CYK

CYK example 2

Two example sentences and their potential analysis
He [gave[the young cat][to Bill]].
He [gave [the young cat][some milk]].

The corresponding grammar rules:

$$\mathrm{VP} \rightarrow \mathrm{V}_{ditrans} \quad \mathrm{NP} \;\; \mathrm{PP}_{to}$$
$$\mathrm{VP} \rightarrow \mathrm{V}_{ditrans} \quad \mathrm{NP}\;\mathrm{VP}$$

Regardless of the final sentence analysis, the ditransitive verb (gave) and its first object NP (the young cat) will have the same analysis
-> No need to analyze it twice.

Non-directional bottom-up methods as CYK

Solutions: chart parsing

1. Store analyzed constituents: well formed substring table or (passive) chart
2. Partial and complete analyses: (active) chart

In other words, instead of recalculating that the young cat is an NP, we will store that information

- **Dynamic** programming: never go backwards

CKY algorithm

```
program CKY Parser;
begin
for $p$ := 1 to $n$ do $V[p,1]$ := $\{A|A \rightarrow a_p \in P\}$;
for $q$ := 2 to $n$ do
    for $p$ := 1 to $n-q+1$ do
        $V[p,q] = \emptyset$;
        for $k$ :=1 to $q-1$ do
            $V[p,q] =$
                $V[p,q] \cup$
                $\cup \{A|A \rightarrow BC \in P, B \in V[p,k], C \in V[p+k,q-k]\}$;
        od
    od
od
end
```

Complexity of CKY is $O(n^3)$

počítá se (trojúhelníková) matice V:

- sloupce = pozice ve vstupní větě
- řádky = délky (pod)řetězců vstupní věty
- prvky = množiny neterminálů, které pokrývají odpovídající čás
  vstupní věty

první cyklus naplní první řádek matice
ve vnitřním cyklu se B a C vybírají vždy z už hotových políček matice (menší řetězce) – tj. od 2. řádku už vůbec nepracujeme se vstupní větou, jen s předchozími řádky

neznámé terminály na vstupu se *ignorují*

CKY example

- input grammar:

Definition

$S \rightarrow AA|BB|AX|BY|a|b$
$X \rightarrow SA$
$Y \rightarrow SB$
$A \rightarrow a$
$B \rightarrow b$

- input string $w = abaaba$.

CKY example – solution

a b a a b a

Definition

$S \rightarrow AA|BB|AX|BY|a|b$
$X \rightarrow SA$
$Y \rightarrow SB$
$A \rightarrow a$
$B \rightarrow b$

p – position, q – length

q \ p	1	2	3	4	5	6
1	S, A	S, B	S, A	S, A	S, B	S, A
2	Y	X	S, X	Y	X	
3	S	$\emptyset$	Y	S		
4	X	S	$\emptyset$			
5	$\emptyset$	X				
6	S					

nechat počítat na tabuli studenty – políčka v prvních řádcích jdou rychleji
napsat na tabuli prázdnou matici V a do ní doplňovat.
postup: např. 2. řádek, políčko [1,2] vzniká z [1,1] a [2,1] – 4 kombinace $SS, SB, AS, AB \rightarrow$ v gramatice je jen SB, tj. Y.
kombinace se vždycky počítají ze dvou políček, které se pohybují ve “véčku” nad počítaným polem.
$\emptyset$ v políčku znamená, že příslušný podřetězec nejde vygenerovat z žádného pravidla gramatiky.
složitost CKY je $\underline{\text{vždycky}}$ $O(n^3)$ na rozdíl od ostatních, kde je jen $\Omega(n^3)$
výsledek = *true*/*false* podle toho, jestli je v políčku dole kořen. pro generování stromů z CKY tabulky bychom si museli pamatovat v každém políčku, z jakých políček vznikl který neterminál.

CKY online demo

`http://www.diotavelli.net/people/void/demos/cky.html`

DCG

DCG=
Definite Clause Grammars

- Syntactic shorthand for producing parsers with Prolog clauses: Prolog-based parsing
- Represent the input with difference lists: two lists with the first containing the input to parse (a suffix of the entire input string) and the second containing the string remaining after a successful parse.
    - These two lists correspond to the input and output variables of the clauses.
    - Each clause corresponds to a non-terminal in the grammar.

Earley parser

- Jay Earley, 1968
- Strong resemblance to LR parsing but more dynamic
- Work with what are called Earley items
  - Earley item is a production augmented with a marker inserted at some point in the production's right hand side and a number to indicate where in the input matching of the production began.
  - Earley item sets are constructed by applying three operations to the current list of Earley item sets: scanner, predictor, completor

Earley algorithm

Repeat until no new item can be added:

1. Prediction

   For every state in agenda of the form $(X \rightarrow \alpha \bullet Y \beta, j)$, add $(Y \rightarrow \bullet \gamma, k)$ to agenda for every production in the grammar with Y on the left-hand side $(Y \rightarrow \gamma)$.

2. Scanning

   If a is the next symbol in the input stream, for every state in agenda of the form $(X \rightarrow \alpha \bullet a \beta, j)$, add $(X \rightarrow \alpha a \bullet \beta, j)$ to agenda.

3. Completion

   For every state in agenda of the form $(X \rightarrow \gamma \bullet, j)$, find states in agenda of the form $(Y \rightarrow \alpha \bullet X \beta, i)$ and add $(Y \rightarrow \alpha X \bullet \beta, i)$ to agenda.

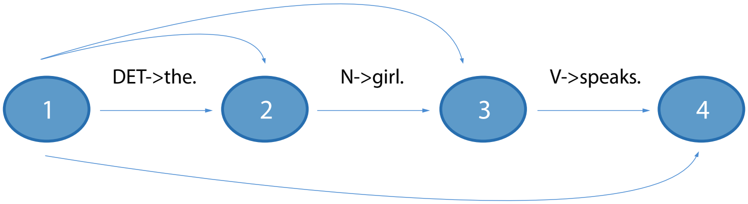
Earley algorithm

Earley's example

A pointed rule (Marker) is a production increased by a point.
The point indicates the current state of application of the rule

The girl speaks

S->•GN GV
S->GN•GV
GN-> • GN GNP
GN->GN•GNP



Earley algorithm

4	S->NP•VP		V -> speaks•
3	S->NP•VP, NP->NP•NPP	N -> girl•	
2	DET->the•, NP->DET•N		
	1	2	3
	The	**girl**	**speaks**

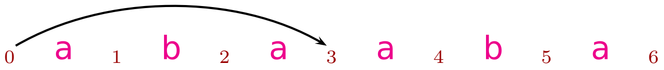
Chart parsing

- The Earley parser can be modified to work bottom-up or head-corner
- $\Rightarrow$ a variety of chart parsing algorithms (Kay, 1980)

Chart parsing

- Three basic approaches:
  - top-down
  - bottom-up
  - head-driven
- No constraints on the CF grammar
- Chart parsers usually contain two data structures *chart* and *agenda*, both of contain which contain *edges*.
- **Edge** is a triple [$A \rightarrow \alpha_{\bullet}\beta$, *i, j*], where:
  - $i, j \in \mathbb{N}$, $0 \leq i \leq j \leq n$ for *n* input words
  - $A \rightarrow \alpha\beta$ is a grammar rule

$$[A \rightarrow BC \bullet DE, 0, 3]$$



General chart parser

```
program Chart Parser;
begin
    initialize (CHART);
    initialize (AGENDA);
    while (AGENDA not empty) do
        E := take edge from AGENDA;
        for each (edge F, which can be created by
              the edge E and another edge from CHART) do
              if ((F is not in AGENDA) and (F is not in CHART) and
                (F is different from E)
                then add F to AGENDA;
              fi;
        od;
        add E to CHART;
    od;
end;
```

tato struktura programu je společná všem typům chart parserů. ty se navzájem liší v:

1. jak se *inicializuje*
2. jak se vybírá *F*

dá se to udělat i jinak (bez agendy), ale tato metoda je nejčastější
proč se nezacyklí:

1. hran je konečný počet
2. každou hranu projde maximálně jednou

Top-down approach

Initialization:

- $\forall\, p \in P \mid p = S \rightarrow \alpha$ add edge [$S \rightarrow \bullet\alpha$, 0, 0] to agenda.
- startup chart is empty.

Iteration – take edge E from agenda and then:

a) (*fundamental rule*) if E is in the form of [$A \rightarrow \alpha\bullet$, j, k], then for each edge [$B \rightarrow \gamma\bullet\, A\, \beta$, i, j] in the chart, create an edge [$B \rightarrow \gamma\, A\, \bullet\beta$, i, k].

b) (*closed edges*) if E is in the form of [$B \rightarrow \gamma\bullet\, A\, \beta$, i, j], then for each edge [$A \rightarrow \alpha\bullet$, j, k] in the chart, create an edge [$B \rightarrow \gamma\, A\, \bullet\beta$, i, k].

c) (*read terminal*) if E is in the form of [$A \rightarrow \alpha\bullet a_{j+1}\beta$, i, j], create an edge [$A \rightarrow \alpha\, a_{j+1}\bullet\beta$, i, j+1].

d) (*prediction*) if E is in the form of [$A \rightarrow \alpha\bullet\, B\, \beta$, i, j] then for each grammar rule $B \rightarrow \gamma \in P$, create an edge [$B \rightarrow \bullet\, \gamma$, i, i].

Example – chart parsing

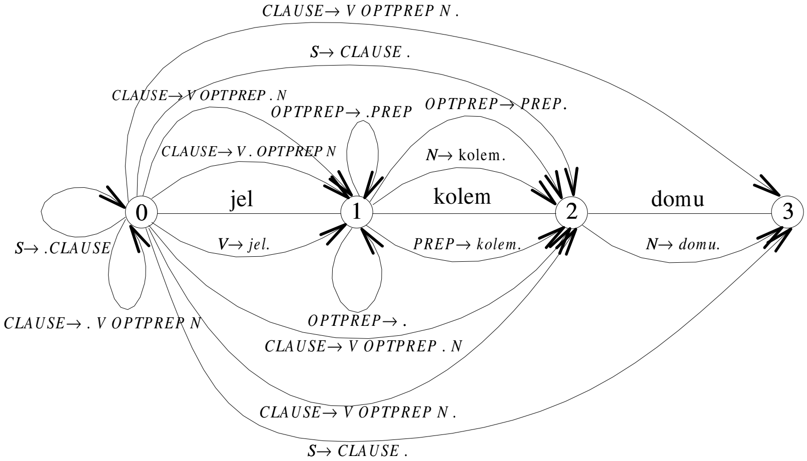
Grammar:

$$\begin{array}{rcl}
S & \rightarrow & CLAUSE \\
CLAUSE & \rightarrow & V\ OPTPREP\ N \\
OPTPREP & \rightarrow & \epsilon \\
OPTPREP & \rightarrow & PREP \\
V & \rightarrow & jel \\
PREP & \rightarrow & kolem \\
N & \rightarrow & domu \\
N & \rightarrow & kolem
\end{array}$$

Sentence:

”jel kolem domu” (a_1=*jel*, a_2=*kolem*, a_3=*domu*).

Example – chart after top-down analysis



1. inicializace
2. predikce – aplikace d)
3. predikce – aplikace d)
4. terminal – aplikace c)
5. uzavrena hrana – aplikace a)
6. …
7. v posledním – vynechané ϵ-hrany (nevešly by se)

složitost – počet pravidel bereme jako konstantu $\rightarrow$ pak máme podle délky vstupu n celkem n^2 možných hran a v každém kroku zpracujem až n hran $\rightarrow$ $O(n^3)$.

Bottom-up approach

Initialization:

- $\forall\, p \in P \mid p = A \rightarrow \epsilon$ add edges [$A \rightarrow \bullet$, 0, 0], [$A \rightarrow \bullet$, 1, 1], ..., [$A \rightarrow \bullet$, n, n] to agenda.
- $\forall\, p \in P \mid p = A \rightarrow a_i\alpha$ add edge [$A \rightarrow \bullet a_i\alpha$, i-1, i-1] to agenda.
- startup chart is empty.

Iteration – take an edge E from agenda and then:

a) (*fundamental rule*) if E is in the form of [$A \rightarrow \alpha\bullet$, j, k], then for each edge [$B \rightarrow \gamma\bullet\, A\, \beta$, i, j] in the chart, create an edge [$B \rightarrow \gamma\, A\, \bullet\beta$, i, k].

b) (*closed edges*) if E is in the form of [$B \rightarrow \gamma\bullet\, A\, \beta$, i, j], then for each edge [$A \rightarrow \alpha\bullet$, j, k] in the chart, create an edge [$B \rightarrow \gamma\, A\, \bullet\beta$, i, k].

c) (*read terminal*) if E is in the form of [$A \rightarrow \alpha\bullet a_{j+1}\beta$, i, j], then create an edge [$A \rightarrow \alpha\, a_{j+1}\bullet\beta$, i, j+1].

d) (*prediction*) if E is in the form of [$A \rightarrow \alpha\bullet$, i, j], then for each grammar rule $B \rightarrow A\gamma$ create an edge [$B \rightarrow \bullet A\gamma$, i, i].

a), b) a c) jsou stejné jako u shora dolů, liší se jen v d).
většinou vytváří víc nadbytečných hran.

Head-driven chart parsing

- **Rule head** – any particular right-hand side non-terminal E.g. in the rule *CLAUSE* $\rightarrow$ *V* $\underline{\textit{OPTPREP}}$ *N* heads can be *V*, *OPTPREP*, *N*.
- An edge is a triple [$A \rightarrow \alpha_\bullet \beta_\bullet \gamma$, *i, j*], where *i, j* $\in \mathbb{N}$, $0 \leq i \leq j \leq n$ for *n* input words, $A \rightarrow \alpha\beta\gamma$ is a grammar rule and the head is in β.
- The algorithm (bottom-up approach) is very similar to the previous simpler one. The analysis does not go left to right, but begins on the head of each rule instead.

Head-driven chart parsing

Initialization

- $\forall\ p \in P \mid p = A \rightarrow \epsilon$ add edges $[A \rightarrow \bullet\bullet$, 0, 0], $[A \rightarrow \bullet\bullet$, 1, 1], ..., $[A \rightarrow \bullet\bullet$, n, n] to agenda.
- $\forall\ p \in P \mid p = A \rightarrow \alpha \underline{a_i} \beta$ (a_i is rule head) add edge $[A \rightarrow \alpha_\bullet a_{i\bullet} \beta$, i-1, i] to agenda.
- startup chart is empty.

Head-driven chart parsing

Iteration – take and edge E from agenda and then:

- a_1) if E is in the form of $[A \rightarrow {}_\bullet\alpha_\bullet, j, k]$, then for each edge $[B \rightarrow \beta_\bullet\gamma_\bullet A\delta, i, j]$ in the chart, create edge $[B \rightarrow \beta_\bullet\gamma A_\bullet\delta, i, k]$.
- a_2) $[B \rightarrow \beta A_\bullet\gamma_\bullet\delta, k, l]$ in the chart, create edge $[B \rightarrow \beta_\bullet A\gamma_\bullet\delta, j, l]$.
- b_1) if E is in the form of $[B \rightarrow \beta_\bullet\gamma_\bullet A\delta, i, j]$, then for each edge $[A \rightarrow {}_\bullet\alpha_\bullet, j, k]$ in the chart, create edge $[B \rightarrow \beta_\bullet\gamma A_\bullet\delta, i, k]$.
- b_2) if E is in the form of $[B \rightarrow \beta A_\bullet\gamma_\bullet\delta, k, l]$, then $[A \rightarrow {}_\bullet\alpha_\bullet, j, k]$ in the chart, create edge $[B \rightarrow \beta_\bullet A\gamma_\bullet\delta, j, l]$.
- c_1) if E is in the form of $[A \rightarrow \beta a_{i\bullet}\gamma_\bullet\delta, i, j]$, then create edge $[A \rightarrow \beta_\bullet a_i\gamma_\bullet\delta, i\text{-}1, j]$.
- c_2) if E is in the form of $[A \rightarrow \beta_\bullet\gamma_\bullet a_{j+1}\delta, i, j]$, then create edge $[A \rightarrow \beta_\bullet\gamma a_{j+1\bullet}\delta, i, j+1]$.
- d) if E is in the form of $[A \rightarrow {}_\bullet\alpha_\bullet, i, j]$, then for each grammar rule $B \rightarrow \beta\ \underline{A}\ \gamma$ create edge $[B \rightarrow \beta_\bullet A_\bullet\gamma, i, j]$ (A is rule head).

Generalized LR method by Tomita

- Tomita’s Algorithm extends the standard LR parsing algorithm: LR parsing is very efficient, but can only handle a small subset of CFG
- can handle arbitrary CFG
- LR efficiency is preserved
- In order to keep a record of the parse-state, we maintain a stack consisting of symbol/state pairs.

Generalized LR method by Tomita

- *generalized LR parser (GLR)*
- Masaru Tomita: Efficient parsing for natural language, 1986
- uses a standard LR table which may contain conflicts
- stack is represented as a DAG
- reduction performed before reading action

Tree ranking

- all chart parsing methods: parallelization as means of fighting the ambiguity
- key concept: a polynomial data structure holding up to exponential parse trees
- efficient algorithms to retrieve *n*-best trees according to some ranking
- enable taking into account a probabilistic notion of a sentence

PCFG

- = Probabilistic CFG
- each rule $r \in R$ has a probability $P(r)$ assigned
- probability of a tree $t \in T$ usually computed as

$$P(t) = \Pi_{r \in t} P(r)$$

- $\Rightarrow t_{\text{best}} = argmax_t(P(t))$

Statistical parsing

- CFG $\rightarrow$ PCFG $\rightarrow$ learned grammar
- $\rightarrow$ statistical parsing
- $\rightarrow$ how to obtain probabilities (= how to *train* the parser?)

Statistical NLP

- In the 90’s: a change of paradigm in (computational) linguistics from rationalism to empiricism (corpus-based evidence)
- Simultaneously in NLP: big development of language modelling and statistical methods based on machine learning (both supervised and unsupervised).
- $\rightarrow$ statistical parsing
- vs. Chomsky:

  It must be recognised that the notion of a ‘probability of a sentence’ is an entirely useless one, under any interpretation of this term (Chomsky, 1969)

  [taken from Chapter 1 of Young and Bloothooft, eds, Corpus-Based Methods in Language and Speech Processing]

Summary

- (Probabilistic) Context-free grammar used in parsing natural language
- Chart parsing methods: CKY, Earley, head-driven chart parsing

References

- H. Bunt, M. Tamita: *Recent advances in parsing technology*, Kluwer, 1996
- H. Bunt, P. Merlo, & J. Nivre (eds.): *Trends in Parsing Technology: Dependency Parsing, Domain Adaptation, and Deep Parsing*, Springer Dordrecht, Heidelberg/London/New York 2010
- G. Dick: *Parsing techniques: a practical guide*, Springer, 2008
- J. Earley: *An efficient context-free parsing algorithm. Communications of the ACM*, 13(2):94–102, 1970
- M. Kay: *Algorithm schemata and data structures in syntactic processing. In Readings in natural language processing, pages 35–70. Morgan Kaufmann Publishers Inc.*, San Francisco, CA, USA, 1986
- M.-J. Nederhof: *Generalized left-corner parsing. In Proceedings of the sixth conference on European chapter of the Association for Computational Linguistics, pages 305–314*, Morristown, NJ, USA, 1993. Association for Computational Linguistics.

Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

Dependency Syntax and Parsing

Outline

1 Motivation

2 Dependency Syntax

3 Dependency Parsing

Motivation

- what you have seen as far: applying analysis of formal languages to a natural language – creating a phrase-structure derivation tree according to some grammar
- PS accounts for one important syntactic property: **constituency**
- is that all?
- but what about: discontinuous phrases, structure sharing

Motivation

- another crucial syntactic phenomenon is **dependency**
- what is a dependency? ”some relation between two words“
- what is the difference to phrase-structure?
- what does constituency express?
- what does dependency express?

Dependency Syntax (Melʼchuk 1988)

A more formal account – what is a dependency? A relation!

Dependency Relation

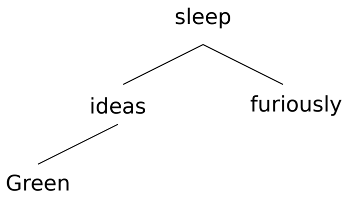
Let W be a set of all words within a sentence, then dependency relation $\rightarrow$ is $D \subseteq W \times W$ such that:

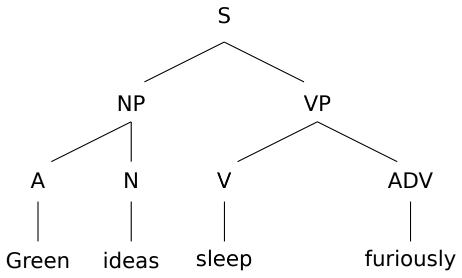
- D is **anti-reflexive**: $a \rightarrow b \Rightarrow a \neq b$
- D is **anti-symmetric**: $a \rightarrow b \wedge b \rightarrow a \Rightarrow a = b, \equiv$ (anti-reflexivity) $a \rightarrow b \Rightarrow b \nrightarrow a$
- D is **anti-transitive**: $a \rightarrow b \wedge b \rightarrow c \Rightarrow a \nrightarrow c$
- optionally: D is **labeled**: there is a mapping $l : D \rightarrow L, L$ being the set of labels

Dependency Representation

- $a \rightarrow b$: a depends on b, a is a dependent b, b is the head of a
- a **dependency graph**
- a **dependency tree**

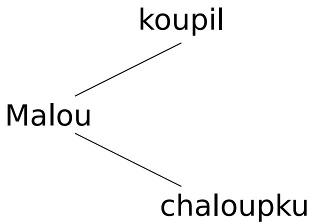
Dependency Tree vs. PS Tree





Non-projectivity

- a property of a dependency tree: a sentence is non-projective whenever drawing (projecting) a line from a node to the surface of the tree crosses an arc
- a lot of attention has been paid to this problem
- practical implications are rather limited (in most cases non-projectivity can be easily handled or avoided)
- hard cases:



Czech Tradition of Dependency Syntax

- a long tradition of dependency syntax in the Prague linguistic school (Sgall, Hajičová, Panevová)
- Institute of Formal and Applied Linguistics at Charles University
- formalized as Functional Generative Description (FGD) of language
- Prague Dependency Treebank (PDT)

Dependencies vs. PS

- is one of the formalisms clearly better than the other one? **No.**
  - dependencies: $\oplus$ account for relational phenomena, $\oplus$ simple
  - phrase-structure: $\oplus$ account for constituency, $\oplus$ easy chunking
- can we perform transformation from one of the formalism to the other one a vice versa? **Technically yes, but . . .**
  - It is not a problem to convert the structure between a dependency tree and a PS tree ...
  - ... but it is a problem to transform the information included
- $\Rightarrow$ both of the formalisms are convertible but not mutually equivalent

Dependency Parsing

- rule-based vs. statistical
- transition-based ($\rightarrow$ deterministic parsing)
- graph-based ($\rightarrow$ spanning trees algorithms)
- various other approaches (ILP, PS conversion, . . .)
- very recent advances (vs. long studied PS parsing algorithms)

Introduction to Dependency parsing

Motivation

a. dependency-based syntactic representation seem to be useful in many applications of language technology: machine translation, information extraction

   $\rightarrow$ transparent encoding of predicate-argument structure

b. dependency grammar is better suited than phrase structure grammar for language with free or flexible word order

   $\rightarrow$ analysis of diverse languages within a common framework

c. leading to the development of accurate syntactic parsers for a number of languages

   $\rightarrow$ combination with machine learning from syntactically annotated corpora (e.g. treebank)

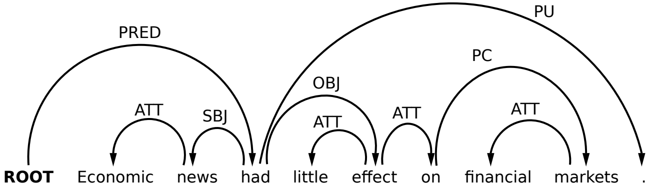
Introduction to Dependency parsing

- **Dependency parsing**

  "Task of automatically analyzing the dependency structure of a given input sentence"

- **Dependency parser**

  "Task of producing a labeled dependency structure of the kind depicted in the follow figure, where the words of the sentence are connected by typed dependency relations"



Definitions of dependency graphs and dependency parsing

Dependency graphs: syntactic structures over sentences

Def. 1.: A sentence is a sequence of tokens denoted by

$$S = w_0 w_1 \dots w_n$$

Def. 2.: Let $R = \{r_1, \dots, r_m\}$ be a finite set of *possible dependency relation types* that can hold between any two words in a sentence. A relation type $r \in R$ is additionally called an *arc label*.

Definitions of dependency graphs and dependency parsing

Dependency graphs: syntactic structures over sentences

Def. 3.: A dependency graph $G = (V, A)$ is a labeled directed graph, consists of nodes, V, and arcs, A, such that for sentence $S = w_0 w_1 \ldots w_n$ and label set R the following holds:

1. $V \subseteq \{w_0 w_1 \ldots w_n\}$
2. $A \subseteq V \times R \times V$
3. if $(w_i, r, w_j) \in A$ then $(w_i, r', w_j) \notin A$ for all $r' \neq r$

Approach to dependency parsing

a. **data-driven**
   it makes essential use of machine learning from linguistic data in order to parse new sentences

b. **grammar-based**
   it relies on a formal grammar, defining a formal language, so that it makes sense to ask whether a given input is in the language defined by the grammar or not.

$\rightarrow$ **Data-driven have attracted the most attention in recent years.**

Data-driven approach

according to the *type of parsing model* adopted,
the algorithms used to learn the model from data
the algorithms used to parse new sentences with the model

a. **transition-based**
start by defining a transition system, or state machine, for mapping a sentence to its dependency graph.

b. **graph-based**
start by defining a space of candidate dependency graphs for a sentence.

Data-driven approach

a. **transition-based**

- **learning problem:** induce a model for predicting the next state transition, given the transition history
- **parsing problem:** construct the optimal transition sequence for the input sentence, given induced model

b. **graph-based**

- **learning problem:** induce a model for assigning scores to the candidate dependency graphs for a sentence
- **parsing problem:** find the highest-scoring dependency graph for the input sentence, given induced model

Transition-based Parsing

- Transition system consists of a set C of parser configurations and of a set D of transitions between configurations.
- **Main idea:** a sequence of valid transitions, starting in the *initial configuration* for a given sentence and ending in one of several *terminal configurations*, defines a valid dependency tree for the input sentence.

$$D_{1'm} = d_1(c_1), \ldots, d_m(c_m)$$

Transition-based Parsing

- **Definition**
  Score of $D_{1'm}$ factors by configuration-transition pairs (c_i, d_i):

  $$s(D_{1'm}) = \sum_{i=1}^{m} s(c_i, d_i)$$

- **Learning**
  Scoring function $s(c_i, d_i)$ for $d_i(c_i) \in D_{1'm}$

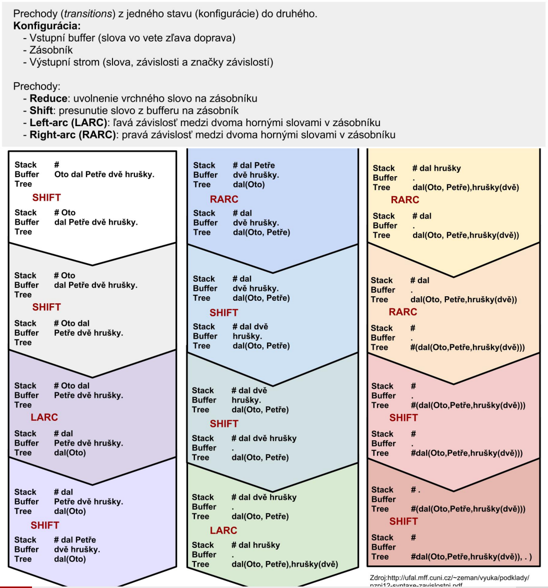
- **Inference**
  Search for highest scoring sequence $D^*_{1'm}$ given $s(c_i, d_i)$

Transition-based Parsing

Inference for transition-based parsing

- **Common inference strategies:**
  - Deterministic [Yamada and Matsumoto 2003, Nivre et al. 2004]
  - Beam search [Johansson and Nugues 2006, Titov and Henderson 2007]
  - Complexity given by upper bound on transition sequence length
- **Transition system**
  - Projective O(n) [Yamada and Matsumoto 2003, Nivre 2003]
  - Limited non-projective O(n) [Attardi 2006, Nivre 2007]
  - Unrestricted non-projective O(n2) [Nivre 2008, Nivre 2009]

Transition-based Parsing – Nivre algorithm



Prechody (*transitions*) z jedného stavu (konfigurácie) do druhého.
Konfigurácia:
- Vstupní buffer (slova vo vete zľava doprava)
- Zásobník
- Výstupní strom (slova, závislosti a značky závislostí)

Prechody:
- **Reduce**: uvolnenie vrchného slovo na zásobníku
- **Shift**: presunutie slovo z bufferu na zásobník
- **Left-arc (LARC)**: ľavá závislosť medzi dvoma hornými slovami v zásobníku
- **Right-arc (RARC)**: pravá závislosť medzi dvoma hornými slovami v zásobníku

Stack	#
Buffer	Oto dal Petře dvě hrušky.
Tree	

SHIFT

Stack	# Oto
Buffer	dal Petře dvě hrušky.
Tree	

Stack	# Oto
Buffer	dal Petře dvě hrušky.
Tree	

SHIFT

Stack	# Oto dal
Buffer	Petře dvě hrušky.
Tree	

Stack	# Oto dal
Buffer	Petře dvě hrušky.
Tree	

LARC

Stack	# dal
Buffer	Petře dvě hrušky.
Tree	dal(Oto)

Stack	# dal
Buffer	Petře dvě hrušky.
Tree	dal(Oto)

SHIFT

Stack	# dal Petře
Buffer	dvě hrušky.
Tree	dal(Oto)

Stack	# dal Petře
Buffer	dvě hrušky.
Tree	dal(Oto)

RARC

Stack	# dal
Buffer	dvě hrušky.
Tree	dal(Oto, Petře)

Stack	# dal
Buffer	dvě hrušky.
Tree	dal(Oto, Petře)

SHIFT

Stack	# dal dvě
Buffer	hrušky.
Tree	dal(Oto, Petře)

Stack	# dal dvě
Buffer	hrušky.
Tree	dal(Oto, Petře)

SHIFT

Stack	# dal dvě hrušky
Buffer	.
Tree	dal(Oto, Petře)

Stack	# dal dvě hrušky
Buffer	.
Tree	dal(Oto, Petře)

LARC

Stack	# dal hrušky
Buffer	.
Tree	dal(Oto, Petře),hrušky(dvě)

Stack	# dal hrušky
Buffer	.
Tree	dal(Oto, Petře),hrušky(dvě)

RARC

Stack	# dal
Buffer	.
Tree	dal(Oto, Petře,hrušky(dvě))

Stack	# dal
Buffer	.
Tree	dal(Oto, Petře,hrušky(dvě))

RARC

Stack	#
Buffer	.
Tree	#(dal(Oto,Petře,hrušky(dvě)))

Stack	#
Buffer	.
Tree	#(dal(Oto,Petře,hrušky(dvě)))

SHIFT

Stack	#
Buffer	.
Tree	#dal(Oto,Petře,hrušky(dvě)))

Stack	# .
Buffer	
Tree	#(dal(Oto,Petře,hrušky(dvě)))

SHIFT

Stack	#
Buffer	
Tree	#dal(Oto,Petře,hrušky(dvě)), .)

Zdroj:http://ufal.mff.cuni.cz/~zeman/vyuka/podklady/pzpi12-syntaxe-zavislostni.pdf

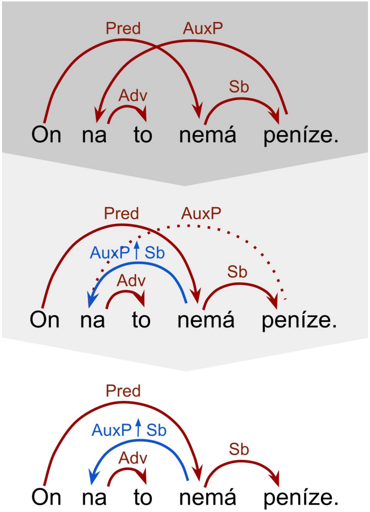
Transition-based Parsing

Learning for transition-based parsing

- **Typical scoring function:**
  - $s(c_i, d_i) = w \cdot f(c_i, d_i)$ where $f(c_i, d_i)$ is a feature vector over configuration c_i and transition d_i and w is a weight vector $[w_i =$ weight of feature$f_i(c_i, d_i)]$
- **Transition system**
  - Projective O(n) [Yamada and Matsumoto 2003, Nivre 2003]
  - Limited non-projective O(n) [Attardi 2006, Nivre 2007]
  - Unrestricted non-projective O(n2) [Nivre 2008, Nivre 2009]
- **Problem**
  - Learning is local but features are based on the global history

Transition-based Parsing

Projectivization to pseudo-projectivity:



Graph-based Parsing

- For a input sentence S we define a graph $G_s = (V_s, A_s)$ where $V_s = \{w_0, w_1, \dots, w_n\}$ and $A_s = \{(w_i, w_j, l) | w_i, w_j \in V$ and $l \in L\}$
- Score of a dependency tree T factors by subgraphs $G_s, \dots, Gs$:

$$s(T) = \sum_{i-1}^{m} s(G_i)$$

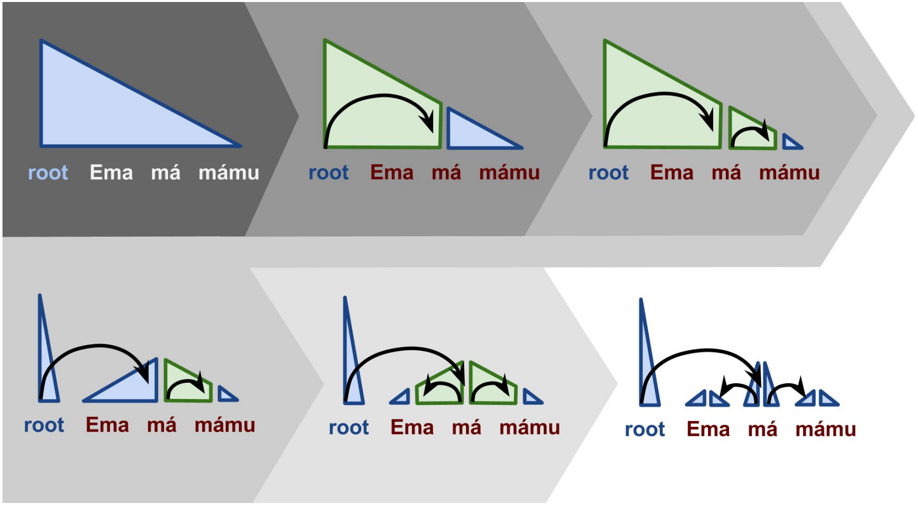
- Learning: **Scoring function** $s(G_i)$ for a subgraph $G_i \in T$
- Inference: Search for maximum spanning tree scoring sequence T^* of G_s given $s(G_i)$

Graph-based Parsing

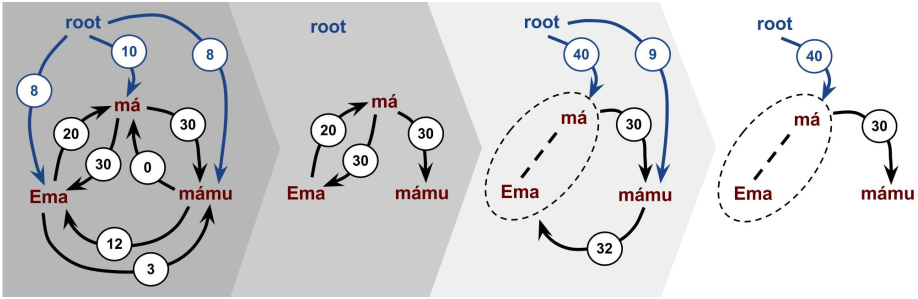
Learning graph-based models

- **Typical scoring function:**
  - $s(G_i) = w \cdot f(G_i)$ where $f(G_i)$ is a high-dimensional feature vector over subgraphs and w is a weight vector $[w_j =$ weight of feature $f_j(G_i)]$
- **Structured learning** [McDonald et al. 2005a, Smith and Johnson 2007]:
  - Learn weights that maximize the score of the correct dependency tree for every sentence in the training set
- **Problem**
  - Learning is global (trees) but features are local (subgraphs)

Graph-based Parsing – Eisner algorithm



Graph-based Parsing – Chu-Liu-Edmonds algorithm



Grammar-based approach

a. **context-free dependency parsing**
exploits a mapping from dependency structures to CFG structure representations and reuses parsing algorithms originally developed for CFG $\rightarrow$ chart parsing algorithms

b. **constraint-based dependency parsing**

- parsing viewed as a constraint satisfaction problem
- grammar defined as a set of constraints on well-formed dependency graphs
- finding a dependency graph for a sentence that satisfies all the constraints of the grammar (having the best score)

Grammar-based approach

a. **context-free dependency parsing**

Advantage: Well-studied parsing algorithms such as CKY, Earley's algorithm can be used for dependency parsing as well.

$\rightarrow$ need to convert dependency grammars into efficiently parsable context-free grammars; (e.g. bilexical CFG, Eisner and Smith, 2005)

b. **constraint-based dependency parsing**

defines the problem as constraint satisfaction

- Weighted constraint dependency grammar (WCDG, Foth and Menzel, 2005)
- Transformation-based CDG

Conclusions

1. Dependency syntax vs. constituency (phrase-structure) syntax
2. Non-projectivity
3. Graph-based and Transition-based methods

Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

Parsing with (L)TAG and LFG

(Lexicalized) Tree Adjoining Grammar (TAG) and Lexical Functional Grammar (LFG)

A) **Same goal**

- formal system to model human speech
- model the syntactic properties of natural language
- syntactic frame work which aims to provide a computationally precise and psychologically realistic representation of language

B) **Properties**

- Unfication based
- Constraint-based
- Lexicalized grammar

How to parse the sentence in TAG?

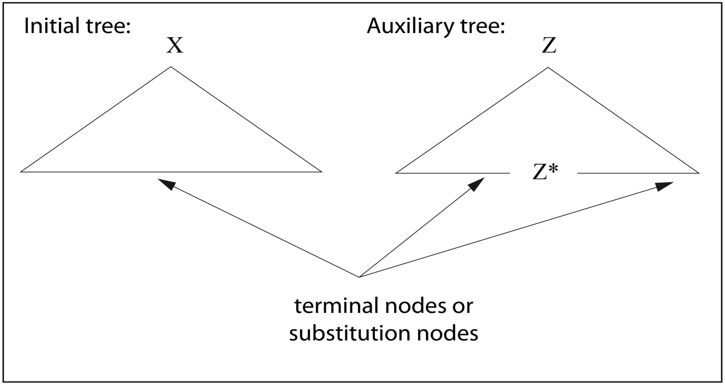
by Joshi, A. Levy, L and Takahashi, M. in 1975

TAG’s basic component

- Representation structure: phrase-structure trees
- Finite set of elementary trees
  - Two kinds of elementary trees
    - **Initial trees** (α): trees that can be substituted
    - **Auxiliary trees** (β): trees that can be adjoined

TAG’s basic component

- The tree in (X ∪ Z) are called elementary trees.



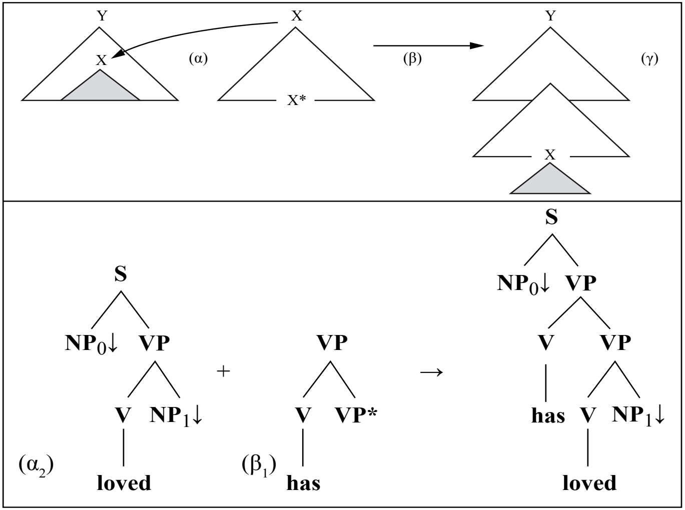
TAG’s basic component

- An initial tree (α)
  - all interior nodes are labeled with non-terminal symbols
  - the nodes on the frontier of initial tree are either labeled with terminal symbols, or with non-terminal symbols marked for substitution ($\downarrow$)
- An auxiliary tree (β)
  - one of its frontier nodes must be marked as foot node ($*$)
  - the foot node must be labeled with a non-terminal symbol which is identical to the label of the root node.
- A derived tree (γ)
  - tree built by composition of two other trees
  - the two composition operations that TAG uses adjoining and substitution.

Main operations of combination (1): adjunction

- Sentence of the language of a TAG are derived from the composition of an α and any number of β by this operation.
  - It allows to insert a complete structure into an interior node of another complete structure.
- Three constraints possible
  - Null adjunction (NA)
  - Obligatory adjunction (OA)
  - Selectional adjunction (SA)

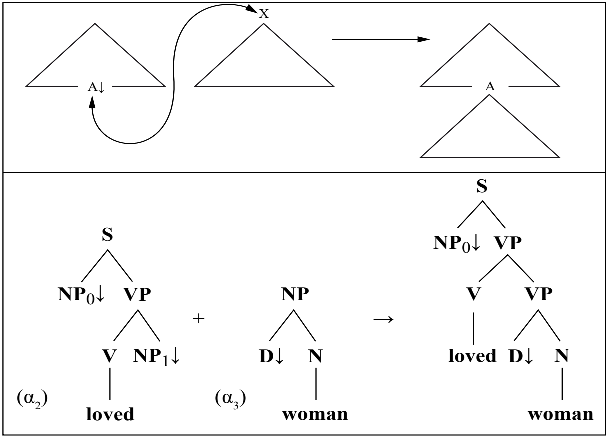
Main operations of combination (1): adjunction



Adjoining

Main operations of combination (2): substitution

- It inserts an initial tree or a lexical tree into an elementary tree.
- One constraint possible
  - Selectional substitution



Substitution

Adjoining constraints

Selective Adjunction ($SA(T)$): only members of a set $T \subseteq A$ can be adjoined on the given node, but the adjunction is not mandatory

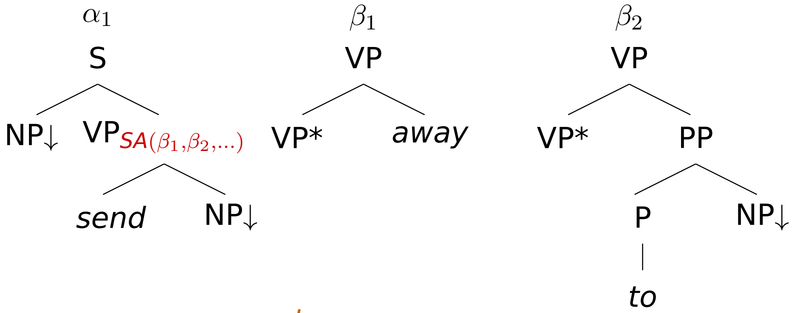
Null Adjunction (NA): any adjunction is disallowed for the given node ($NA = SA(\phi)$)

Obligatory Adjunction ($OA(T)$): an auxiliary tree member of the set $T \subseteq A$ must be adjoined on the given node

for short $OA = OA(A)$

Example 1: selective adjunction (SA)

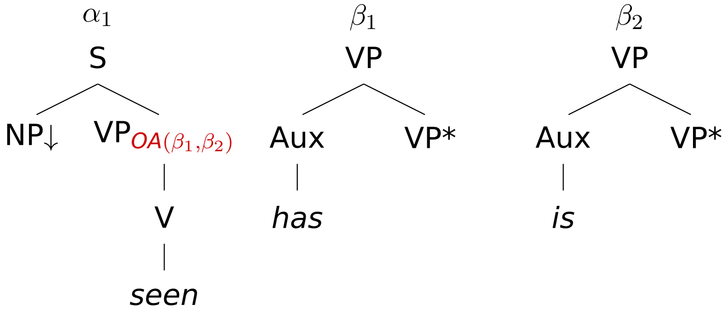
- One possible analysis of “send” could involve selective adjunction:



$$\begin{tikzcd}
 & \textit{send away} \\
\textit{send} \arrow[ur] \arrow[r] \arrow[dr] & \textit{send to} \\
 & \textit{send something}
\end{tikzcd}$$

Example 2: obligatory adjunction

- For when you absolutely must have adjunction at a node:

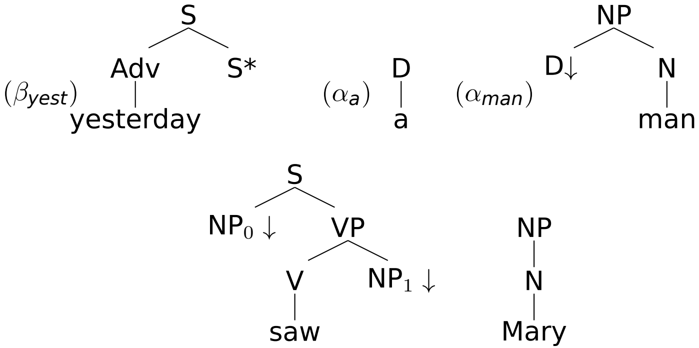


has $\longrightarrow$ *has seen*

is $\longrightarrow$ *is seen*

Elementary trees (initial trees and auxiliary trees)

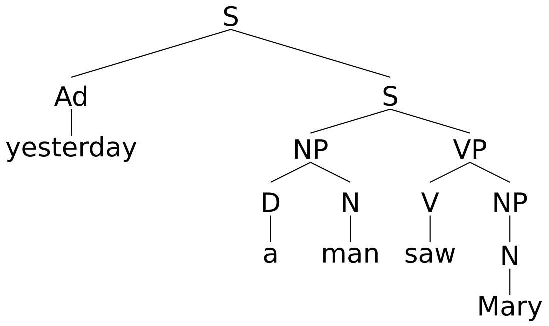
Yesterday a man saw Mary



*: foot node/root node
↓: substitution node

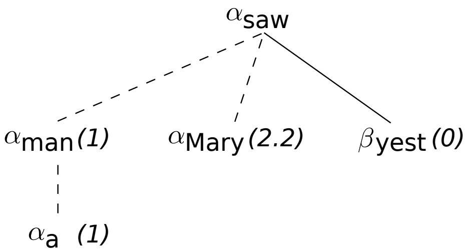
Elementary trees (initial trees and auxiliary trees)

(α_5)

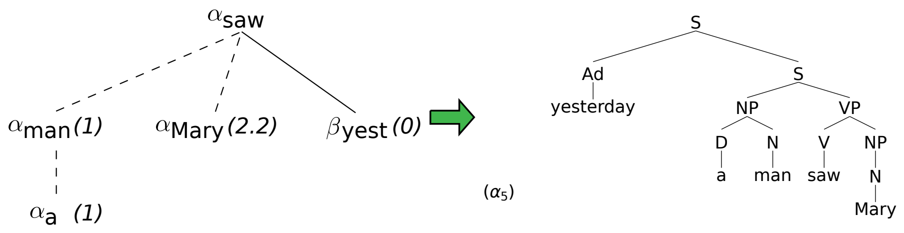


Derivation tree

- Specifies how a derived tree was constructed
  - The root node is labeled by an S-type initial tree.
  - Other nodes are labeled by auxiliary trees in the case of adjoining or initial trees in the case of substitution.
  - A tree address of the parent tree is associated with each node.


α_{saw}
α_{man} *(1)*
α_{Mary} *(2.2)*
β_{yest} *(0)*
α_{a} *(1)*

Derivation tree and derived tree α_5

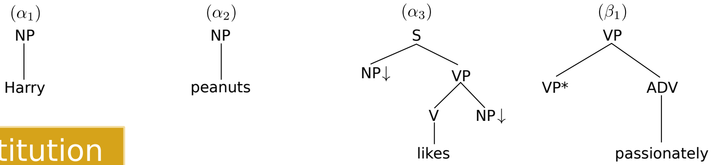


_ _ _ _ : substitution operation

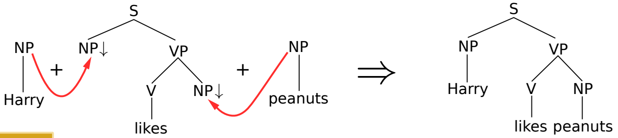
______ : adjunction operation

Example 1: Harry likes peanuts passionately

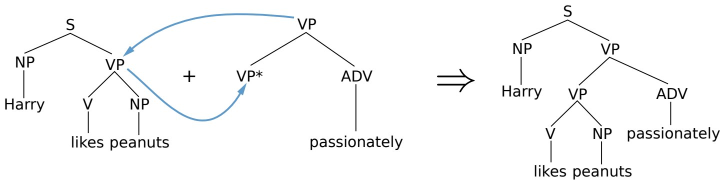
Step 1



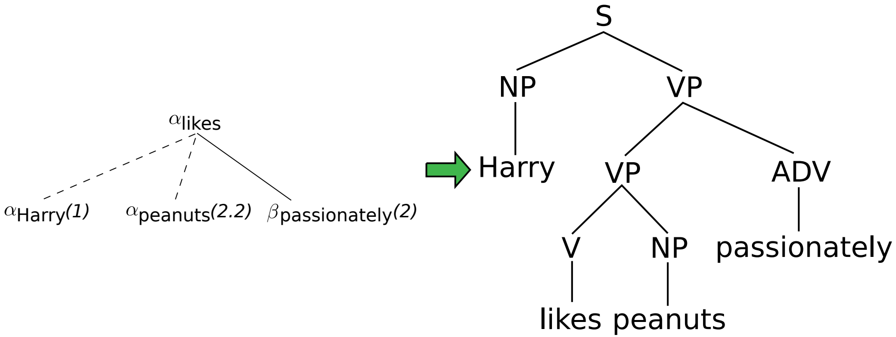
Step 2: substitution



Step 3: adjunction



Derivation tree and derived tree of Harry likes peanuts passionately

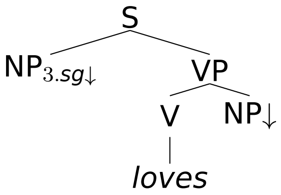

α_{likes}
α_{Harry}*(1)*
α_{peanuts}*(2.2)*
$\beta_{\text{passionately}}$*(2)*
S
NP
VP
Harry
VP
ADV
V
NP
passionately
likes
peanuts

Two important properties of TAG

- Elementary trees can be of arbitrary size, so the domain of locality is increased
  - Extended domain of locality (EDL)
- Small initial trees can have multiple adjunctions inserted within them, so what are normally considered non-local phenomena are treated locally
  - Factoring recursion from the domain of dependency (FRD)

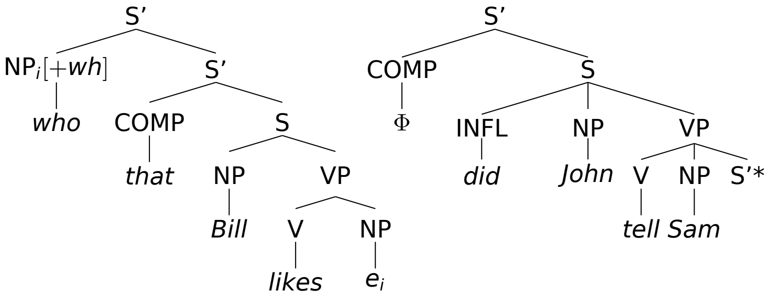
Extended domain of locality (EDL): Agreement

- The lexical entry for a verb like “loves” will contain a tree like the following:



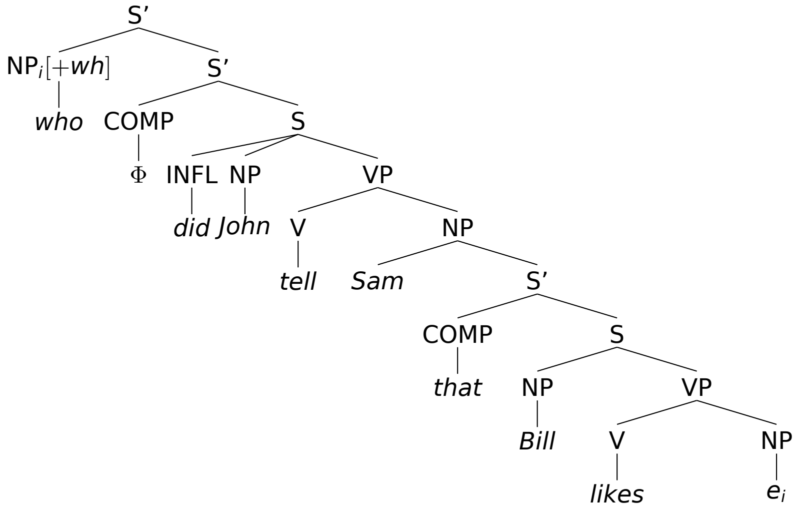
With EDL, we can easily state agreement between the subject and the verb in a lexical entry

Factoring recursion from the domain of dependency (FRD): Extraction



The above trees for the sentence "who did John tell Sam that Bill likes ?" allow the insertion of the auxiliary tree in between the WH-phrase and its extraction site, resulting a long distance dependency; yet this is factored out from the domain of locality in TAG.

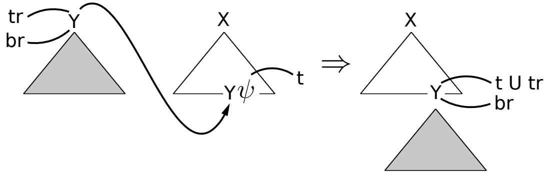
Factoring recursion from the domain of dependency (FRD): Extraction



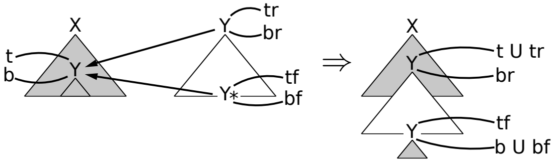
Variations of TAG

- Feature Structure Based TAG (FTAG: Joshi and Shanker, 1988)

each of the nodes of an elementary tree is associated with two feature structures:
top & bottom Substitution



Substitution with features



Adjoining with features

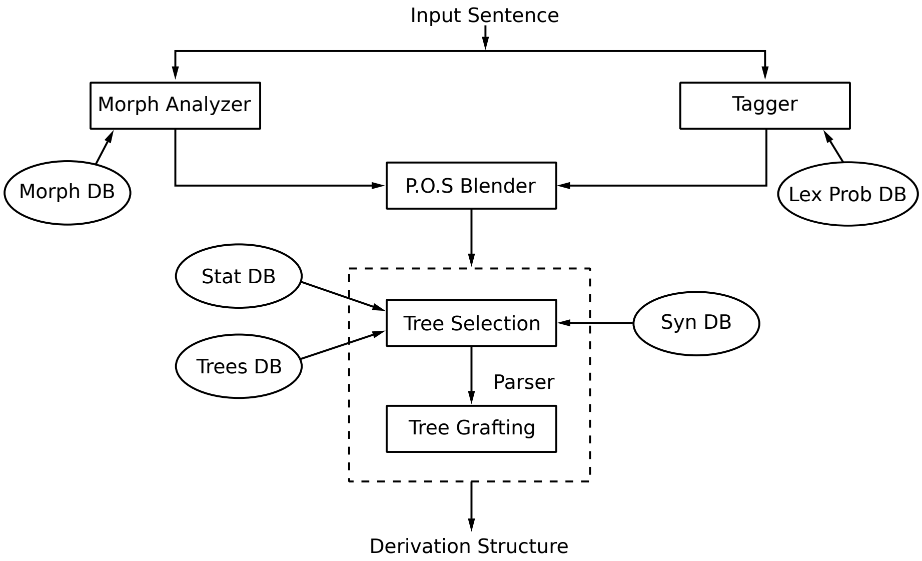
Variations of TAG

- Synchronous TAG (STAG: Shieber and Schabes, 1990)
  - A pair of TAGs characterize correspondences between languages
  - Semantic interpretation, language generation and translation
- Muti-component TAG (MCTAG: Chen-Main and Joshi, 2007)
  - A set of auxiliary tree can be adjoined to a given elementary tree
- Probabilistic TAG (PTAG: Resnik, 1992, Shieber, 2007)
  - Associating a probability with each elementary tree
  - Compute the probability of a derivation

XTAG Project (UPenn, since 1987 ongoing)

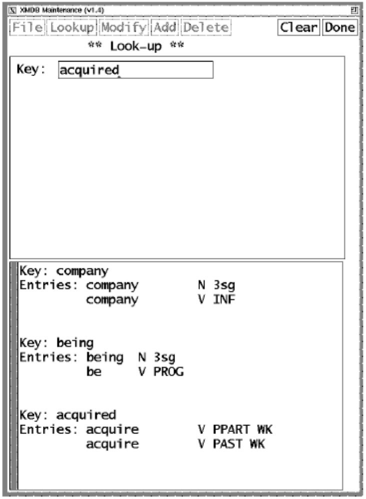
- A long-term project to develop a wide-coverage grammar for English using the Lexicalized Tree-Adjoining Grammar (LTAG) formalism
- Provides a grammar engineering platform consisting of a parser, a grammar development interface, and a morphological analyzer
- The project extends to variants of the formalism, and languages other than English

XTAG system


Input Sentence
Morph Analyzer
Tagger
Morph DB
P.O.S Blender
Lex Prob DB
Stat DB
Tree Selection
Syn DB
Trees DB
Parser
Tree Grafting
Derivation Structure

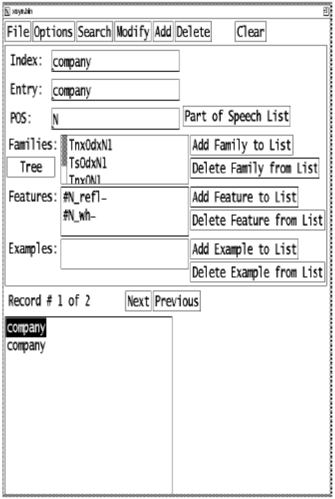
Components in XTAG system

- Morphological Analyzer & Morph DB: 317K inflected items derived from over 90K stems
- POS Tagger & Lex Prob DB: Wall Street Journal-trained 3-gram tagger with N-best POS sequences
- Syntactic DB: over 30K entries, each consisting of:
    - Uninflected form of the word
    - POS
    - List of trees or tree-families associated with the word
    - List of feature equations
- Tree DB: 1004 trees, divided into 53 tree families and 221 individual trees



Key: acquired

```
Key: company
Entries: company        N  3sg
         company        V  INF


Key: being
Entries: being  N  3sg
         be     V  PROG


Key: acquired
Entries: acquire        V  PPART WK
         acquire        V  PAST WK
```



Index: company

Entry: company

POS: N

Families: TnxOdxN1, TsOdxN1, TnxON1

Features: #N_refl-, #N_wh-

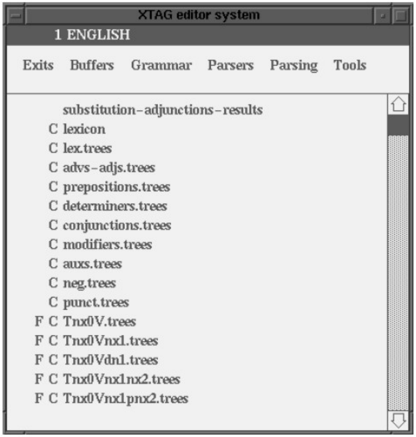
Examples:

Record # 1 of 2

company
company

(a) Morphology database (b) syntactic database

Interfaces to the database maintenance tools



substitution-adjunctions-results
C lexicon
C lex.trees
C advs-adjs.trees
C prepositions.trees
C determiners.trees
C conjunctions.trees
C modifiers.trees
C auxs.trees
C neg.trees
C punct.trees
F C Tnx0V.trees
F C Tnx0Vnx1.trees
F C Tnx0Vdn1.trees
F C Tnx0Vnx1nx2.trees
F C Tnx0Vnx1pnx2.trees

Interface to the XTAG system

Parser evaluation in XTAG Project by [Bangalore,S. *et.al*, 1998]
`http://www.cis.upenn.edu/~xtag/`

How to parse the sentence in LFG?

by Bresnan, J. and Kaplan, R.M. In 1982

Main representation structures

- ***c-structure***: constituent structure

level where the surface syntactic form, including categorical information, word order and phrasal grouping of constituents, is encoded.

- ***f-structure***: functional structure

internal structure of language where grammatical relations are represented. It is largely invariable across languages. (e.g. SUBJ, OBJ, OBL, (X)COMP, (X)ADJ)

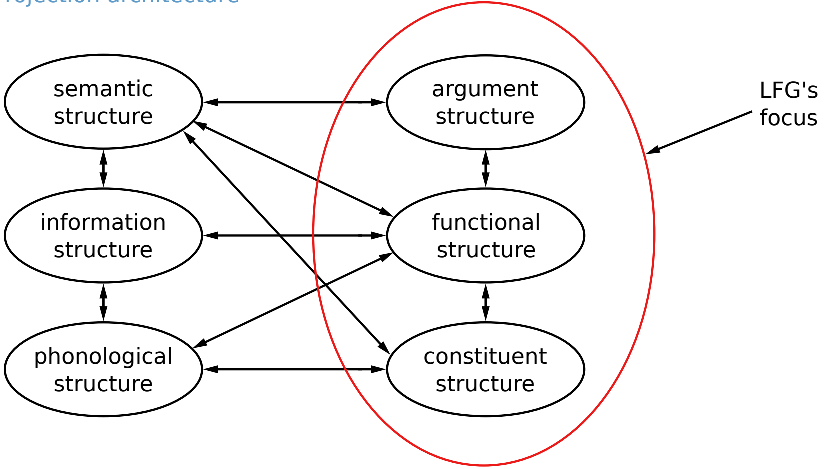
- ***a-structure***: argument structure

They encode the number, type and semantic roles of the arguments of a predicate.

Level of structures and their interaction in LFG

Functional

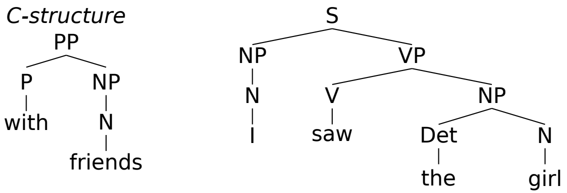
Projection architecture



Level of structures and their interaction in LFG

- In LFG, the parsing result is grammatically correct only if it satisfies 2 criteria:
  1. the grammar must be able to assign a correct c-structure
  2. the grammar must be able to assign a correct well-formed f-structure

c-structure



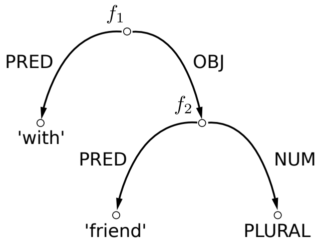
- The constituent structure represents the organization of overt phrasal syntax
- It provides the basis for phonological interpretation
- Languages are very different on the c-structure level :external factors that usually vary by language

Properties of c-structure

- *c-structures* are conventional phrase structure trees:

  they are defined in terms of syntactic categories, terminal nodes, dominance and precedence.
- They are determined by a context free grammar that describes all possible surface strings of the language.
- LFG does not reserve constituent structure positions for affixes: all leaves are individual words.

f-structure



- Attribute-Value notation for *f-structure*

$$\begin{bmatrix} \text{PRED} & \text{'with'} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{'friend'} \\ \text{NUM} & \text{PLURAL} \end{bmatrix} \end{bmatrix}$$

1. representation of the functional structure of a sentence
2. *f-structure* match with *c-structure*
3. it has to satisfy three formal constraints: consistency, coherence, completeness
4. language are similar on this level: allow to explain cross-linguistic properties of phenomena

Examples of *f-structure*

1

$$\begin{bmatrix} \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{'Veit'} \end{bmatrix} \\ \text{TENSE} & \text{PAST} \\ \text{PRED} & \text{'send}\langle\text{SUBJ, OBJ, OBJ2}\rangle\text{'} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'Sabine'} \end{bmatrix} \\ \text{OBJ2} & \begin{bmatrix} \text{PRED} & \text{'e-mail'} \\ \text{DEF} & - \\ \text{NUM} & \text{SG} \end{bmatrix} \end{bmatrix}$$

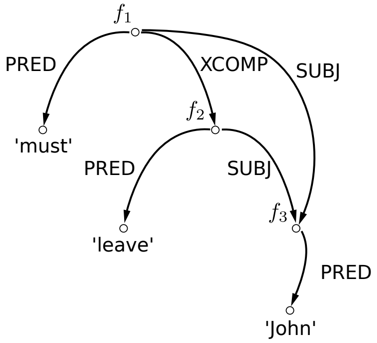
2

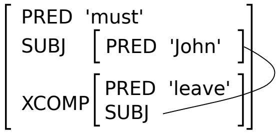
$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'teacher'} \\ \text{DEF} & + \\ \text{NUM} & \text{SG} \end{bmatrix} \\ \text{TENSE} & \text{PAST} \\ \text{PRED} & \text{'insist}\,\langle\,\text{SUBJ, OBL}_{\text{on}}\,\text{OBJ}\,\rangle\,\text{'} \\ \text{OBL}_{\text{on}} & \begin{bmatrix} \text{PCASE} & \text{OBL}_{\text{on}} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{'homework'} \\ \text{DEF} & + \\ \text{NUM} & \text{SG} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Constraint 1: *f-structure* must be consistent

1. Two paths in the graph structure may designate the same element-called unification, structure-sharing

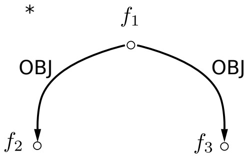
Ex: *John must leave*





Constraint 1: *f-structure* must be consistent

2. attributes are functionally unique - there may not be two arcs with the same attribute from the same f-structure

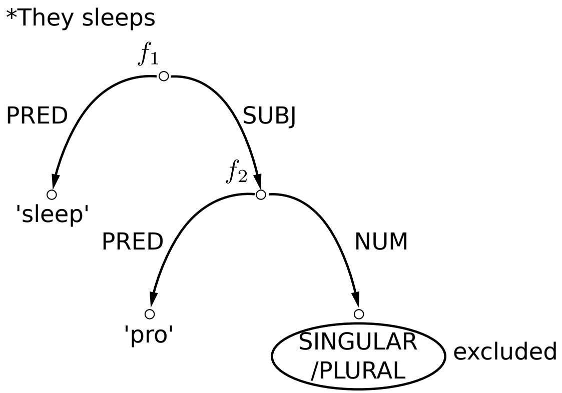


Incosnistent f-structure

$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'Veit'} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'Tom'} \end{bmatrix} \\ \text{PRED} & \text{'sleep}\left\langle\left(\uparrow\text{SUBJ}\right)\right\rangle\text{'} \\ \text{TENSE} & \text{PAST} \\ \text{TENSE} & \text{FUT} \end{bmatrix}$$

Constraint 1: *f-structure* must be consistent

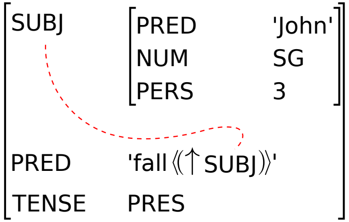
3. The symbols used for atomic f-structure are distinct - it is impossible to have two names for a single atomic f-structure ("clash")



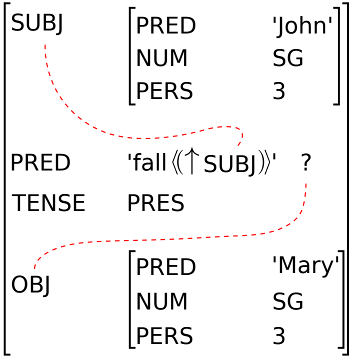
Constraint 2: *f-structure* must be coherent

All argument functions in an *f-structure* must be selected by the local PRED feature.

Complete f-structure



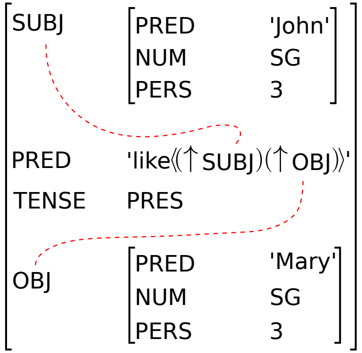
Incoherent f-structure



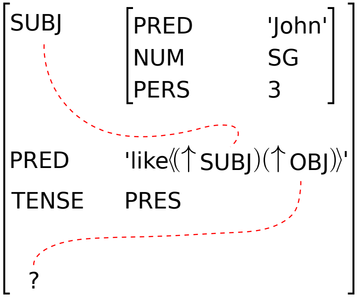
Constraint 3: *f-structure* must be complete

All functions specified in the value of a PRED feature must be present in the *f-structure* of that PRED.

Complete f-structure

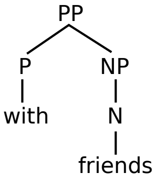


Incoherent f-structure



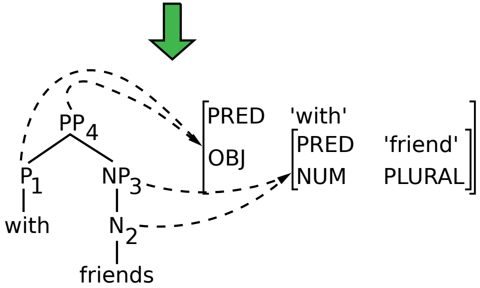
Correspondence between different levels in LFG

C-structure



+

$$\begin{bmatrix} \text{PRED} & \text{'with'} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{'friend'} \\ \text{NUM} & \text{PLURAL} \end{bmatrix} \end{bmatrix}$$

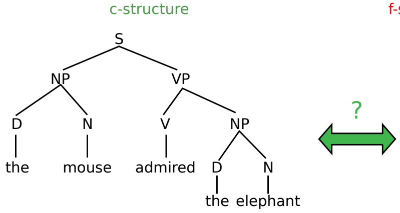


Structural correspondence

- *c-structures* and *f-structures* represent different properties of an utterance
- How can these structures be associated properly to a particular sentence?
- Words and their ordering carry information about the linguistic dependencies in thesentence
- This is represented by the *c-structure* (licensed by a CFG)
- LFG proposes simple mechanisms that maps between elements from one structure and those of another: correspondence functions
- A function allows to map c-structures to f-structures $\Phi : N \rightarrow F$

Mapping the c-structure into the f-structure

- Since there is no isomorphic relationship between structure and function LFG assumes *c-structure* and *f-structure*
- The mapping between *c-structure* and *f-structure* is the core of LFG‘s descriptive power
- The mapping between *c-structure* and *f-structure* is located in the grammar (PS) rules



f-structure

$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'mouse'} \\ \text{DEF} & + \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{bmatrix} \\ \text{TENSE} & \text{PAST} \\ \text{PRED} & \text{'admire}\langle(\uparrow\text{SUBJ})\,(\uparrow\text{OBJ})\rangle\text{'} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{'elephant'} \\ \text{DEF} & + \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{bmatrix} \end{bmatrix}$$

Mapping mechanism: 6 steps

STEP 1: PS rules

- Context-free phrase structure rules
- Annotated with functional schemata

- EX.:

$$S \rightarrow \underset{(\uparrow \mathrm{SUBJ})=\downarrow}{NP} \quad \underset{\uparrow=\downarrow}{VP}$$

mother node (without functional schemata)

daughter nodes (with (a list of) functional schemata)

- EX.:

$$NP \rightarrow \underset{\uparrow=\downarrow}{NP} \quad \underset{\uparrow=\downarrow}{NP}$$

$$VP \rightarrow \underset{\uparrow=\downarrow}{V} \quad \underset{(\uparrow \mathrm{SUBJ})=\downarrow}{(NP)}$$

Note:
↑=↓ is sometimes omitted!

(this means nodes without functional schemata percolate their entire functional schema unchanged to the mother node

Mapping mechanism: 6 steps

STEP 2: Lexicon entries

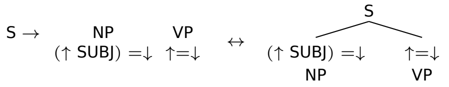
- Lexicon entries consists of three parts: representation of the word, syntactic category, list of functional schemata

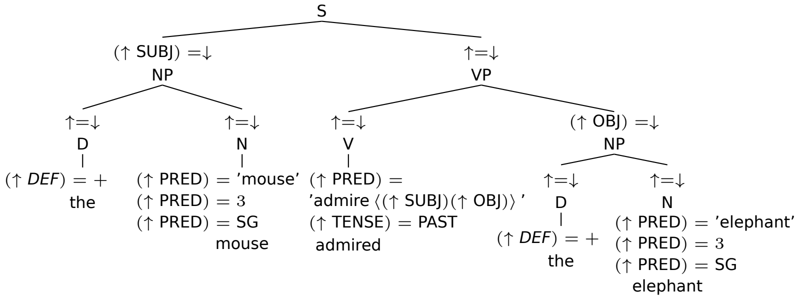
Ex.:	mouse	N	$(\uparrow$PRED$)=$'mouse'
			$(\uparrow$PERS$)=3$
			$(\uparrow$NUM$)=$SG
	the	D	$(\uparrow$DEF$)=+$
	admire	V	$(\uparrow$PRED$)=$'admire $\langle(\uparrow$ SUBJ$)(\uparrow$ OBJ$)\rangle$'
	-ed	Aff	$(\uparrow$TENSE$)=$PAST

Mapping mechanism: 6 steps

STEP 3: *c-structure*

- Like the PS rules, each node in the tree is associated with a functional schemata
- With the functional schemata of the lexical entries at the leaves we obtain a complete *c-structure*

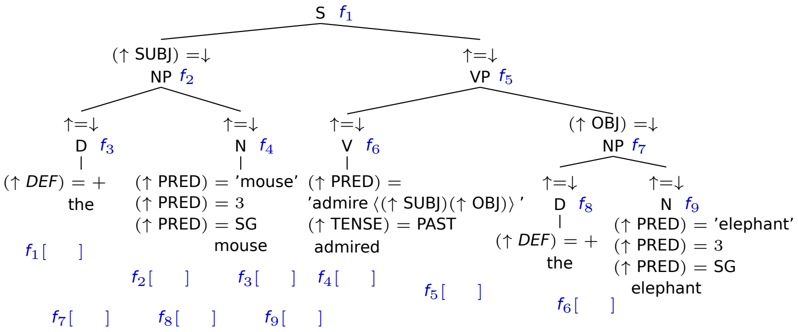




Mapping mechanism: 6 steps

STEP 4: Co-indexation

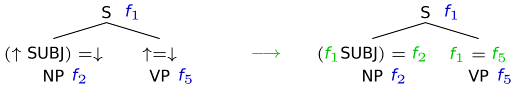
- An f-structure is assigned to each node of the c-structure
- Each of these f-structures obtains a name $(f_1 - f_n)$
- Nodes in the c-structure and associated f-structure are co-indexed, i.e. obtain the same name
- F-structure names $f_1 - f_n$ can be chosen freely but they may not occur twice

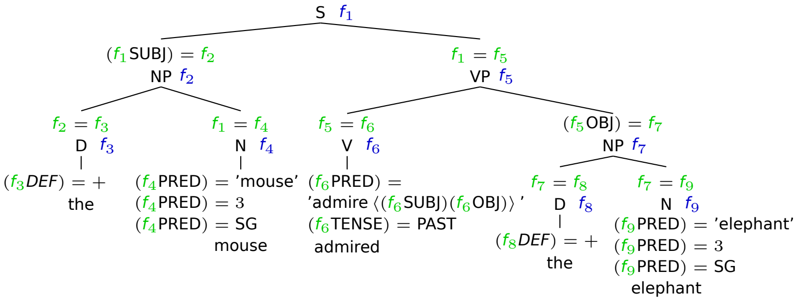


Mapping mechanism: 6 steps

STEP 5: Metavariable biding

- All meta-variables are replaced by the names of the *f-structure* representation





Mapping mechanism: 6 steps

- We introduce at this point the notion of **functional equation**
- By listing all functional equations from a *c-structure* we obtain the functional description, called **f-description**

$(f_1\text{SUBJ}) = f_2$	$(f_6\text{PRED}) = \text{'admire}\langle(f_6\text{SUBJ})(f_6\text{OBJ})\rangle\text{'}$
$f_2 = f_3$	$(f_6\text{TENSE}) = \text{PAST}$
$(f_3\text{DEF}) = +$	$(f_5\text{OBJ}) = f_7$
$f_2 = f_4$	$f_7 = f_8$
$(f_4\text{PRED}) = \text{'mouse'}$	$(f_8\text{DEF}) = +$
$(f_4\text{PERS}) = 3$	$f_7 = f_9$
$(f_4\text{NUM}) = \text{SG}$	$(f_9\text{PRED}) = \text{'elephant'}$
$f_1 = f_5$	$(f_9\text{PERS}) = 3$
$f_5 = f_6$	$(f_9\text{NUM}) = \text{SG}$

Table : f-description

Mapping mechanism: 6 steps

STEP 6: From f-description to *f-structure*

- Computation of an *f-structure* is based on the **f-description**
- For the derivation of *f-structures* from the **f-description** it is important that no information is lost and that no information will be added
- The derivation is done by the application of the **functional equations**

List of functional equations

a) simple equations of the form: $f_n A) = B$

b) f-equations of the form: $f_n = f_m$

c) f-equations of the form: $f_n A) = f_m$

$\rightarrow$ Functional equations with the same name are grouped into an *f-structure* of the same name

Application of the functional equation (a): $(f_nA) = B$

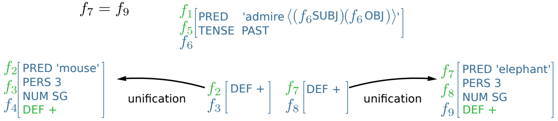
$(f_3$DEF$)=+$
$(f_4$PRED$)=$'mouse'
$(f_4$PERS$)=3$
$(f_4$NUM$)=$SG
$(f_6$PRED$)=$'admire $\langle(f_6$SUBJ$)(f_6$OBJ$)\rangle$'
$(f_6$TENSE$)=$PAST
$(f_8$DEF$)=+$
$(f_9$PRED$)=$'ELEPHANT'
$(f_9$PERS$)=3$
$(f_9$NUM$)=$SG

$$f_8 \begin{bmatrix} \text{DEF } + \end{bmatrix} \quad f_3 \begin{bmatrix} \text{DEF } + \end{bmatrix}$$

$$f_4 \begin{bmatrix} \text{PRED 'mouse'} \\ \text{PERS 3} \\ \text{NUM SG} \end{bmatrix} \quad f_6 \begin{bmatrix} \text{PRED} & \text{'admire } \langle(f_6\text{SUBJ})(f_6\text{OBJ})\rangle\text{'} \\ \text{TENSE} & \text{PAST} \end{bmatrix} \quad f_9 \begin{bmatrix} \text{PRED 'elephant'} \\ \text{PERS 3} \\ \text{NUM SG} \end{bmatrix}$$

Application of the functional equation (b): $f_n = f_m$

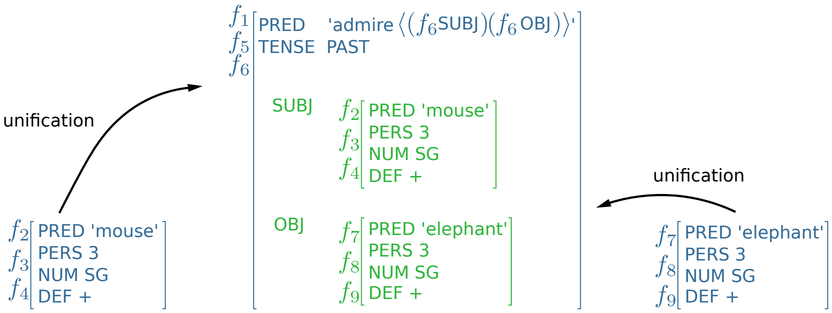
$f_2 = f_3$
$f_2 = f_4$
$f_1 = f_5$
$f_5 = f_6$
$f_7 = f_8$
$f_7 = f_9$



$f_1 f_5 f_6$ [PRED 'admire ⟨(f6SUBJ)(f6 OBJ)⟩' TENSE PAST]
$f_2 f_3 f_4$ [PRED 'mouse' PERS 3 NUM SG DEF +]
unification
$f_2 f_3$ [DEF +]
$f_7 f_8$ [DEF +]
unification
$f_7 f_8 f_9$ [PRED 'elephant' PERS 3 NUM SG DEF +]

Application of the functional equation (c): $(f_n A) = f_m$

$(f_1\,\text{SUBJ}) = f_2$
$(f_5\,\text{OBJ}) = f_7$


f_1 f_5 f_6
PRED 'admire $\langle (f_6\,\text{SUBJ})(f_6\,\text{OBJ}) \rangle$'
TENSE PAST
SUBJ
f_2 f_3 f_4 PRED 'mouse' PERS 3 NUM SG DEF +
OBJ
f_7 f_8 f_9 PRED 'elephant' PERS 3 NUM SG DEF +
unification
f_2 f_3 f_4 PRED 'mouse' PERS 3 NUM SG DEF +
unification
f_7 f_8 f_9 PRED 'elephant' PERS 3 NUM SG DEF +

STEP 1: lexical entries

made: V	(↑PRED)='MAKE⟨SUBJ,OBJ,XCOMP⟩' (↑XCOMP SUBJ)=(↑OBJ) (↑TENSE)=SIMPLEPAST
gave: V	(↑PRED)='GIVE⟨SUBJ,OBJ,OBJ2⟩' (↑TENSE)=SIMPLEPAST
had said: V	(↑PRED)='SAY⟨SUBJ,OBJ⟩' (↑TENSE)=PASTPERFECT
the: D	(↑PRED)='THE' (↑SPECTYPE)=DEF
about: P	(↑PRED)='ABOUT⟨OBJ⟩'
which: N	(↑PRED)='PRO' (↑PRONTYPE)=REL
John's: D	(↑PRED)='JOHN' (↑SPECTYPE)=POSS
many: D	(↑PRED)='MANY' (↑SPECTYPE)=QUANT
things: N	(↑PRED)='THINGS' (↑NUM)=PLURAL

STEP 2: c-structure

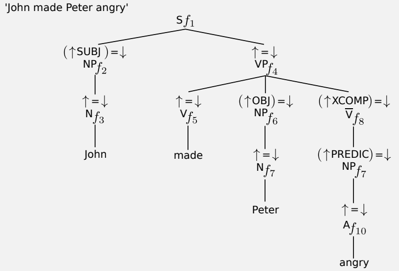
a. $\text{S} \rightarrow \begin{array}{c}\text{NP}\\ (\uparrow \text{SUBJ}) = \downarrow\end{array} \quad \begin{array}{c}\text{VP}\\ \uparrow=\downarrow\end{array}$

b. $\text{NP} \rightarrow \left\{ \begin{array}{c}\text{A}\\ \uparrow=\downarrow\end{array} \middle| \begin{array}{c}\text{N}\\ \uparrow=\downarrow\end{array} \right\}$

c. $\text{VP} \rightarrow \begin{array}{c}\text{V}\\ \uparrow=\downarrow\end{array} \quad \begin{array}{c}\text{NP}\\ (\uparrow \text{SUBJ}) = \downarrow\end{array} \quad \begin{array}{c}\overline{\text{V}}\\ (\uparrow \text{XCOMP}) = \downarrow\\ (\uparrow \text{XCOMP PRED}) = \text{'be} \langle \text{SUBJ, PREDIC} \rangle \text{'}\end{array}$

d. $\overline{\text{V}} \rightarrow \begin{array}{c}\overline{\text{NP}}\\ (\uparrow \text{PREDIC}) = \downarrow\end{array}$

STEP 3: f-structure



$f_1 = f_4 = f_5$
$(f_1 \text{SUBJ}) = f_2$
$f_2 = f_3$
$(f_4 \text{OBJ}) = f_6$
$f_6 = f_7$
$(f_4 \text{XCOMP}) = f_8$
$(f_4 \text{XCOMP PREDIC}) = \text{'be} \langle \text{SUBJ, PRED} \rangle \text{'}$
$(f_8 \text{PREDIC}) = f_9$
$f_9 = f_{10}$

STEP 4: unification

$$f_1, f_4, f_5 \begin{bmatrix} \text{PRED} & \text{'make} \langle \text{SUBJ, XCOMP} \rangle \text{'} \\ \text{TENSE} & \text{simplepast} \\ \text{SUBJ} & f_2,\ f_3 \begin{bmatrix} \text{PRED} & \text{'John'} \end{bmatrix} \\ \text{OBJ} & f_6,\ f_7 \begin{bmatrix} \text{PRED} & \text{'Peter'} \end{bmatrix} \\ \text{XCOMP} & f_8 \begin{bmatrix} \text{PRED} & \text{'be} \langle \text{SUBJ, PRED} \rangle \text{'} \\ \text{SUBJ} & \\ \text{PREDIC} & f_9,\ f_{10} \begin{bmatrix} \text{PRED} & \text{'angry'} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

Parsing with CCG

Outline

1 A-B categorial system

2 Lambek calculus

3 Extended Categorial Grammar
- Variation based on Lambek calculus
  - Abstract Categorial Grammar, Categorial Type Logic
- Variation based on Combinatory Logic
  - Combinatory Categorial Grammar (CCG)
  - Multi-modal Combinatory Categorial Grammar

- Categorial Grammar is
  - : a lexicalized theory of grammar along with other theories of grammar such as HPSG, TAG, LFG, …
  - : linguistically and computationally attractive
    $\longrightarrow$ language invariant combination rules, high efficient parsing

Main idea in CG and *application operation*

- All natural language consists of operators and of operands.
    - Operator (functor) and operand (argument)
    - Application: (operator(operand))
    - Categorial type: typed operator and operand

1. A-B categorial system

The product of the directional adaptation by Bar-Hillel (1953) of Ajdukiewicz's calculus of syntactic connection (Ajdukiewicz, 1935)

Definition 1 (AB categories).

Given *A*, a finite set of *atomic categories*, the **set of categories C** is the smallest set such that:

- $A \subseteq C$
- $(X \backslash Y), (X / Y) \in C$ *if* $X, Y \in C$

1. A-B categorial system

- **Categories** (type): primitive categories and derivative categories
  - Primitive: S for sentence, N for nominal phrase
  - Derivative: $S/N, N/N, (S\backslash N)/N, NN/N, S/S \ldots$
- Forward($>$) and backward ($<$) **functional application**

$$a.\ X/Y\ Y \Rightarrow X \qquad (>)$$
$$b.\ Y\ X\backslash Y \Rightarrow X \qquad (<)$$

1. A-B categorial system

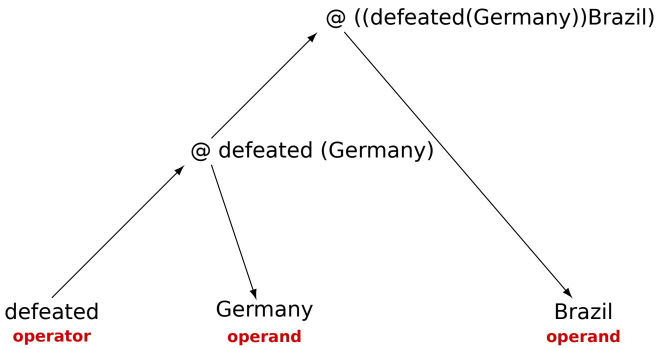
- **Calculus on types** in CG are analogue to **algebraic operations**

$$x/y\ y \rightarrow x \quad \approx \quad 3/5 * 5 = 3$$

$$\dfrac{\dfrac{\textit{Brazil}}{n} \qquad \dfrac{\dfrac{\textit{defeated}}{(s\backslash n)/n} \qquad \dfrac{\textit{Germany}}{n}}{s\backslash n}>}{s}<$$

1. A-B categorial system

Applicative tree of *Brazil defeated Germany*



Limitation of AB system

1 Relative construction

a. team$_i$ that t$_i$ defeated Germany
b. team$_i$ that Brazil defeated t$_i$

a’. that $(n\backslash n)/(s\backslash n)$ team [that]$_{(n\backslash n)/(s\backslash n)}$ [defeated Germany]$_{s\backslash n}$

b’. that $(n\backslash n)/(s/n)$ team [that]$_{(n\backslash n)/(s/n)}$ [Brazil defeated]$_{s/n}$

$$\begin{tikzcd}
\text{team} & \text{that} & \text{Brazil} & \text{defeated} \\
 & (n\backslash n)/(s/n) & n & (s\backslash n)/n \\
 & & {} & {} \arrow[l, "(?)"']
\end{tikzcd}$$

3 Many others complex phenomena

- Coordination, object extraction, phrasal verbs, ...

4 AB’s generative power is too weak – *context-free*

2. Lambek calculus (Lambek, 1958, 1961)

the calculus of **syntactic types**
still *context-free*

The axioms of Lambek calculus are the following:

1. $x \to x$
2. $(xy)z \to x(yz) \to (xy)z$ (the axioms 1, 2 with inference rules, 3, 4, 5)
3. If $xy \to z$ then $x \to z/y,$ if $xy \to z$ then $y \to x\backslash z;$
4. If $x \to z/y$ then $xy \to z,$ if $y \to x\backslash z$ then $xy \to z;$
5. If $x \to y$ and $y \to z$ then $x \to z.$

2. Lambek calculus (Lambek, 1958, 1961)

The rules obtained from the previous axioms are the following:

1. Hypothesis: if x and y are types, then x/y and $y\backslash x$ are types.
2. Application rules : $(x/y)y \rightarrow x, y(y\backslash x) \rightarrow x$
   ex: *Poor John works.*
3. *Associativity rule :* $(x\backslash y)/z \leftrightarrow x\backslash(y/z)$
   ex: John likes Jane.
4. Composition rules : $(x/y)(y/z) \rightarrow x/z, (x\backslash y)(y\backslash z) \rightarrow x\backslash z$
   ex: *He likes him.*
   $s/(n\backslash s)n\backslash s/n$
5. *Type-raising rules :* $x \rightarrow y/(x/y), x \rightarrow (y/x)\backslash y$

3. Combinatory Categorial Grammar

- Developed originally by M. Steedman (1988, 1990, 2000, ...)
- Combinatory Categorial Grammar (CCG) is a grammar formalism equivalent to Tree Adjoining Grammar, i.e.
  - it is lexicalized
  - it is parsable in polynomial time (See Vijay-Shanker and Weir, 1990)
  - it can capture cross-serial dependencies
- Just like TAG, CCG is used for grammar writing
- CCG is especially suitable for statistical parsing

3. Combinatory Categorial Grammar

- several of the **combinators which Curry and Feys** (1958) use to define **the λ-calculus** and applicative systems in general are of considerable syntactic interest (Steedman, 1988)
- The relationships of these combinators to terms of the λ-calculus are defined by the following equivalences (Steedman, 2000b):

$$a.\mathbf{B}fg \equiv \lambda x.f(gx) \text{ ... composition}$$
$$b.\mathbf{T}x \equiv \lambda f.fx \text{ ... type-raising}$$
$$c.\mathbf{S}fg \equiv \lambda x.fx(gx) \text{ ... substitution}$$

CCG categories

- Atomic categories: S, N, NP, PP, TV…
- Complex categories are built recursively from atomic categories and slashes
- Example complex categories for verbs:
    - intransitive verb: $S\backslash NP$ walked
    - transitive verb: $(S\backslash NP)/NP$ respected
    - ditransitive verb: $((S\backslash NP)/NP)/NP$ gave

Lexical categories in CCG

- An elementary syntactic structure – a lexical category – is assigned to each word in a sentence, eg:

  walked: $S\backslash NP$ ‘give me an NP to my left and I return a sentence’

- Think of the lexical category for a verb as a function: NP is the argument, S the result, and the slash indicates the direction of the argument

The typed lexicon item

- The CCG lexicon assigns categories to words, i.e. it specifies which categories a word can have.
- Furthermore, the lexicon specifies the semantic counterpart of the syntactic rules, e.g.:

  love $(S\backslash NP)/NP \lambda x \lambda y.\mathrm{loves}' xy$
- Combinatory rules determine what happens with the category and the semantics on combination

The typed lexicon item

- Attribution of types to lexical items: examples

Predicate

ex: is as an identificator of nominal

as an <u>*operator of predication*</u> from a nominal $\longrightarrow (S\backslash NP)/NP$

from an adjective $\longrightarrow (S\backslash NP)/(N/N)$

from an adverb $\longrightarrow (S\backslash NP)/(S\backslash NP)\backslash(S\backslash NP)$

from a preposition $\longrightarrow (S\backslash NP)/((S\backslash NP)\backslash(S\backslash NP)/NP)$

$$\begin{tikzcd}
 & \text{unary } (S\backslash NP) \\
\text{ex: verbs} \arrow[ur] \arrow[r] \arrow[dr] & \text{binary } (S\backslash NP)/NP \\
 & \text{ternary } (S\backslash NP)/NP/NP
\end{tikzcd}$$

The typed lexicon item

Adverbs

Adverb of verb

$(S\backslash NP)/(S\backslash NP)$

$(S\backslash NP)/NP/(S\backslash NP)/NP$

Adverb of adverb

$(S\backslash NP)/(S\backslash NP)/(S\backslash NP)/(S\backslash NP)$

$(S\backslash NP)/NP/(S\backslash NP)/NP/(S\backslash NP)/NP/(S\backslash NP)/NP$

Adverb of adjective

$(N/N)/(N/N)$

$(N\backslash N)/(N\backslash N)$

Adverb of proposition

S/S

Adverb: operator of determination of type (X/X)

The typed lexicon item

$$\begin{tikzcd}
 & \text{Preposition} \arrow[dl] \arrow[dr] & \\
\textbf{Prep. 1: constructor of adverbial phrase} & & \textbf{Prep. 2: constructor of adjectival phrase}
\end{tikzcd}$$

Prep. 1:
constructor of adverbial phrase

$(S\backslash NP)\backslash(S\backslash NP)/NP$
$(S/S)/NP$
$(S/S)/N$

Prep. 2:
constructor of adjectival phrase

$(N\backslash N)/NP$
$(N\backslash N)/N$

Preposition: constructor of determination of type (X/X)

Dictionary of typed words

Syntactic categories	Syntactic types	Lexical entries
Nom.	N	*Olivia, apple...*
Completed nom.	NP	*an apple, the school*
Pron.	NP	*She, he...*
Adj.	$(N/N), (N\backslash N)$	***pretty** woman,...*
Adv.	$(N/N)/(N/N)$,	***very** delicious,...*
	$(S\backslash NP)\backslash(S\backslash NP)...$	
Vb	$(S\backslash NP), (S\backslash NP)/NP...$	*run, give...*
Prep.	$(S\backslash NP)\backslash(S\backslash NP)/NP$	*run in the park,*
	$(NP\backslash NP)/NP...$	*book of John, ...*
Relative	$(S\backslash NP)/S...$	*I believe that...*

Combinatorial categorial rules

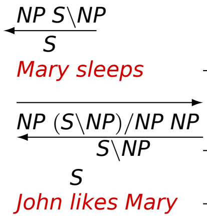
- Functional application $(>, <)$
- Functional composition $(> \mathbf{B}, < \mathbf{B})$
- Type-raising $(< \mathbf{T}, > \mathbf{T})$
- Distribution $(< \mathbf{S}, > \mathbf{S})$
- Coordination $(< \Phi, > \Phi)$

Functional application (FA)

$X/Y : f \quad Y : a \Rightarrow X : fa$ (forward functional application, $>$)

$Y : a \quad X\backslash Y : f \Rightarrow X : fa$ (backward functional application, $<$)

- Combine a function with its argument:



Mary sleeps $\rightarrow$ (sleeps (Mary))

$\rightarrow$ (likes (Mary))

John likes Mary $\rightarrow$ ((likes (Mary))John)

- Direction of the slash indicates position of the argument with respect to the function

Derivation in CCG

- The combinatorial rule used in each derivation step is usually indicated on the right of the derivation line
- Note especially what happens with the semantic information

$$\dfrac{\dfrac{\mathit{John}}{\mathit{NP} : \mathit{John}'} \quad \dfrac{\dfrac{\mathit{loves}}{(S\backslash \mathit{NP})/\mathit{NP} : \lambda x \lambda y.\mathit{loves}'xy} \quad \dfrac{\mathit{Mary}}{\mathit{NP} : \mathit{Mary}'}}{S\backslash \mathit{NP} : \lambda y.\mathit{loves}'\mathit{Mary}'y} >}{S : \mathit{loves}'\mathit{Mary}'\mathit{John}'} <$$

Function composition (FC)

Generalized forward composition $(> \boldsymbol{Bn})$

$$X/Y : f \qquad Y/Z : g \quad \Rightarrow_B \quad X/Z : \lambda x.f(gx) \qquad (> B)$$

- Functional composition composes two complex categories (two functions):

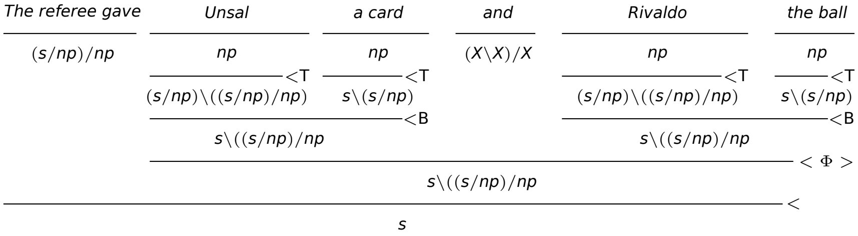
$$(S\backslash NP)/PP \quad (PP/NP) \Rightarrow_B (S\backslash NP)/NP$$
$$S/(S\backslash NP) \quad (S\backslash NP)/NP \Rightarrow_B S/NP$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{birds}}{NP}}{S/(S\backslash NP)}{>T} \qquad \dfrac{\text{like}}{(S\backslash NP)/NP}}{S/NP}{>B} \qquad \dfrac{\text{bugs}}{NP}}{S}{>}$$

Function composition (FC)

Generalized backward composition $(< \boldsymbol{Bn})$

$$Y\backslash Z : f \qquad X\backslash Y : g \Rightarrow_B X\backslash Z : x.f(gx) \qquad (< B)$$



Type-raising (T)

Forward type-raising $(> \boldsymbol{T})$

$$X : a \Rightarrow T/(T\backslash X) : \lambda f.fa \qquad (> T)$$

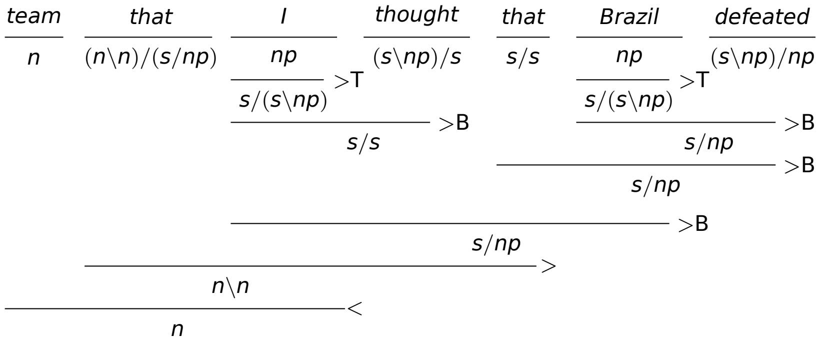
- Type-raising turns an argument into a function (e.g. for case assignment)

$$NP \Rightarrow S/(S\backslash NP) \text{ (nominative)}$$

$$\cfrac{\cfrac{birds}{NP} \qquad \cfrac{fly}{S\backslash NP}}{S}{<} \qquad\qquad\qquad \cfrac{\cfrac{\cfrac{birds}{NP}}{S/(S\backslash NP)}{>T} \qquad \cfrac{fly}{S\backslash NP}}{S}{>}$$

- This must be used *e.g. in the case of WH-questions*

Example of functional composition $(> B)$ and type-raising (T)



Example of functional composition $(> B)$ and type-raising (T)

Backward type-raising $(< \boldsymbol{T})$

$$X : a \Rightarrow T\backslash(T/X) : \lambda f.fa \qquad (< T)$$

- Type-raising turns an argument into a function (e.g. for case assignment)

$$NP \Rightarrow (S\backslash NP)\backslash((S\backslash NP)/NP) \text{ (accusative)}$$

$$\begin{array}{cccccc}
\textit{The referee gave} & \textit{Unsal} & \textit{a card} & \textit{and} & \textit{Rivaldo} & \textit{the ball} \\ \hline
(s/np)/np & np & np & (X\backslash X)/X & np & np \\ \cline{2-3} \cline{5-6}
 & (s/np)\backslash((s/np)/np) \; {<}\mathrm{T} & s\backslash(s/np) \; {<}\mathrm{T} & & (s/np)\backslash((s/np)/np) \; {<}\mathrm{T} & s\backslash(s/np) \; {<}\mathrm{T} \\ \cline{2-3} \cline{5-6}
 & \multicolumn{2}{c}{s\backslash((s/np)/np) \; {<}\mathrm{B}} & & \multicolumn{2}{c}{s\backslash((s/np)/np) \; {<}\mathrm{B}} \\ \cline{2-6}
 & \multicolumn{5}{c}{s\backslash((s/np)/np) \quad < \Phi >} \\ \hline
\multicolumn{6}{c}{s \quad <}
\end{array}$$

Coordination (&)

$$X\ CONJ\ X \Rightarrow_{\Phi} X \qquad (Coordination(\Phi))$$

$$\begin{array}{cccccc}
\textit{give} & \textit{a dog} & \textit{a bone} & \textit{and} & \textit{a policeman} & \textit{a flower}\\
\hline
 & {<}\mathrm{T} & {<}\mathrm{T} & & {<}\mathrm{T} & {<}\mathrm{T}\\
(VP/NP)/NP & (VP/NP)\backslash((VP/NP)/NP) & VP\backslash(VP/NP) & \mathrm{conj} & (VP/NP)\backslash((VP/NP)/NP) & VP\backslash(VP/NP)\\
\hline
 & \multicolumn{2}{c}{{<}\mathrm{B}} & & \multicolumn{2}{c}{{<}\mathrm{B}}\\
 & \multicolumn{2}{c}{VP\backslash((VP/NP)/NP)} & & \multicolumn{2}{c}{VP\backslash((VP/NP)/NP)}\\
\hline
 & \multicolumn{5}{c}{<\&>}\\
 & \multicolumn{5}{c}{VP\backslash((VP/NP)/NP)}\\
\hline
\multicolumn{6}{c}{<}\\
\multicolumn{6}{c}{VP}
\end{array}$$

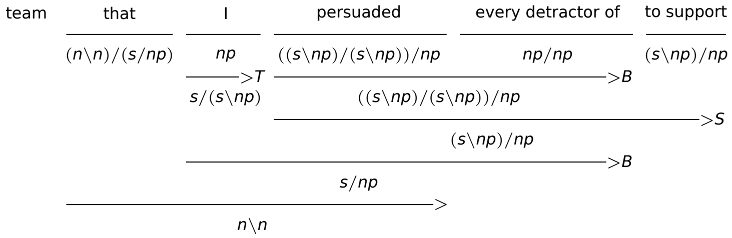
Substitution (S)

Forward substitution ($> S$)

$$(X/Y)/Z \; Y/Z \Rightarrow_S X/Z$$

- Application to parasitic gap such as the following:

a. team that I persuaded every detractor of to support



Substitution (S)

Backward crossed substitution ($< \boldsymbol{S}\times$)

$$Y/Z \; (X\backslash Y)/Z \Rightarrow_S X/Z$$

- Application to parasitic gap such as the following:

a. *John watched without enjoying the game between Germany and Paraguay.*

b. game that John watched without enjoying

game that John [watched]$_{(s\backslash np)/np}$ [without enjoying]$_{((s\backslash np)\backslash(s\backslash np))/np}$

$$\mathit{game} \quad \dfrac{\dfrac{\mathit{that}}{(n\backslash n)/(s/np)} \qquad \dfrac{\dfrac{\dfrac{\mathit{John}}{np}}{s/(s\backslash np)}{>}\mathrm{T} \qquad \dfrac{\dfrac{\mathit{watched}}{(s\backslash np)/np} \qquad \dfrac{\mathit{without\ enjoying}}{((s\backslash np)\backslash(s\backslash np))/np}}{(s\backslash np)/np}{<}\mathrm{S}_\times}{s/(s\backslash np)}{>}\mathrm{B}}{n\backslash n}{>}$$

Limit on possible rules

- The Principle of Adjacency:

Combinatory rules may only apply to entities which are linguistically realised and adjacent.

- The Principle of Directional Consistency:

All syntactic combinatory rules must be consistent with the directionality of the principal function. ex: $X\backslash Y\ Y \not=> X$

- The Principle of Directional Inheritance:

If the category that results from the application of a combinatory rule is a function category, then the slash defining directionality for a given argument in that category will be the same as the one defining directionality for the corresponding arguments in the input functions. ex: $X/Y\ Y/Z \not=> X\backslash Z$.

Semantic in CCG

- CCG offers a syntax-semantics interface.
- The lexical categories are augmented with an explicit identification of their semantic interpretation and the rules of functional application are accordingly expanded with an explicit semantics.
- Every syntactic category and rule has a semantic counterpart.
- The lexicon is used to pair words with syntactic categories and semantic interpretations:

$$\textit{love}\ (S\backslash NP)/NP \Rightarrow \lambda x \lambda y.loves' xy$$

Semantic in CCG

- The semantic interpretation of all combinatory rules is fully determined by the **Principle of Type Transparency**:
  - **Categories**: All syntactic categories reflect the semantic type of the associated logical form.
  - **Rules**: All syntactic combinatory rules are type-transparent versions of one of a small number of semantic operations over functions including application, composition, and type-raising.

Semantic in CCG

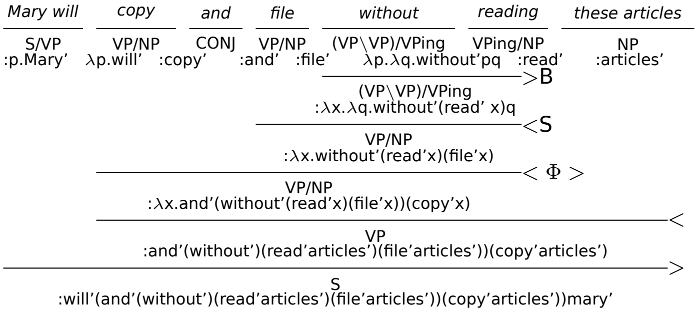
proved $:= (S\backslash NP_{3s})/NP : \lambda x \lambda y.prove'xy$

- the semantic type of the reduction is the same as its syntactic type, here functional application.

$$\begin{array}{ccc}
\textit{Marcel} & \textit{proved} & \textit{completeness} \\
\hline
NP_{3sm} : marcel' & (S\backslash NP_{3s})/NP : \lambda x\lambda y.prove'xy & NP : completeness' \\
\cline{2-3}
 & \multicolumn{2}{c}{S\backslash NP_{3s} : \lambda y.prove'completeness'y \quad >} \\
\hline
\multicolumn{3}{c}{S : prove'completeness'marcel' \quad <}
\end{array}$$

Semantic in CCG

CCG with semantics : *Mary will copy and file without reading these articles*



Mary will
S/VP
:p.Mary′ λp.will′
copy
VP/NP
:copy′
and
CONJ
:and′
file
VP/NP
:file′
without
(VP\VP)/VPing
λp.λq.without′pq
reading
VPing/NP
:read′
these articles
NP
:articles′
>B
(VP\VP)/VPing
:λx.λq.without′(read′ x)q
<S
VP/NP
:λx.without′(read′x)(file′x)
< Φ >
VP/NP
:λx.and′(without′(read′x)(file′x))(copy′x)
<
VP
:and′(without′)(read′articles′)(file′articles′))(copy′articles′)
>
S
:will′(and′(without′)(read′articles′)(file′articles′))(copy′articles′))mary′

Parsing a sentence in CCG

Step 1: tokenization

Step 2: tagging the concatenated lexicon

Step 3:

- calculate on types attributed to the concatenated lexicons by applying the adequate combinatorial rules
- eliminate the applied combinators (we will see how to do on next week)

Step 4: finding the parsing results presented in the form of an operator/operand structure (predicate -argument structure)

Parsing a sentence in CCG

Example: *I requested and would prefer musicals*

STEP 1 : tokenization/lemmatization $\rightarrow$ *ex) POS Tagger, tokenizer, lemmatizer*

a. *I-requested-and-would-prefer-musicals*
b. *I-request-ed-and-would-prefer-musical-s*

STEP 2 : tagging the concatenated expressions $\rightarrow$ *ex) Supertagger, Inventory of typed words*

I	NP
Requested	$(S\backslash NP)/NP$
And	$CONJ$
Would	$(S\backslash NP)/VP$
Prefer	VP/NP
musicals	NP

Parsing a sentence in CCG

STEP 3 : categorial calculus

a. apply the type-raising rules $\longrightarrow$ *Subject Type-raising* $(>T)$
$NP : a \Rightarrow T/(T\backslash NP) : Ta$

b. apply the functional composition rules $\longrightarrow$ *Forward Composition:* $(>B)$
$X/Y : f \quad Y/Z : g \Rightarrow X/Z : Bfg$

c. apply the coordination rules $\longrightarrow$ *Coordination:* $(<\&>)$
$X \text{ conj } X \Rightarrow X$

	I-	requested-	and-	would-	prefer-	musicals	
1/	NP	$(S\backslash NP)/NP$	CONJ	$(S\backslash NP)/VP$	VP/NP	NP	
2/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$	CONJ	$(S\backslash NP)/VP$	VP/NP	NP	$(>T)$
3/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$	CONJ	$(S\backslash NP)/NP$		NP	$(>B)$
4/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$				NP	$(>\Phi)$
5/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$				NP	$(>B)$
6/	S/NP					NP	$(>)$
7/				**S**			

Parsing a sentence in CCG

STEP 4 : semantic representation (predicate-argument structure)

I requested and would prefer musicals

1/ :i' :request' :and' : will' :prefer' : musicals'

2/ :λf.f I'

3/ : λx.λy.will'(prefer'x)y

4/ : λxλy.and'(will'(prefer'x)y)(request'xy)

5/ : λxλy.and'(will'(prefer'x)y)(request'xy)

6/ :λy.and'(would'(prefer' musicals')y)(request' musicals' y)

7/S: and'(will'(prefer' musicals') i')(request' musicals' i')

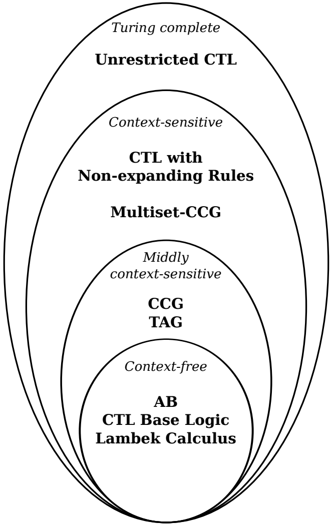
Variation of CCG : Multi-modal CCG (Baldridge, 2002)

- Modalized CCG system
- Combination of Categorial Type Logic (CTL, Morrill, 1994; Moortgat, 1997) into the CCG (Steedman, 2000)
- Rules restrictions by introducing the modalities: *, x, •, $\Diamond$
- Modalized functional composition rules

$$\begin{array}{llcl} (> B) & X/_{\Diamond}Y \quad Y/_{\Diamond}Z & \Rightarrow & X/_{\Diamond}Z \\ (< B) & X\backslash_{\Diamond}Y \quad Y\backslash_{\Diamond}Z & \Rightarrow & X\backslash_{\Diamond}Z \end{array}$$

- Invite you to read the paper “*Multi-Modal CCG*” of (Baldridge and M.Kruijff, 2003)

The positions of several formalisms on the Chomsky hierarchy



Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

Parsing with HPSG

Overview on syntactic formalisms

- Unification based grammars
  - : HPSG, LFG, TAG, UCG...
- Dependency based grammars
  - : Tesnière model; Meaning-Text of Mel’čuk...

Heritage of HPSG

- GPSG – Generalized Phrase-Structure Grammar (Gerald Gazdar)
  - linear order/hierarchy order feature structure for representation of information
- LFG
  - Lexicon contains
  - Lexical rules
- CG
  - Subcategorization

Key points of HPSG

- Monostratal theory without derivation
  - Sharing a given information without movement and transformation
  - One representation for different levels of analysis : phonology, syntax, semantic
  - Constraint-based analysis
- Unification of given information
- Computational formalism

Syntactic representation in HPSG

Typed feature structure

- consists of a couple **“attribute/value”**
- the types are organized into a hierarchy
  - ex: sign>phrase, case>nominative
- feature structure is a directed acyclic graph (DAG), with arcs representing features going between values

Features

- Basic element of structure in HPSG
- Should be appropriate to a type
- Most frequently used features
  - PHON
  - SYNSEM
  - LOC/NON-LOC
  - CAT
  - CONTEXT
  - CONTENT
  - HEAD
  - SUJ
  - COMPS
  - S-ARG

Types

- Types are attributed to features -> typed features
  - sign
  - synsem
  - head
  - phrase
  - content
  - Index
  - ....
- Each of these feature values is itself a complex object:
  - The type sign has the features PHON and SYNSEM appropriate for it
  - The feature SYNSEM has a value of type synsem
  - This type itself has relevant features (LOCAL and NONLOCAL)

Types

- sign is the basic type in HPSG used to describe lexical items (of type word) and phrases (of type phrase).
- All signs carry the following two features:
  - PHON encodes the phonological representation of the sign
  - SYNSEM syntax and semantics

$$_{\textit{sign}}\begin{bmatrix} \text{PHON} & \mathrm{list(phon\text{-}string)} \\ \text{SYNSEM} & \mathrm{synsem} \end{bmatrix}$$

Types

- In attribute-value matrix (AVM) form, here is the skeleton of an object:

$$\begin{bmatrix} \textit{sign} & \\ \mathrm{PHON} & \textit{list}(\mathrm{PHON}) \\ \mathrm{SYNSEM} & \begin{bmatrix} \textit{synsem} & \\ \mathrm{LOCAL} & \textit{local} \\ \mathrm{NON\text{-}LOCAL} & \textit{non-local} \end{bmatrix} \\ \mathrm{DTRS} & \textit{list}(\mathrm{SIGN}) \end{bmatrix}$$

Structure of signs in HPSG

- *synsem introduces the features LOCAL and NONLOCAL*
- local introduces CATEGORY (CAT), CONTENT (CONT) and CONTEXT(CONX)
- non-local will be discussed in connection with unbounded dependencies
- category includes the syntactic category and the grammatical argument of the word/phrase

Description of an object in HPSG:

lexical sign and phrasal sign

$$\begin{tikzcd}
 & \textit{sing}\begin{bmatrix}\text{PHON} & \mathrm{list(phon\text{-}string)}\\ \text{SYNSEM} & \mathrm{synsem}\end{bmatrix} \arrow[dl, no head] \arrow[dr, no head] & \\
\textit{word} & & \textit{phrase}\begin{bmatrix}\text{DTRS} & \mathrm{constituent\text{-}struc}\end{bmatrix}
\end{tikzcd}$$

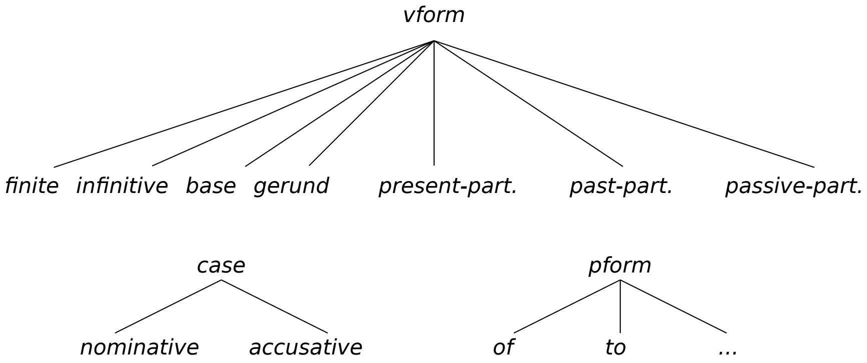
$$\textit{synsem}\begin{bmatrix}\text{LOCAL} & \mathrm{local}\\ \text{NON-LOCAL} & \mathrm{non\text{-}local}\end{bmatrix} \qquad \textit{local}\begin{bmatrix}\text{CATEGORY} & \mathrm{category}\\ \text{CONTENT} & \mathrm{content}\\ \text{CONTEXT} & \mathrm{context}\end{bmatrix} \qquad \textit{category}\begin{bmatrix}\text{HEAD} & \mathrm{head}\\ \text{VAL} & \dots\\ \dots & \dots\end{bmatrix}$$

CATEGORY

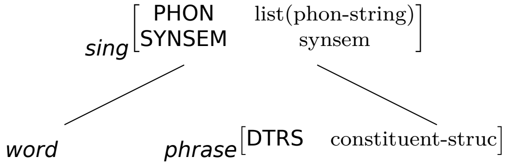
- **CATEGORY** encode the sign’s syntactic category
  - Given via the feature **[HEAD head]**, where head is the supertype for noun, verb, adjective, preposition, determiner, marker; each of these types selects a particular set of head features
  - Given via the feature **[VALENCE ...]**, possible to combine the signs with the other signs to a larger phrases

$$\left[\text{SYNSEM|LOC|CAT|VALENCE} \quad {}_{\textit{valence}}\begin{bmatrix}\text{SUBJECT} & \mathrm{list}(\mathrm{synsem})\\ \text{SPECIFIER} & \mathrm{list}(\mathrm{synsem})\\ \text{COMPLEMENTS} & \mathrm{list}(\mathrm{synsem})\end{bmatrix}\right]$$

Sub-categorization of head type



Description of an object in HPSG

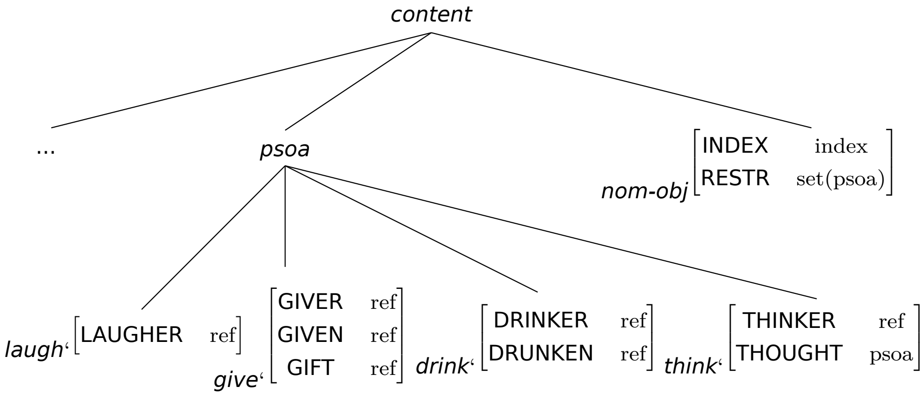


$$\textit{synsem}\begin{bmatrix} \text{LOCAL} & \text{local} \\ \text{NON-LOCAL} & \text{non-local} \end{bmatrix} \quad \textit{local}\begin{bmatrix} \text{CATEGORY} & \text{category} \\ \text{CONTENT} & \text{content} \\ \text{CONTEXT} & \text{context} \end{bmatrix} \quad \textit{category}\begin{bmatrix} \text{HEAD} & \text{head} \\ \text{VAL} & \ldots \\ \ldots & \ldots \end{bmatrix}$$

Semantic representation: CONTENT (& CONTEXT) feature

- Semantic interpretation of the sign is given as the value to **CONTENT**
  - **nominal-object**: an individual/entity (or a set of them), associated with a referring index, bearing agreement features → INDEX, RESTR
  - **Parameterized-state-of-affairs** (psoa): a partial situate; an event relation along with role names for identifying the participants of the event→ BACKGR
  - **quantifier**: some, all, every, a, the, . . .
- Note: many of these have been reformulated by “Minimal Recursion Semantics (MRS)” which allows underspecification of quantifier scopes.

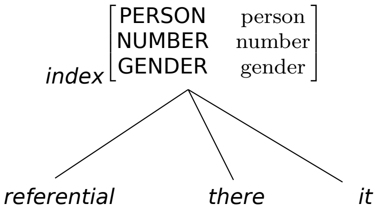
Sub-categorization of content type

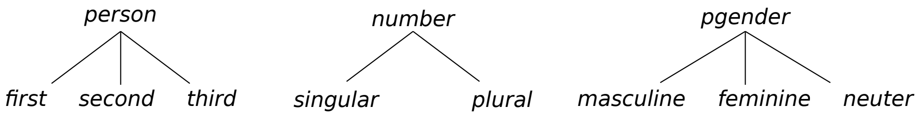


Note:

Semantic restriction on the index are represented as a value of RESTR. RESTR is an attribute of a nominal object. The value of RESTR is a set of psoa. In turn, RESTR has the attribute of REL whose value can either be referential indices or psoas.

Sub-categorization of **index** type

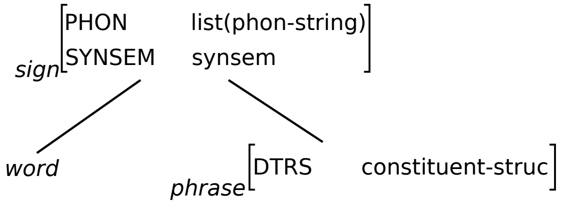




Lexical input of *She*

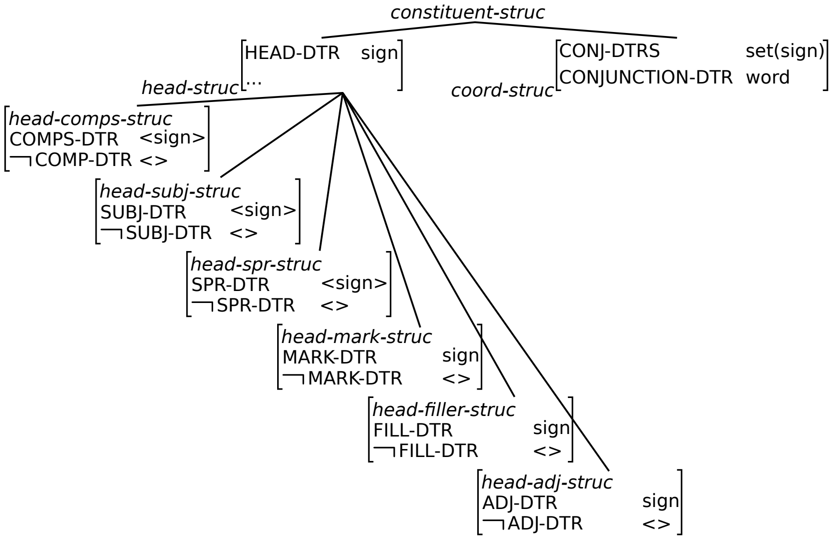
$$
{}_{\textit{word}}\begin{bmatrix}
\text{PHON} & \langle \text{she} \rangle \\
\text{SYNSEM} & {}_{\textit{synsem}}\begin{bmatrix}
\text{LOCAL} & {}_{\textit{local}}\begin{bmatrix}
\text{CATEGORY} & {}_{\textit{cat}}\begin{bmatrix}
\text{HEAD} & {}_{\textit{noun}}\begin{bmatrix} \text{CASE} & \text{nom} \end{bmatrix} \\
\text{VALENCE} & {}_{\textit{val}}\begin{bmatrix} \text{SUBJ} & \langle\rangle \\ \text{COMPS} & \langle\rangle \\ \text{SPR} & \langle\rangle \end{bmatrix}
\end{bmatrix} \\
\text{CONTENT} & {}_{\textit{ppro}}\begin{bmatrix}
\text{INDEX} & \boxed{1}\ {}_{\textit{ref}}\begin{bmatrix} \text{PER} & \text{3rd} \\ \text{NUM} & \text{sing} \\ \text{GEND} & \text{fem} \end{bmatrix} \\
\text{RESTR} & \{\}
\end{bmatrix} \\
\text{CONTEXT} & {}_{\textit{context}}\begin{bmatrix}
\text{BACKGR} & \left\{ {}_{\textit{psoa}}\begin{bmatrix} \text{RELN} & \text{female} \\ \text{INST} & \boxed{1} \end{bmatrix} \right\}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Lexical input of *She*

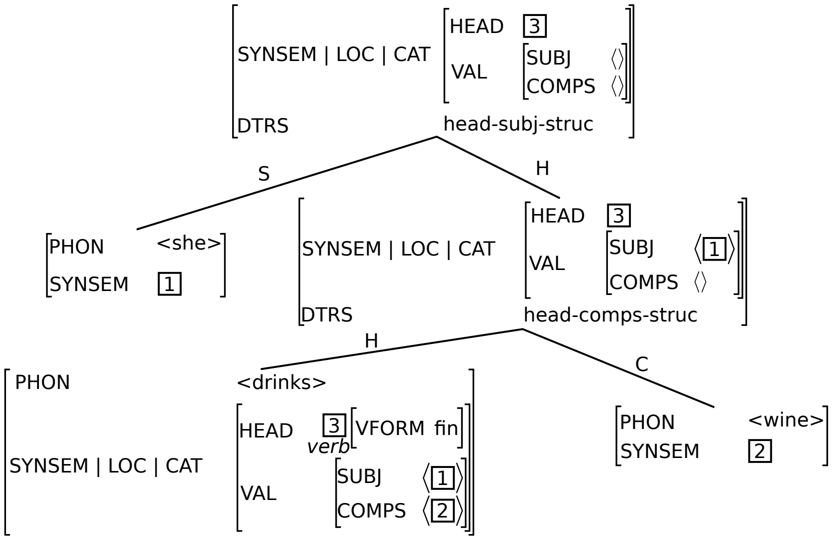


- Each *phrase* has a DTRS attribute which has a constituent-structure value
- This DTRS value corresponds to what we view in a tree as daughters (with additional grammatical role information, e.g. adjunct, complement, etc.)
- By distinguishing different kinds of constituent-structures, we can define different kinds of constructions in a language

Structure of phrase



head-subject/complement structure



Questions! (1)

- How exactly did the last example work?
  - *drink* has head information specifying that it is a finite verb and subcategories for a subject and an object
    - The head information gets percolated up (the HEAD feature principle)
    - The valence information gets “checked off” as one moves up in the tree (the VALENCE principle)
- Such principles are treated as linguistic universals in HPSG

HEAD-feature principle

- The value of the HEAD feature of any headed phrase is token-identical with the HEAD value of the head daughter

$$_{\textit{phrase}}\begin{bmatrix}\text{DTRS} & \textit{head-struc}\end{bmatrix} \longrightarrow \begin{bmatrix}\text{SYNSEM} \mid \text{LOC} \mid \text{CAT} \mid \text{HEAD} & \boxed{1} \\ \text{DTRS} \mid \text{HEAD-DTR} \mid \text{SYNSEM} \mid \text{LOC} \mid \text{CAT} \mid \text{HEAD} & \boxed{1}\end{bmatrix}$$

VALENCE principle

- *In a headed phrase, for each valence feature F, the F value of the head daughter is the concatenation of the phrase's F value with the list of F-DTR's SYNSEM* (Pollard and Sag, 1994:348)

$$\begin{bmatrix} \textit{phrase} & \\ \text{SS} \mid \text{LOC} \mid \text{CAT} \mid \text{VAL} & \begin{bmatrix} \text{SUBJ} & [a] \\ \text{COMPS} & [b] \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \text{HEAD-DTR} & \left\langle \begin{bmatrix} \text{SS} \mid \text{LOC} \mid \text{CAT} \mid \text{VAL} \begin{bmatrix} \text{SUBJ} & [1] \oplus [a] \\ \text{COMPS} & [2],\ldots,[n] \oplus [b] \end{bmatrix} \end{bmatrix} \right\rangle \\ \text{SUBJ-DTR} & \left\langle \begin{bmatrix} \text{SS } [1] \end{bmatrix} \right\rangle \\ \text{COMP-DTR} & \left\langle \begin{bmatrix} \text{SS } [2] \end{bmatrix} ,\ldots, \begin{bmatrix} \text{ss}[n] \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$$

- Note: **Valence Principle** constrains the way in which information is shared between phrases and their head daughters.
  - F can be any one of SUBJ, COMPS, SPR
  - When the F-DTR is empty, the F valence feature of the head daughter will be copied to the mother phrase

Questions! (2)

- Note that agreement is handled neatly, simply by the fact that the SYNSEM values of a word's daughters are token-identical to the items on the VALENCE lists
- How exactly do we decide on a syntactic structure?
- Why the subject is checked off at a higher point in the tree?

Immediate Dominance (ID) Principle

- Every headed phrase must satisfy exactly one of the ID schemata
  - The exact inventory of valid ID schemata is language specific
  - We will introduce a set of ID schemata for English

Immediate Dominance (ID) Schemata

$$\textit{phrase}\begin{bmatrix}\text{DTRS} & \textit{head-struc}\end{bmatrix} \longrightarrow \begin{bmatrix}\text{SS} \mid \text{LOC} \mid \text{CAT} \mid \text{VAL} \mid \text{COMPS} & \langle\rangle \\ \text{DTRS} & \textit{head-subj-struc}\end{bmatrix} \qquad \text{(head-subject)}$$

$$\vee \begin{bmatrix}\text{DTRS} & \textit{head-comps-struc}\end{bmatrix} \qquad \text{(head-complement)}$$

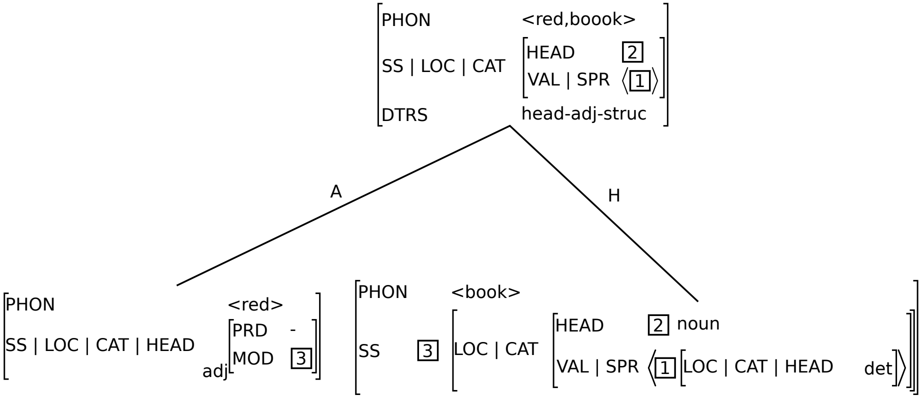
$$\vee \begin{bmatrix}\text{SS} \mid \text{LOC} \mid \text{CAT} \mid \text{VAL} \mid \text{COMPS} & \langle\rangle \\ \text{DTRS} & \textit{head-spr-struc}\end{bmatrix} \qquad \text{(head-specifier)}$$

$$\vee \begin{bmatrix}\text{DTRS} & \begin{bmatrix}\textit{head-marker-struc} & \\ \text{MARK-DTR} \mid \text{SS} \mid \text{LOC} \mid \text{CAT} \mid \text{HEAD} & \textit{marker}\end{bmatrix}\end{bmatrix} \qquad \text{(head-marker)}$$

$$\vee \begin{bmatrix}\text{DTRS} & \begin{bmatrix}\textit{head-adj-struc} & \\ \text{ADJ-DTR} \mid \text{SS} \mid \text{LOC} \mid \text{CAT} \mid \text{HEAD} \mid \text{MOD} & \boxed{1} \\ \text{HEAD-DTR} \mid \text{SS} & \boxed{1}\end{bmatrix}\end{bmatrix} \qquad \text{(head-adjunct)}$$

$$\vee \ \ldots$$

head-adjunct structure


PHON <red,boook>
SS | LOC | CAT HEAD 2 VAL | SPR ⟨1⟩
DTRS head-adj-struc
A
H
PHON <red>
SS | LOC | CAT | HEAD adj PRD - MOD 3
PHON <book>
SS 3 LOC | CAT HEAD 2 noun VAL | SPR ⟨1 LOC | CAT | HEAD det⟩

Semantic principle

- The CONTENT value of a headed phrase is token identical to the CONTENT value of the semantic head daughter
- The semantic head daughter is identified as
  - The ADJ-DTR in a head-adjunct phrase
  - The HEAD-DTR in other headed phrases

$$
{}_{\textit{phrase}}\begin{bmatrix}\text{DTRS} & \text{head-struc}\end{bmatrix} \longrightarrow \begin{bmatrix}\text{SYNSEM} \mid \text{LOC} \mid \text{CONT} & \boxed{1} \\ \text{DTRS} & \begin{bmatrix}\text{head-adj-struc} \\ \text{ADJ-DTR} \mid \text{SYNSEM} \mid \text{LOC} \mid \text{CONT} & \boxed{1}\end{bmatrix}\end{bmatrix} \quad \text{(head-adjunct)}
$$

$$
\vee \begin{bmatrix}\text{SYNSEM} \mid \text{LOC} \mid \text{CONT} & \boxed{1} \\ \text{DTRS} & \begin{bmatrix}\neg\text{head-adj-struc} \\ \text{HEAD-DTR} \mid \text{SYNSEM} \mid \text{LOC} \mid \text{CONT} & \boxed{1}\end{bmatrix}\end{bmatrix} \quad \text{(non-head-adjunct)}
$$

Example 2

***Kim** likes bagels*

$$\begin{bmatrix} \textit{word} & \\ \text{PHON} & \langle \text{Kim} \rangle \\ \text{SYNSEM} & \begin{bmatrix} \text{LOCAL} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{noun} & \\ \text{ARG} & \textit{3sg} \end{bmatrix} \\ \text{SUBJ} & \langle\rangle \\ \text{SPR} & \langle\rangle \\ \text{COMPS} & \langle\rangle \\ \text{ARG-ST} & \langle\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} & \boxed{1} \\ \text{KEY} & \boxed{2} \\ \text{RELS} & \left\langle \boxed{2} \begin{bmatrix} \textit{named\_rel} & \\ \text{INST} & \boxed{1} \\ \text{ARG} & \text{Kim} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Example 2

*Kim **likes(1)** bagels*

$$
\begin{bmatrix}
\textit{word} & \\
\text{PHON} & \langle \text{likes} \rangle \\
\text{SYNSEM} & \begin{bmatrix}
\textit{synsem} & \\
\text{LOCAL} & \begin{bmatrix}
\textit{local} & \\
\text{CAT} & \begin{bmatrix}
\textit{category} & \\
\text{HEAD} & \begin{bmatrix} \textit{verb} & \\ \text{FORM} & \textit{fin} \end{bmatrix} \\
\text{SUBJ} & \left\langle \boxed{1} \begin{bmatrix}
\textit{synsem} & \\
\text{LOCAL} & \begin{bmatrix}
\textit{local} & \\
\text{CAT} & \begin{bmatrix}
\textit{category} & \\
\text{HEAD} & \begin{bmatrix} \textit{noun} & \\ \text{ARG} & \textit{3sg} \end{bmatrix} \\
\text{SUBJ} & \langle\rangle \\
\text{SPR} & \langle\rangle \\
\text{COMPS} & \langle\rangle
\end{bmatrix} \\
\text{CONT} & \begin{bmatrix} \textit{content} & \\ \text{INDEX} & \boxed{4} \end{bmatrix}
\end{bmatrix}
\end{bmatrix} \right\rangle \\
\text{SPR} & \langle\rangle \\
\text{COMPS} & \left\langle \boxed{2} \begin{bmatrix}
\textit{synsem} & \\
\text{LOCAL} & \begin{bmatrix}
\textit{local} & \\
\text{CAT} & \begin{bmatrix}
\textit{category} & \\
\text{HEAD} & \textit{noun} \\
\text{SUBJ} & \langle\rangle \\
\text{SPR} & \langle\rangle \\
\text{COMPS} & \langle\rangle
\end{bmatrix} \\
\text{CONT} & \begin{bmatrix} \textit{content} & \\ \text{INDEX} & \boxed{5} \end{bmatrix}
\end{bmatrix}
\end{bmatrix} \right\rangle \\
\text{ARG-ST} & \langle \boxed{1}, \boxed{2} \rangle
\end{bmatrix} \\
\text{CONT} & \begin{bmatrix}
\textit{content} & \\
\text{INDEX} & \boxed{3} \\
\text{KEY} & \boxed{6} \\
\text{RELS} & \left\langle \boxed{6} \begin{bmatrix} \textit{like\_rel} & \\ \text{EVENT} & \boxed{3} \\ \text{ARG1} & \boxed{4} \\ \text{ARG2} & \boxed{5} \end{bmatrix} . \begin{bmatrix} \textit{t\_overlap\_rel} & \\ \text{ARG1} & \boxed{3} \\ \text{ARG2} & \textit{now} \end{bmatrix} \right\rangle
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Example 2

Kim ***likes(2)*** *bagels*

$$\begin{bmatrix} \textit{word} & \\ \text{PHON} & \langle\text{likes}\rangle \\ \text{SYNSEM} & \begin{bmatrix}\text{LOCAL} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix}\textit{verb} & \\ \text{FORM} & \textit{fin}\end{bmatrix} \\ \text{SUBJ} & \langle \boxed{1} \rangle \\ \text{SPR} & \langle\ \rangle \\ \text{COMPS} & \langle \boxed{2}\rangle \\ \text{ARG-ST} & \langle \boxed{1}\ \text{NP}[\textit{3sg}]\boxed{4}, \boxed{2}\text{NP}\boxed{5}\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} & \boxed{3} \\ \text{RELS} & \left\langle \begin{bmatrix}\textit{like\_rel} & \\ \text{EVENT} & \boxed{3} \\ \text{ARG1} & \boxed{4} \\ \text{ARG2} & \boxed{5}\end{bmatrix}, \begin{bmatrix}\textit{t-overlap\_rel} & \\ \text{ARG1} & \boxed{3}\\ \text{ARG2} & \textit{now}\end{bmatrix}\right\rangle \end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Example 2

*Kim likes **bagels***

$$
\begin{bmatrix}
\textit{word} & \\
\text{PHON} & \langle \text{bagels} \rangle \\
\text{SYNSEM} & \begin{bmatrix}
\text{LOCAL} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix}
\text{HEAD} & \begin{bmatrix} \textit{noun} & \\ \text{AGR} & \textit{pl} \end{bmatrix} \\
\text{SUBJ} & \langle \rangle \\
\text{SPR} & \langle (\boxed{3}) \rangle \\
\text{COMPS} & \langle \rangle \\
\text{ARG-ST} & \langle (\boxed{3}\ \text{DetP}) \rangle
\end{bmatrix} \\
\text{CONT} & \begin{bmatrix}
\text{INDEX} & \boxed{1} \\
\text{KEY} & \boxed{2} \\
\text{RELS} & \left\langle \boxed{2} \begin{bmatrix} \textit{bagel\_rel} & \\ \text{INST} & \boxed{1} \end{bmatrix} \right\rangle
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Example 2

- *head-complement* schema

$$
\begin{bmatrix}
\textit{head-comps-ph} & \\
\text{PHON} & \boxed{C} \oplus \boxed{D} \oplus \ldots \oplus \boxed{N} \\
\text{SYNSEM} & \begin{bmatrix} \text{LOCAL} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{SUBJ} & \boxed{A} \\ \text{SPR} & \boxed{B} \\ \text{COMPS} & \langle\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} & \boxed{2} \\ \text{KEY} & \boxed{3} \\ \text{RELS} & \boxed{F} \oplus \boxed{M} \oplus \ldots \oplus \boxed{Z} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\
\text{HEAD-DTR} & \begin{bmatrix} \text{PHON} & \boxed{C} \\ \text{SYNSEM} & \begin{bmatrix} \text{LOCAL} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{SUBJ} & \boxed{A} \\ \text{SPR} & \boxed{B} \\ \text{COMPS} & \text{sts}\langle \boxed{E} \rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} & \boxed{2} \\ \text{KEY} & \boxed{3} \\ \text{RELS} & \boxed{F} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\
\text{NON-HEAD-DTRS} & \boxed{E} \left\langle \begin{bmatrix} \text{PHON} & \boxed{D} \\ \text{SYNSEM} & \begin{bmatrix} \ldots & \text{RELS} \; \boxed{M} \end{bmatrix} \end{bmatrix}, \ldots \begin{bmatrix} \text{PHON} & \boxed{N} \\ \text{SYNSEM} & \begin{bmatrix} \ldots & \text{RELS} \; \boxed{Z} \end{bmatrix} \end{bmatrix} \right\rangle
\end{bmatrix}
$$

Example 2

- *head-complement* schema headed by *likes*

$$\begin{bmatrix}
\textit{head-comps-ph} & \\
\text{PHON} & \boxed{C} \oplus \boxed{D} \\
\text{SYNSEM} & \begin{bmatrix}\text{LOCAL} & \begin{bmatrix}\text{CAT} & \begin{bmatrix}\text{HEAD} & \boxed{1}\\ \text{SUBJ} & \boxed{A}\\ \text{SPR} & \boxed{B} \\ \text{COMPS} & \langle\rangle\end{bmatrix}\\ \text{CONT} & \begin{bmatrix}\text{INDEX} & \boxed{2}\\ \text{KEY} & \boxed{3}\\ \text{RELS} & \boxed{E}\oplus\boxed{F}\end{bmatrix}\end{bmatrix}\end{bmatrix}\\
\text{HEAD-DTR} & \begin{bmatrix}\text{PHON} & \boxed{A}\langle\text{likes}\rangle \\ \text{SYNSEM} & \begin{bmatrix}\text{LOCAL} & \begin{bmatrix}\text{CAT} & \begin{bmatrix}\text{HEAD} & \boxed{1}\begin{bmatrix}\textit{verb} & \\ \text{FORM} & \textit{fin}\end{bmatrix}\\ \text{SUBJ} & \boxed{A}\langle \text{NP}\begin{bmatrix}\textit{3sg}\end{bmatrix}_{\boxed{4}}\rangle \\ \text{SPR} & \boxed{B}\langle\rangle\\ \text{COMPS} & \langle \boxed{6}\text{NP}_{\boxed{5}}\rangle\end{bmatrix}\\ \text{CONT} & \begin{bmatrix}\text{INDEX} & \boxed{2}\\ \text{KEY} & \boxed{3}\\ \text{RELS} & \boxed{E}\left\langle \boxed{3}\begin{bmatrix}\textit{like\_rel} & \\ \text{EVENT} & \boxed{2}\\ \text{ARG1} & \boxed{4}\\ \text{ARG2} & \boxed{5}\end{bmatrix}, \begin{bmatrix}\textit{t-overlap\_rel} & \\ \text{ARG1} & \boxed{2}\\ \text{ARG2} & \textit{now}\end{bmatrix}\right\rangle\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}\\
\text{NON-HEAD-DTRS} & \left\langle\begin{bmatrix}\text{PHON} & \boxed{D}\\ \text{SYNSEM} & \boxed{6}\begin{bmatrix}\text{LOCAL | CONT | RELS} & \boxed{F}\end{bmatrix}\end{bmatrix}\right\rangle
\end{bmatrix}$$

Example 2

Kim ***likes bagels***

$$\begin{bmatrix} \textit{head-comps-ph} & \\ \text{PHON} & \langle \text{likes, bagels}\rangle \\ \text{SYNSEM} & \begin{bmatrix} \text{LOCAL} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix}\textit{verb} & \\ \text{FORM} & \textit{fin}\end{bmatrix} \\ \text{SUBJ} & \langle \text{NP}\begin{bmatrix}\textit{3sg}\end{bmatrix}\boxed{4}\rangle \\ \text{SPR} & \langle\rangle \\ \text{COMPS} & \langle\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} & \boxed{2} \\ \text{KEY} & \boxed{3} \\ \text{RELS} & \left\langle \boxed{3}\begin{bmatrix}\textit{like\_rel} & \\ \text{EVENT} & \boxed{2}\\ \text{ARG1} & \boxed{4} \\ \text{ARG2} & \boxed{5}\end{bmatrix}, \begin{bmatrix}\textit{t-overlap\_rel} & \\ \text{ARG1} & \boxed{2} \\ \text{ARG2} & \textit{now}\end{bmatrix}, \begin{bmatrix}\textit{bagel\_rel} & \\ \text{INST} & \boxed{5}\end{bmatrix}\right\rangle \end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Example 2

- *head-subject* schema

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{head-subj-ph}} \\
\text{PHON} & \boxed{B} \oplus \boxed{A} \\
\text{SYNSEM} & \left[\begin{array}{ll}
\text{LOCAL} & \left[\begin{array}{ll}
\text{CAT} & \left[\begin{array}{ll}
\text{HEAD} & \boxed{1} \\
\text{SUBJ} & \langle\rangle \\
\text{SPR} & \boxed{C} \\
\text{COMPS} & \boxed{D}
\end{array}\right] \\
\text{CONT} & \left[\begin{array}{ll}
\text{INDEX} & \boxed{2} \\
\text{KEY} & \boxed{3} \\
\text{RELS} & \boxed{E} \oplus \boxed{F}
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\text{HEAD-DTR} & \left[\begin{array}{ll}
\text{PHON} & \boxed{A} \\
\text{SYNSEM} & \left[\begin{array}{ll}
\text{LOCAL} & \left[\begin{array}{ll}
\text{CAT} & \left[\begin{array}{ll}
\text{HEAD} & \boxed{1} \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{verb}} \\
\text{FORM} & \textit{fin}
\end{array}\right] \\
\text{SUBJ} & \langle \boxed{4} \rangle \\
\text{SPR} & \boxed{C} \\
\text{COMPS} & \boxed{D} \langle\rangle
\end{array}\right] \\
\text{CONT} & \left[\begin{array}{ll}
\text{INDEX} & \boxed{2} \\
\text{KEY} & \boxed{3} \\
\text{RELS} & \boxed{E}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\text{NON-HEAD-DTRS} & \left\langle \left[\begin{array}{ll}
\text{PHON} & \boxed{B} \\
\text{SYNSEM} & \boxed{4} \left[\text{LOCAL} \mid \text{CONT} \mid \text{RELS}\ \boxed{F}\right]
\end{array}\right] \right\rangle
\end{array}\right]
$$

Example 2

- *head-subject* schema headed by *likes bagels*

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{head-subj-ph}} \\
\text{PHON} & \boxed{B} \oplus \boxed{A} \\
\text{SYNSEM} & \left[\begin{array}{ll}
\text{LOCAL} & \left[\begin{array}{ll}
\text{CAT} & \left[\begin{array}{ll}
\text{HEAD} & \boxed{1} \\
\text{SUBJ} & \langle\rangle \\
\text{SPR} & \boxed{C} \\
\text{COMPS} & \boxed{D}
\end{array}\right] \\
\text{CONT} & \left[\begin{array}{ll}
\text{INDEX} & \boxed{2} \\
\text{KEY} & \boxed{3} \\
\text{RELS} & \boxed{E} \oplus \boxed{F}
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\text{HEAD-DTR} & \left[\begin{array}{ll}
\text{PHON} & \boxed{A}\ \langle \text{likes, bagels} \rangle \\
\text{SYNSEM} & \left[\begin{array}{ll}
\text{LOCAL} & \left[\begin{array}{ll}
\text{CAT} & \left[\begin{array}{ll}
\text{HEAD} & \boxed{1} \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{verb}} \\ \text{FORM} & \textit{fin} \end{array}\right] \\
\text{SUBJ} & \langle \boxed{4} \text{NP}\left[\textit{3sg}\right]_{\boxed{5}} \rangle \\
\text{SPR} & \boxed{C} \\
\text{COMPS} & \boxed{D}\ \langle\rangle
\end{array}\right] \\
\text{CONT} & \left[\begin{array}{ll}
\text{INDEX} & \boxed{2} \\
\text{KEY} & \boxed{3} \\
\text{RELS} & \boxed{E} \left\langle \boxed{3} \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{like\_rel}} \\ \text{EVENT} & \boxed{2} \\ \text{ARG1} & \boxed{5} \\ \text{ARG2} & \boxed{6} \end{array}\right], \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{t-overlap\_rel}} \\ \text{ARG1} & \boxed{2} \\ \text{ARG2} & \textit{now} \end{array}\right], \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{bagel\_rel}} \\ \text{INST} & \boxed{6} \end{array}\right] \right\rangle
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\text{NON-HEAD-DTRS} & \left\langle \left[\begin{array}{ll}
\text{PHON} & \boxed{B} \\
\text{SYNSEM} & \boxed{4} \left[\text{LOCAL} \mid \text{CONT} \mid \text{RELS}\ \boxed{F} \right]
\end{array}\right] \right\rangle
\end{array}\right]
$$

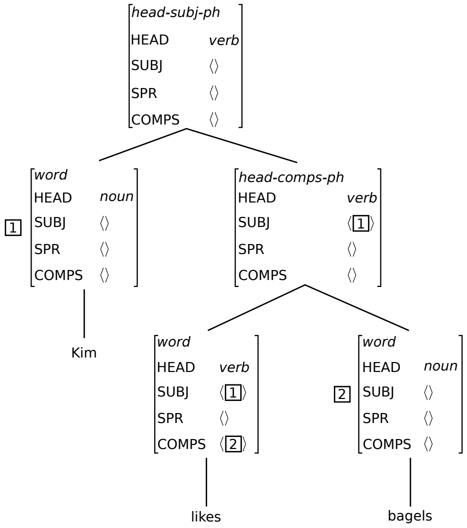
Example 2

Kim likes bagels

$$\left[\begin{array}{ll}
\textit{head-subj-ph} & \\
\text{PHON} & \langle \text{Kim, likes, bagels} \rangle \\
\text{SYNSEM} & \left[\begin{array}{ll}
\text{LOCAL} & \left[\begin{array}{ll}
\text{CAT} & \left[\begin{array}{ll}
\text{HEAD} & \left[\begin{array}{ll} \textit{verb} & \\ \text{FORM} & \textit{fin} \end{array}\right] \\
\text{SUBJ} & \langle\rangle \\
\text{SPR} & \langle\rangle \\
\text{COMPS} & \langle\rangle
\end{array}\right] \\
\text{CONT} & \left[\begin{array}{ll}
\text{INDEX} & \boxed{2} \\
\text{KEY} & \boxed{3} \\
\text{RELS} & \left\langle
\left[\begin{array}{ll} \textit{named\_rel} & \\ \text{INST} & \boxed{5} \\ \text{ARG} & \text{Kim} \end{array}\right],
\left[\begin{array}{ll} \textit{like\_rel} & \\ \text{EVENT} & \boxed{2} \\ \text{ARG1} & \boxed{5} \\ \text{ARG2} & \boxed{6} \end{array}\right],
\left[\begin{array}{ll} \textit{t-overlap\_rel} & \\ \text{ARG1} & \boxed{2} \\ \text{ARG2} & \textit{now} \end{array}\right],
\left[\begin{array}{ll} \textit{bagel\_rel} & \\ \text{INST} & \boxed{6} \end{array}\right]
\right\rangle
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

Example 2

Tree of ***Kim likes bagels***



Compare HPSG to CFG

- Each sign or HPSG rule consists of SYNSEM, DTRS, and PHON parts.
- The SYNSEM part specifies how the syntax and semantics of the phrase (or word) are constrained. It corresponds roughly to the left-hand side of CFG rules but contains much more information.
- The DTRS part specifies the constituents that make up the phrase (if it is a phrase). (Each of these constituents is a complete sign.) This corresponds to part of the information on the right-hand side of CFG rules, but not to ordering information.
- The PHON part specifies the ordering of the constituents in DTRS (where this is constrained) and the pronunciation of these (if this is specifiable). This corresponds to the the ordering information on the right-hand side of CFG rules.

Simulation of Bottom-up parsing algorithm in HPSG

- Unify input lexical-signs with lexical-signs in the lexicon.
- Until no more such unifications are possible
  - Unify instantiated signs with the daughters of instantiated phrasal signs or with phrasal signs in the grammar.

```
if
    all instantiated signs but one saturated one (S) are associated with daughters of
    other instantiated signs and the PHON value of all instantiated signs is
    completely specified

    return the complete S structure
else fail.
```

Example 2: processing of unification

Kim walks

The words in the sentence specify only their pronunciations and their positions.

1 [PHON (($_{0\ 1}$ kim))]
2 [PHON (($_{1\ 2}$ walks))]

STEP 1: Unifying 1 with the lexical entry for Kim gives

```
3 [PHON ((0 1 kim))
   SYNSEM [CAT [HEAD noun SUBCAT ()]
              CONTENT [INDEX 1 [PER 3rd NUM sing]]
              CONTEXT [BACKGR {[RELN naming BEARER 1 NAME Kim]}]]]
```

We now know something about the meaning of Kim (it refers to somebody named Kim) and something about its syntactic properties (it is third person singular).

Example 2: processing of unification

1 [PHON ((0 1 kim))]
2 [PHON ((1 2 walks))]

STEP 2: Unifying 2 with the lexical entry for walks gives

```
4 [PHON ((1 2 walks))
   SYNSEM [CAT [HEAD [VFORM fin]
                SUBCAT ([CAT [HEAD noun SUBCAT ()]
                           CONTENT [INDEX 1 [PER 3rd NUM sing]]])]
           CONTENT [RELN walk WALKER 1]]]
```

We know that walks refers to walking and that it requires a subject noun phrase which refers to the walker but doesn't require any object.

Example 2: processing of unification

HEAD-DTR rule

```
[SYNSEM [CAT [HEAD 1 SUBCAT (2)]
             CONTENT 4]
 DTRS [HEAD-DTR [SYNSEM [CAT [HEAD 1 SUBCAT (2)]
                              CONTENT 4]
                        PHON 3]
         SUBJ-DTRS ()]
 PHON 3]
```

STEP 3: Unifying 4 with the HEAD-DTR of this rule gives

```
5 [SYNSEM [CAT [HEAD [VFORM fin]
                    SUBCAT 2([CAT [HEAD noun SUBCAT ()]
                                   CONTENT [INDEX 1 [PER 3rd NUM sing]]]])]
               CONTENT 4[RELN walk WALKER 1]]
   DTRS [HEAD-DTR [SYNSEM [CAT [HEAD [VFORM fin] SUBCAT (2)]]
                          CONTENT [4]
                          PHON 3((1 2 walks))]
         SUBJ-DTRS ()]
   PHON 3((1 2 walks))]
```

Now we have a VP with the transitive verb walks as its head (and only constituent).

Example 2: processing of unification

HEAD-DTR rule

```
6 [SYNSEM [CAT [HEAD 1 SUBCAT ()]
              CONTENT 4]
   DTRS [HEAD-DTR [SYNSEM [CAT [HEAD 1 SUBCAT (2)]
                                    CONTENT 4]
                         PHON 3]
           SUBJ-DTRS ([PHON 5
                          SYNSEM 2])]
   PHON (5 < 3)]
```

STEP 4: Unifying 5 with the HEAD-DTR of this rule gives

```
7 [SYNSEM [CAT [HEAD 1[VFORM fin SUBCAT ()]]
              CONTENT 4[RELN walk WALKER ]]
   DTRS [HEAD-DTR [SYNSEM [CAT [HEAD 1[VFORM fin]
                                    SUBCAT 2([CAT [HEAD noun SUBCAT ()]
                                                CONTENT [INDEX
                                                          [PER 3rd NUM sing]]]])]
                                CONTENT [RELN walk WALKER 4]]
                         PHON 3((1 2 walks))]
           SUBJ-DTRS ([PHON 5
                         SYNSEM 2[CAT [HEAD noun SUBCAT ()]
                                    CONTENT [INDEX ]]]])]
   PHON ( 5 < 3((1 2 walks)))]
```

Example 2: processing of unification

STEP 5: Unifying 3 with the SUBJ-DTR of 7 gives

```
8 [SYNSEM [CAT [HEAD [VFORM fin SUBCAT ()]]
              CONTENT [RELN walk WALKER [PER 3rd NUM sing]]]
   DTRS [HEAD-DTR [SYNSEM [CAT [HEAD [VFORM fin]
                                             SUBCAT ([CAT [HEAD noun SUBCAT ()]
                                                        CONTENT [INDEX [PER 3rd NUM sing]]]])
                                        CONTENT [RELN walk WALKER [PER 3rd NUM sing]]]
                             PHON ((1 2 walks))]
          SUBJ-DTRS ([PHON ((0 1 kim))
                              SYNSEM [CAT [HEAD noun SUBCAT ()]
                                          CONTENT [INDEX [PER 3rd NUM sing]]]])]
   PHON ((0 1 kim) (1 2 walks))]
```

Now the subject of the sentence is pronounceable, and we're done.

Phenomena covered by HPSG parsers

- Case assignment
- Word order : scrambling
- Long distance dependency
- Coordination
- Scope of adverbs and negation
- Topic drop
- Agreement
- Relative clause
- ...

Example 3: unbounded dependency construction

- An unbounded dependency construction
  - involves constituents with different functions
  - involves constituents of different categories
  - is in principle unbounded
- Two kind of unbounded dependency constructions (UDCs)
  - Strong UDCs
  - Weak UDCs

Strong UDCs

- An overt constituent occurs in a non-argument position:
  - Topicalization:
    Kim$_i$, Sandy loves_ $_i$.
  - Wh-questions:
    I wonder [who$_i$ Sandy loves_ $_i$].
  - Wh-relative clauses:
    This is the politician [who$_i$ Sandy loves_ $_i$].
  - It -clefts:
    It is Kim i [who$_i$ Sandy loves_ $_i$].
  - Pseudoclefts:
    [What$_i$ Sandy loves_ $_i$] is Kim$_i$.

Weak UDCs

- No overt constituent in a non-argument position:
  - Purpose infinitive (for -to clauses):
    I bought it$_i$ for Sandy to eat_ $_i$.
  - Tough movement:
    Sandy$_i$ is hard to love_ $_i$.
  - Relative clause without overt relative pronoun:
    This is [the politician]$_i$ [Sandy loves_ $_i$].
  - It-clefts without overt relative pronoun:
    It is Kim$_i$ [Sandy loves_ $_i$].

Using the feature SLASH

- To account for UDCs, we will use the feature SLASH (so-named because it comes from notation like S/NP to mean an S missing an NP)
- This is a non-local feature which originates with a trace, gets passed up the tree, and is finally bound by a filler

The bottom of a UDC: Traces

$$\begin{bmatrix} \textit{word} & \\ \text{PHON} & \langle\rangle \\ \text{SYNSEM} & \begin{bmatrix} \text{LOCAL} & \boxed{1} \\ \text{NONLOC} & \begin{bmatrix} \text{INHERITED} \mid \text{SLASH} & \{\boxed{1}\} \\ \text{TO-BIND} \mid \text{SLASH} & \{\} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- phonologically null, but structure-shares local and slash values

Traces

- Because the *local* value of a trace is structure-shared with the *slash* value, constraints on the trace will be constraints on the filler.
  - For example, hates specifies that its object be accusative, and this case information is local
  - So, the trace has [synsem|local|cat|head|case acc] as part of its entry, and thus the filler will also have to be accusative

    **He$_i$/Him$_i$, John likes_ $_i$*

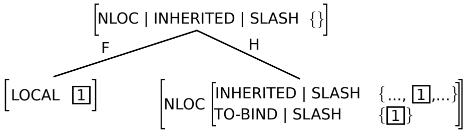
The middle of a UDC: The Nonlocal Feature Principle (NFP)

- For each NON-LOCAL feature, the *inherited* value on the mother is the union of the *inherited* values on the daughter minus the *to-bind* value on the head daughter.
- In other words, the slash information (which is part of inherited) percolates “up” the tree
- This allows the all the local information of a trace to “move up” to the filler

The middle of a UDC: The Nonlocal Feature Principle (NFP)

- The top of a UDC: *filler-head* structures

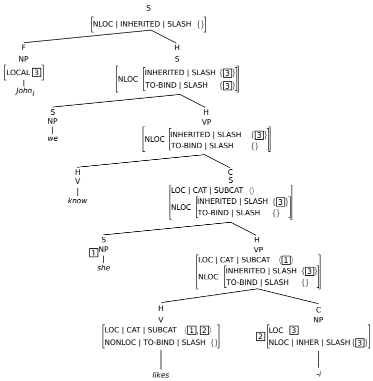
Example for a structure licensed by the *filler-head* schema



The middle of a UDC: The Nonlocal Feature Principle (NFP)

- The analysis of the UDC example

John$_i$ we know She likes_$_i$



Example 4

John ***reads*** *a new book*

$$
{}_{\textit{word}}\begin{bmatrix}
\text{PHON} & \langle \text{reads} \rangle \\
\text{SYNSEM} \mid \text{LOC} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix}
\text{HEAD} & {}_{\textit{verb}}\begin{bmatrix} \text{VFORM} & \text{fin} \\ \text{AUX} & \text{bool} \\ \text{INV} & \text{bool} \end{bmatrix} \\
\text{VAL} & \begin{bmatrix}
\text{SUBJ} & \langle \text{NP}\,\boxed{1}_{[\textit{3rd,sg}]}\ {}^{[\text{nom,-PRD}]} \rangle \\
\text{COMPS} & \langle \text{NP}\,\boxed{2}\ {}^{[\text{acc,-PRD}]} \rangle \\
\text{SPR} & \langle\rangle
\end{bmatrix}
\end{bmatrix} \\
\text{CONT} & {}_{\textit{read}}\begin{bmatrix} \text{READER} & \boxed{1} \\ \text{READEE} & \boxed{2} \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Example 4

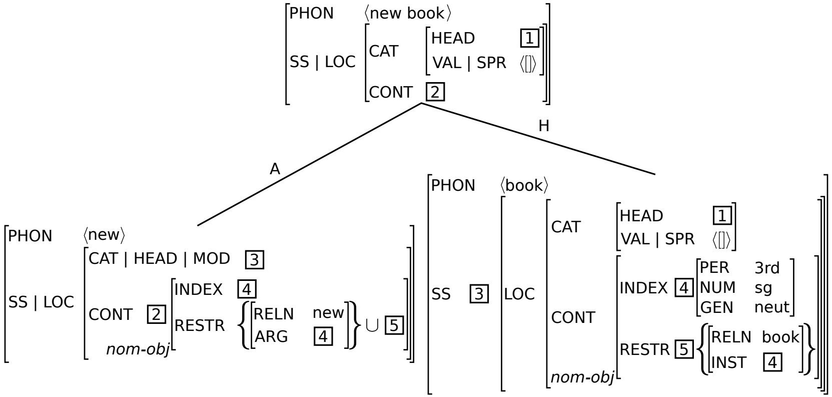
John reads a ***new*** *book*

$$
\begin{bmatrix}
\text{PHON} & \langle \text{new} \rangle \\
\text{SYNSEM} \mid \text{LOCAL} & \begin{bmatrix}
\text{CAT} \mid \text{HEAD} & \begin{bmatrix}
\text{PRD} & - \\
\text{MOD} & \begin{bmatrix}
\text{LOCAL} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix}
\text{HEAD} & \text{noun} \\
\text{VAL} \mid \text{SPR} & \langle [\,] \rangle
\end{bmatrix}_{\textit{cat}} \\
\text{CONT} & \begin{bmatrix}
\text{INDEX} & \boxed{1} \\
\text{RESTR} & \boxed{2}
\end{bmatrix}_{\textit{nom-obj}}
\end{bmatrix}_{\textit{local}}
\end{bmatrix}_{\textit{synsem}}
\end{bmatrix}_{\textit{adj}} \\
\text{CONT} & \begin{bmatrix}
\text{INDEX} & \boxed{1} \\
\text{RESTR} & \left\{ \begin{bmatrix}
\text{RELN} & \text{new} \\
\text{ARG} & \boxed{1}
\end{bmatrix} \right\} \cup \boxed{2}
\end{bmatrix}_{\textit{nom-obj}}
\end{bmatrix}_{\textit{local}}
\end{bmatrix}_{\textit{word}}
$$

Example 4

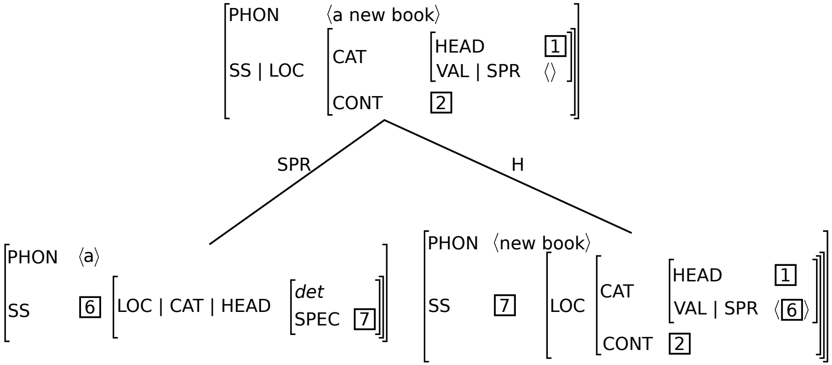
John reads a ***new book***

- Note: apply head-adjunct schema



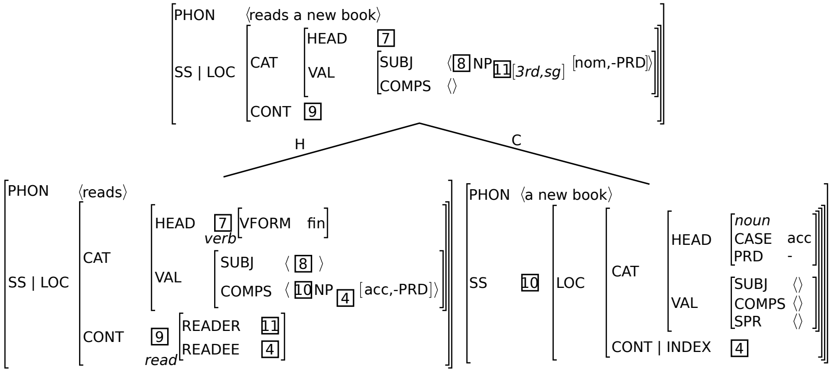
Example 4

John reads a new book



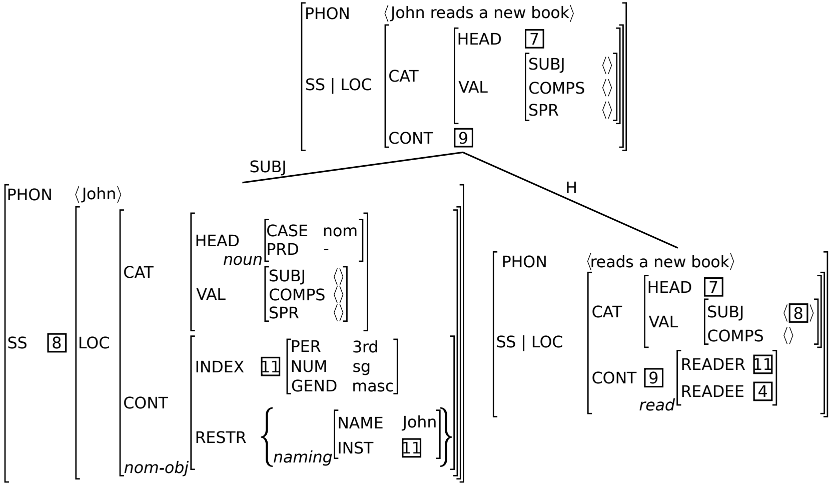
Example 4

John ***reads a new book***



Example 4

John reads a new book - completed analysis



Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

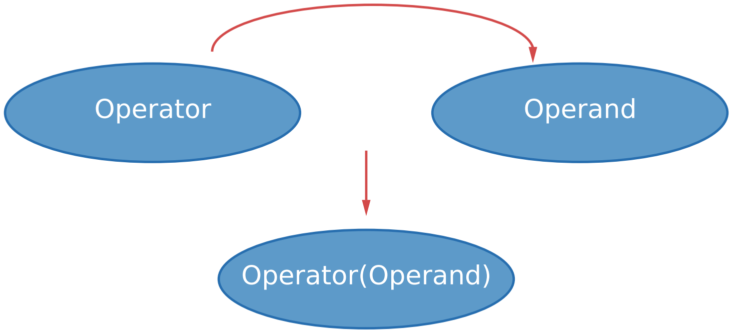
Outline

- Applicative system
- Combinators
- Combinators vs. λ-expressions
- Application to natural language parsing
- Combinators used in CCG

Applicative system

- CL (Curry & Feys, 1958, 1972) as an applicative system

  CL is an applicative system because the basic unique operation in CL is the application of an **operator** to an **operand**



Combinators

CL defines general operators, called Combinators.

- Each combinator composes between them the elementary combinators and defines the complexe combinators.
- Certains combinators are considered as the basic combinators to define the other combinators.

Elementary combinators

I	$=_{\text{def}}$	$\lambda x.x$	*(identificator)*
K	$=_{\text{def}}$	$\lambda x.\lambda y.x$	*(cancellator)*
W	$=_{\text{def}}$	$\lambda x.\lambda y.xyy$	*(duplicator)*
C	$=_{\text{def}}$	$\lambda x.\lambda y.\lambda z.xzy$	*(permutator)*
B	$=_{\text{def}}$	$\lambda x.\lambda y.\lambda z.x(yz)$	*(compositor)*
S	$=_{\text{def}}$	$\lambda x.\lambda y.\lambda z.xz(yz)$	*(substitution)*
Φ	$=_{\text{def}}$	$\lambda x.\lambda y.\lambda z.\lambda u.x(yu)(zu)$	*(distribution)*
Ψ	$=_{\text{def}}$	$\lambda x.\lambda y.\lambda z.\lambda u.x(yz)(yu)$	*(distribution)*

β-reductions

The combinators are associated with the **β-reductions** in a canonical form:

β-reduction relation between X and Y

$$X \geq_\beta Y$$

Y was obtained from X by a β-reduction

β-reductions

$$\begin{array}{lll}
Ix & \geq_\beta & x \\
Kxy & \geq_\beta & x \\
Wxy & \geq_\beta & xyy \\
Cxyz & \geq_\beta & xzy \\
Bxyz & \geq_\beta & x(yz) \\
Sxyz & \geq_\beta & xz(yz) \\
\Phi xyzu & \geq_\beta & x(yu)(zu) \\
\Psi xyzu & \geq_\beta & x(yz)(yu)
\end{array}$$

Each combinator is an operator which has a certain number of arguments (operands); sequences of the arguments which follow the comnator are called “the scope of combinator”.

β-reductions

Intuitive interpretations of the elementary combinators are given by the associated **β-reductions**.

- The combinator *I* expresses the identity.
- The combinator *K* expresses the constant function.
- The combinator *W* expresses the diagonalisation or the duplication of an argument.
- The combinator *C* expresses the conversion, that is, the permutation of two arguments of an binary operator.
- The combinator *B* expresses the functional composition of two operators.
- The combinator *S* expresses the functional composition and the duplication of argument.
- The combinator Φ expresses the composition in parallel of operators acting on the common data.
- The combinator Ψ expresses the composition by distribution.

Introduction and elimination rules of combinators

Introduction and elimination rules of combinators can be presented in the style of Gentzen (*natural deduction*).

Elim. Rules	**Intro. Rules**
If	f
--- [e-**I**]	--- [i-**I**]
f	If
Kfx	f
----- [e-**K**]	---- [i-**K**]
f	Kfx

Introduction and elimination rules of combinators

Elim. Rules	**Intro. Rules**
Cfx	xf
- - - [e-**C**]	- - - [i-**C**]
xf	**C**fx
Bfxy	f(xy)
- - - - - [e-**B**]	- - - - [i-**B**]
f(xy)	**B**fxy
Φfxyz	f(xz)(yz)
- - - - - [e-Φ]	- - - - [i-Φ]
f(xz)(yz)	Φfxyz

Combinators vs. λ -expressions

The most important difference between the CL and λ-calculus is the use of the bounded variables.

Every combinator is an λ -expression.

$$\begin{aligned}\mathbf{B}fg &\equiv \lambda x.f(gx)\\ \mathbf{T}x &\equiv \lambda f.fx\\ \mathbf{S}fg &\equiv \lambda x.fx(gx)\end{aligned}$$

Application to natural language parsing

John is brilliant

- The predicate *is brilliant* is an operator which operate on the operand John to construct the final proposition.
- The applicative representation associated to this analysis is the following:

(is-brillant)John

- We define the operator **John*** as being constructed from the lexicon *John* by

[John* = **C*** John].

1. John* (is-brillant)
2. [John* = **C*** John]
3. **C***John (is-brillant)
4. is-brillant (John)

Application to natural language parsing

John is brilliant in λ-term

Operator John* by λ-expression

[John* = λx.x (John')]

1. John*(λx.is-brilliant'(x))
2. [John* = λx.x (John')]
3. (λx.x(John'))(λx.is-brilliant'(x))
4. (λx.is-brilliant'(x))(John')
5. is-brillinat'(John')

Passivisation

Consider the following sentences

a. The man has been killed.

b. One has killed him.

$\rightarrow$ Invariant of meaning
$\rightarrow$ Relation between two sentences

:a. unary passive predicate (*has-been-killed*)

:b. active transitive predicate (*have-killed*)

Definition of the operator of passivisation ’PASS’

$$[\mathrm{PASS} = \mathrm{B} \sum \mathrm{C} = \sum \circ \mathrm{C}]$$

where B and C are the combinator of composition and of conversion and where $\sum$ is the existential quantificator which, by applying to a binary predicate, transforms it into the unary predicate.

Definition of the operator of passivisation ’PASS’

[PASS = B $\sum$ C = $\sum \circ$ C]

1/	has-been-killed (the-man)	*hypothesis*
2/	[has-been-killed=PASS(has killed)]	*passive lexical predicate*
3/	PASS (has-killed)(the-man)	*repl.2.,1.*
4/	[PASS = ***B*** $\sum$ ***C***]	*definition of ’PASS’*
5/	***B*** $\sum$ ***C*** (has-killed)(the-man)	*repl.4.,3.*
6/	$\sum$ (***C***(has-killed))(the-man)	[e-***B***]
7/	(***C***(has-killed)) x (the-man)	[e-$\sum$]
8/	(has-killed)(the-main) x	[e-***C***]
9/	[x in the agentive subject position = *one*]	*definition of ’one’*
10/	(has-killed)(the-man)*one*	*repl.9.,8., normal form*

Definition of the operator of passivisation ’PASS’

We establish the paraphrastic relation between the passive sentence with expressed agent and its active counterpart:

The man has been killed by the enemy

$$\downarrow$$

The enemy has killed the man

Definition of the operator of passivisation ’PASS’

Relation between give-to and receive-from

z *gives* y *to* x

$\updownarrow$

x *receives* y *from* x

The lexical predicate “*give-to*” has a predicate converse associated to “*receive-from*”;

[receive-from z y x = give-to x y z]

Definition of the operator of passivisation 'PASS'

1/ **(receive-from) z y x**

2/ **C**((receive-from) z) x y

3/ **BC**(receive-from) z x y

4/ **C(BC**(receive-from)) z x y

5/ **C(C(BC**(receive-from)) x) y z

6/ **BC(C(BC**(receive-from))) x y z

7/ [give-to=**BC(C(BC**(receive-from)))]

8/ **give-to x y z**

Combinators used in CCG

Motivation of applying the combinators

to natural language parsing

- Linguistic: complex phenomena of natural language applicable to the various languages
- Informatics: left to right parsing (LR)
  ex: reduce the spurious-ambiguity

Parsing a sentence in CCG

Step 1: tokenization

Step 2: tagging the concatenated lexicon

Step 3: calculate on types attributed to the concatenated lexicons by applying the adequate combinatorial rules

Step 4: eliminate the applied combinators (we will see how to do on next week)

Step 5: finding the parsing results presented in the form of an operator/operand structure (predicate -argument structure)

Parsing a sentence in CCG

Example: *I requested and would prefer musicals*
STEP 1 : tokenization/lemmatization $\rightarrow$ *ex) POS Tagger, tokenizer, lemmatizer*

a. *I-requested-and-would-prefer-musicals*
b. *I-request-ed-and-would-prefer-musical-s*

STEP 2 : tagging the concatenated expressions $\rightarrow$ *ex) Supertagger, Inventory of typed words*

I	NP
Requested	$(S\backslash NP)/NP$
And	$CONJ$
Would	$(S\backslash NP)/VP$
Prefer	VP/NP
musicals	NP

Parsing a sentence in CCG

STEP 3 : categorial calculus

a. apply the type-raising rules $\longrightarrow$ *Subject Type-raising* $(>T)$
$NP : a \Rightarrow T/(T\backslash NP) : Ta$

b. apply the functional composition rules $\longrightarrow$ *Forward Composition:* $(>B)$
$X/Y : f \quad Y/Z : g \Rightarrow X/Z : Bfg$

c. apply the coordination rules $\longrightarrow$ *Coordination:* $(<\&>)$
$X\ conj\ X \Rightarrow X$

	I-	requested-	and-	would-	prefer-	musicals	
1/	NP	$(S\backslash NP)/NP$	CONJ	$(S\backslash NP)/VP$	VP/NP	NP	
2/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$	CONJ	$(S\backslash NP)/VP$	VP/NP	NP	$(>T)$
3/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$	CONJ	$(S\backslash NP)/NP$		NP	$(>B)$
4/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$				NP	$(>\Phi)$
5/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$				NP	$(>B)$
6/	S/NP					NP	$(>)$
7/				**S**			

Parsing a sentence in CCG

STEP 4 : semantic representation (predicate-argument structure)

I	requested	and	would	prefer	musicals

1/ :i' :request' :and' : will' :prefer' : musicals'

2/ :λf.f I'

3/ : λx.λy.will'(prefer'x)y

4/ : λxλy.and'(will'(prefer'x)y)(request'xy)

5/ : λxλy.and'(will'(prefer'x)y)(request'xy)

6/ :λy.and'(would'(prefer' musicals')y)(request' musicals' y)

7/S: and'(will'(prefer' musicals') i')(request' musicals' i')

Semantic representation in term of the *combinators*

```
    I-              requested     and-     would-     prefer      musicals
1/ NP               (S\NP)/NP    CONJ   (S\NP)/VP     VP/NP       NP
2/ S/(S\NP)         (S\NP)/NP    CONJ   (S\NP)/VP     VP/NP       NP       (>T)
    C*I               requested     and      would      prefer      musicals
3/ S/(S\NP)               (S\NP)/NP    CONJ      (S\NP)/NP          NP       (>B)
    C*I                   requested and      B would prefer          musicals
4/ S/(S\NP)                   (S\NP)/NP                             NP       (> Φ)
    C*I                   Φ and requested (B would prefer)      musicals
5/          S/NP                                         NP                  (>B)
    B((C*I)(Φ and requested (B would prefer)))    musicals
6/                   S                                                      (>)
        B((C*I)(Φ and requested (B would prefer)))    musicals
```

Semantic representation in term of the *combinators*

I requested and would prefer musicals

S:	**B((C*I)(Φ and requested (B would prefer))) musicals**	
1/	**B((C*I)(Φ and requested (B would prefer))) musicals**	
2/	**(C*I)((Φ and requested (B would prefer))) musicals)**	**[e-B]**
3/	**((Φ and requested (B would prefer))) musicals) I**	**[e-C*]**
4/	**(and (requested musicals) ((B would prefer) musicals)) I**	**[e-Φ]**
5/	**((and (requested musicals) (would (prefer musicals))) I)**	**[e-B]**

Normal form

A <u>normal form</u> is a combinatory expression which is irreducible in the sense that it contain any occurrence of a redex.

If a combinatory expression X reduce to a combinatory expression N which is in <u>normal form</u>, so N is called the <u>normal form</u> of X.

Example

Bxyz is reducible to x(yz).
x(yz) is a normal form of the combinatory expression **B**xyz.

Normal form

Example

Prove xyz is the normal form of **BBC**xyz.

$$\mathbf{BBC}xyz \to_\beta xyz$$

1/	**BBC**xyz	
2/	**C**(**C**x)yz	[e-**B**]
3/	**C**xzy	[e-**C**]
4/	xyz	[e-**C**]

Classwork

Give the semantic representation in term of combinators.
Please refer to the given paper on last lecture on CCG Parsing.

Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

Outline

- HPSG Parser : **Enju**
  - Parsing method
  - Description of parser
  - Result
- CCG Parser : **C&C Tools**
  - Parsing method
  - Description of parser
  - Result

Theoretical backgrounds

Lecture 3 about HPSG Parsing

Lecture 6 & 7 about CCG Parsing and Combinatory Logic

Enju (Y. Miyao, J.Tsujii, 2004, 2008)

- Syntactic parser for English
- Developed by Tsujii Lab. Of the University of Tokyo
- Based on the wide-coverage probabilistic HPSG
  - HPSG theory [Pollard and Sag, 1994]
- Useful links to Enju
  - `http://www-tsujii.is.s.u-tokyo.ac.jp/enju/demo.html`
  - `http://www-tsujii.is.s.u-tokyo.ac.jp/enju/`

Motivations

Parsing based on a proper linguistic formalism is one of the core research fields in CL and NLP.

But!

a monolithic, esoteric and inward looking field, largely dissociated from real world application.

Motivations

So why not!

The integration of linguistic grammar formalisms with statistical models to propose an robust, efficient and open to eclectic sources of information other than syntactic ones

Motivations

Two main ideas

- Development of wide-coverage linguistic grammars
- Deep parser which produces semantic representation (predicate-argument structures)

Parsing method

- Application of probabilistic model in the HPSG grammar and development of an efficient parsing algorithm
    - Accurate deep analysis
    - Disambiguation
    - Wide-coverage
    - High speed
    - Useful for high level NLP application

Parsing method

1. Parsing based on HPSG
   - Mathematically well-defined with sophisticated constraint-based system
   - Linguistically justified
   - Deep syntactic grammar that provides semantic analysis

Parsing method

Difficulties in parsing based on HPSG

- Difficult to develop a broad-coverage HPSG grammar
- Difficult to disambiguate
- Low efficiency: very slow

Parsing method

Solution:
Corpus-oriented development of an HPSG grammar

- The principal aim of grammar development is treebank construction
- Penn treebank is coverted into an HPSG treebank
- A lexicon and a probabilistic model are extracted from the HPSG treebank

Parsing method

Approach:

- develop grammar rules and an HPSG treebank
- collect lexical entries from the HPSG treebank

How to make an HPSG treebank?

Convert Penn Treebank into HPSG and develop grammar by restructuring a treebank in conformity with HPSG grammar rules

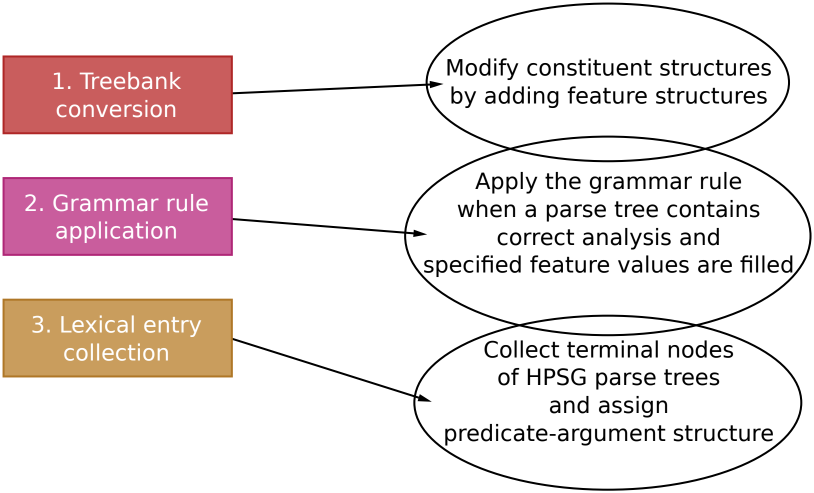
Parsing method

HPSG = lexical entries and grammar rules
Enju grammar has <u>12 grammar rules</u> and
<u>3797 lexical entries for 10,536 words</u>

(Miyao *et al.* 2004)

Parsing method

Overview of grammar development


1. Treebank conversion
Modify constituent structures by adding feature structures
2. Grammar rule application
Apply the grammar rule when a parse tree contains correct analysis and specified feature values are filled
3. Lexical entry collection
Collect terminal nodes of HPSG parse trees and assign predicate-argument structure

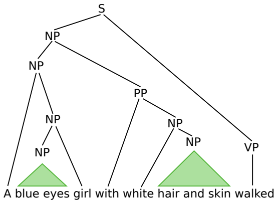
Parsing method

2 Probabilistic model and HPSG:

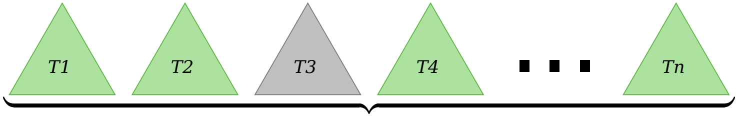
Log-linear model for unification-based grammars

(Abney 1997, Johnson et al. 1999, Riezler et al. 2000, Miyao et al. 2003, Malouf and van Noord 2004, Kaplan et al. 2004, Miyao and Tsujii 2005)

$$p(\mathrm{T}|w)$$
$$w = \text{“}A\ blue\ eyes\ girl\ with\ white\ hair\ and\ skin\ walked$$
$$T =$$



Parsing method



All possible parse trees derived from w with a grammar.
For example, $p(T3|\mathrm{w})$ is the probability of selecting $T3$ from $T1$, $T2$, ..., and Tn.

Parsing method

Log-linear model for unification-based grammars

- Input sentence: w

$w = w_1/P_1, w_2/P_2, \ldots w_n/P_n$

- Output parse tree T

$$\boxed{p(T|\mathbf{w}) = \underbrace{\frac{1}{Z}}_{\text{Normalization factor}} \exp\left(\sum_u \underbrace{\lambda_u}_{\text{Weight for a feature function}} \underbrace{f_u}_{\text{Feature function}}(T)\right)}$$

Description of parser

Description of parser

parsing proceeds in the following steps:

1. preprocessing

Preprocessor converts an input sentence into a word lattice.

2. lexicon lookup

Parser uses the predicate to find lexical entries for the word lattice

3. kernel parsing

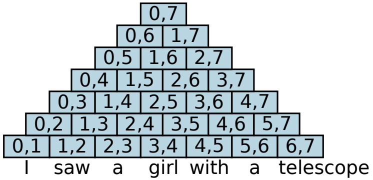
Parser does phrase analysis using the defined grammar rules in the kernel parsing process.

Description of parser

- Chart
  - data structure
  - two dimensional table
  - we call each cell in the table ‘CKY cell.’

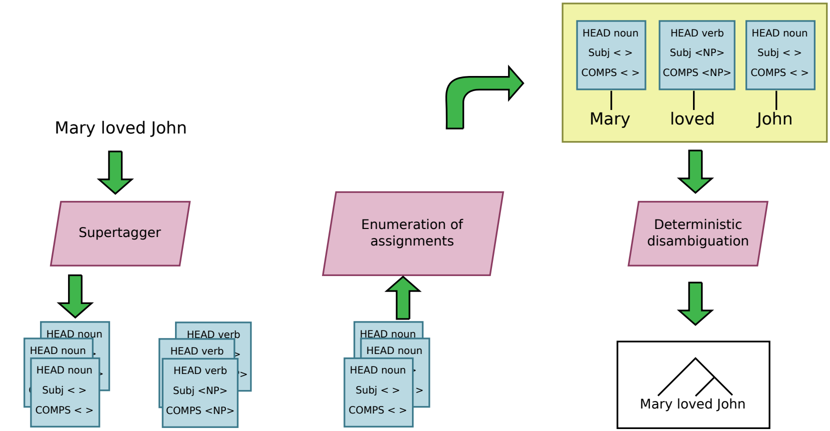
Example

Let an input sentence $s(= w1, w2, w3, ..., wn), w1 = $ ”I”$, w2 =$ ”saw”$, w3 =$ ”a”$, w4 =$ ”girl”$, w5 =$ ”with”$, w6 =$ ”a”$, w7 =$ ”telescope” for the sentence “*I saw a girl with a telescope*”, the chart is arranged as follows.



Description of parser

System overview



Demonstration

`http://www-tsujii.is.s.u-tokyo.ac.jp/enju/demo.html`

Results

- Fast, robust and accurate analysis
  - Phrase structures
  - Predicate argument structures
- **Accurate deep analysis** – the parser can output both phrase structures and predicate-argument structures. The accuracy of predicate-argument relations is around 90% for newswire articles and biomedical papers.
- **High speed** – parsing speed is less than 500 msec. per sentence by default (faster than most Penn Treebank parsers), and less than 50 msec when using the highspeed setting (”mogura”).

C&C tools

- Developed by Curran and Clark [Clark and Curran, 2002, Curran, Clark and Bos, 2007], University of Edinburgh
- Wide-coverage statistical parser based on the CCG: CCG Parser
- Computational semantic tools named **Boxer**
- Useful links
  - `http://svn.ask.it.usyd.edu.au/trac/candc`
  - `http://svn.ask.it.usyd.edu.au/trac/candc/wiki/Demo`

CCG Parser [Clark, 2007]

Statistical parsing and CCG

Advantages of CCG

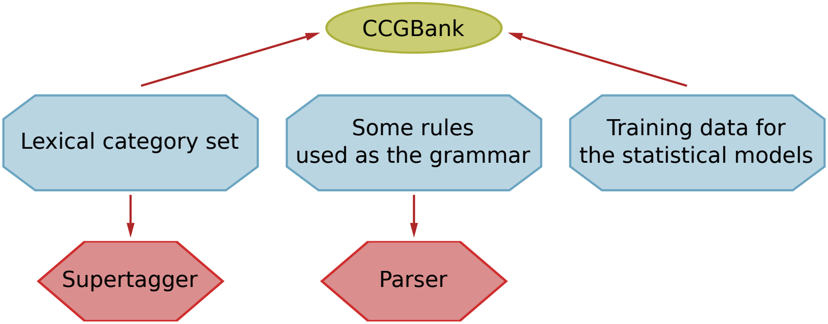
- providing a compositional semantic for the grammar

$\rightarrow$completely transparent interface between syntax and semantics

- the recovery of long-range dependencies can be integrated into the parsing process in a straightforward manner

Parsing method

- Penn Treebank conversion : TAG, LFG, HPSG and CCG
- **CCGBank** [Hockenmaier and Steedman, 2007]
  - CCG version of the Penn Treebank
  - Grammar used in CCG parser


CCGBank
Lexical category set
Some rules used as the grammar
Training data for the statistical models
Supertagger
Parser

Parsing method-CCG Bank

- Corpus translated from the Penn Treebank, CCGBank contains
  - Syntactic derivations
  - Word-word dependencies
  - Predicate-argument structures

Parsing method-CCG Bank

- Semi automatic conversion of phrase-structure trees in the Penn Treebank into CCG derivations
- Consists mainly of newspaper texts
- Grammar:

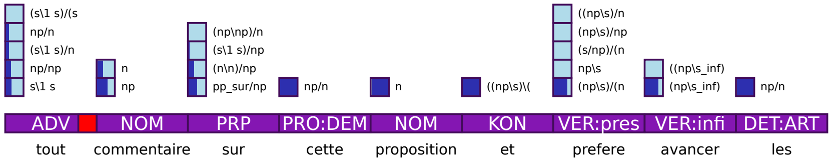
  - Lexical category set
  - Combinatory rules
  - Unary type-changing rules
  - Normal-form constraints
  - Punctuation rules

Parsing method

Supertagging [Clark, 2002]

uses conditional maximum entropy models

implement a maximum entropy supertagger



Parsing method-Supertagger

- Set of 425 lexical categories from the CCGbank
- The per-word accuracy of the Supertagger is around 92% on unseen WSJ text.

$\rightarrow$ Using the multi-supertagger increases the accuracy significantly – to over 98% – with only a small cost in increased ambiguity.

Parsing method-Supertagger

- **Log-linear models in NLP applications:**
  - POS tagging
  - Name entity recognition
  - Chunking
  - Parsing

$\rightarrow$ referred as **maximum entropy models** and **random fields**

Parsing method-Supertagger

Log-linear parsing models for CCG

1. the probability of a dependency structure
2. the normal-form model: the probability of a single derivation

$\rightarrow$ modeling 2) is simpler than 1)

1. defined as $P(\pi|S) = \sum_{d \in \Delta(\pi)} P(d, \pi|S)$
2. defined using a log-linear form as follows: $P(w|S) = \frac{1}{Z_S} e^{\lambda.f(w)}$

$$Z_S = \sum_{w \in p(S)} e^{\lambda.f(w')}$$

Parsing method-Supertagger

Features common to the dependency and normal-form models

Feature type	Example
LexCat + word	$(S/S)/NP +$ Before
LexCat + POS	$(S/S)/NP +$ IN
RootCat	$S[dcl]$
RootCat + World	$S[dcl] +$ was
RootCat + POS	$S[dcl] +$ VBD
Rule	$S[dcl] \rightarrow NP\ S[dcl]\backslash NP$
Rule + Word	$S[dcl] \rightarrow NP\ S[dcl]\backslash NP +$ bought
Rule + POS	$S[dcl] \rightarrow NP\ S[dcl]\backslash NP +$ VBD

Parsing method-Supertagger

Predicate-argument dependency features for the dependency model

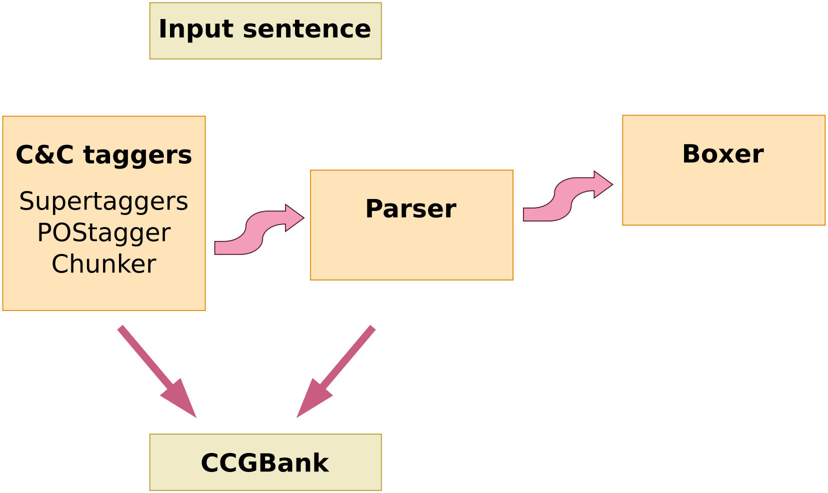
Feature type	Example
Word-Word	$\langle \mathit{bought}, (S\backslash NP_1)/NP_2, 2, \mathit{stake}, (NP\backslash NP)/(S[dcl]/NP)\rangle$
Word-POS	$\langle \mathit{bought}, (S\backslash NP_1)/NP_2, 2, NN, (NP\backslash NP)/(S[dcl]/NP)\rangle$
POS-Word	$\langle VBD, (S\backslash NP_1)/NP_2, 2, \mathit{stake}, (NP\backslash NP)/(S[dcl]/NP)\rangle$
POS-POS	$\langle VBD, (S\backslash NP_1)/NP_2, 2, NN, (NP\backslash NP)/(S[dcl]/NP)\rangle$
Word + Distance(words)	$\langle \mathit{bought}, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 2$
Word + Distance(punct)	$\langle \mathit{bought}, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 0$
Word + Distance(verbs)	$\langle \mathit{bought}, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 0$
POS + Distance(words)	$\langle VBD, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 2$
POS + Distance(punct)	$\langle VBD, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 0$
POS + Distance(verbs)	$\langle VBD, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 0$

Parsing method-Supertagger

Rule dependency features for the normal-form model

Feature type	Example
Word-Word	$\langle company, S[dcl] \rightarrow NP\ S[dcl]\backslash NP, bought\rangle$
Word-POS	$\langle company, S[dcl] \rightarrow NP\ S[dcl]\backslash NP, VBD\rangle$
POS-Word	$\langle NN, S[dcl] \rightarrow NP\ S[dcl]\backslash NP, bought\rangle$
POS-POS	$\langle NN, S[dcl] \rightarrow NP\ S[dcl]\backslash NP, VBD\rangle$
Word + Distance(words)	$\langle bought, S[dcl] \rightarrow NP\ S[dcl]\backslash NP\rangle + > 2$
Word + Distance(punct)	$\langle bought, S[dcl] \rightarrow NP\ S[dcl]\backslash NP\rangle + 2$
Word + Distance(verbs)	$\langle bought, S[dcl] \rightarrow NP\ S[dcl]\backslash NP\rangle + 0$
POS + Distance(words)	$\langle VBD, S[dcl] \rightarrow NP\ S[dcl]\backslash NP\rangle + > 2$
POS + Distance(punct)	$\langle VBD, S[dcl] \rightarrow NP\ S[dcl]\backslash NP\rangle + 2$
POS + Distance(verbs)	$\langle VBD, S[dcl] \rightarrow NP\ S[dcl]\backslash NP\rangle + 0$

Description of parser



Demonstration

http://svn.ask.it.usyd.edu.au/trac/candc/wiki/Demo

Results

Supertagger ambiguity and accuracy on section00

β	k	CATS/WORD	ACC	SENT ACC	ACC(POS)	SENT ACC
0.075	20	1.27	97.34	67.43	96.34	60.27
0.030	20	1.43	97.92	72.87	97.05	65.50
0.010	20	1.72	98.37	77.73	97.63	70.52
0.005	20	1.98	98.52	79.25	97.86	72.24
0.001	150	3.57	99.17	87.19	98.66	80.24

Results

Parsing accuracy on DepBank

	CCG parser			CCGbank			
Relation	Prec	Rec	F	Prec	Rec	F	# GRs
dependent	84.07	82.19	83.12	88.83	84.19	86.44	10,696
aux	95.03	90.75	92.84	96.47	90.33	93.30	400
conj	79.02	75.97	77.46	83.07	80.27	81.65	595
ta	51.52	11.64	18.99	62.07	12.59	20.93	292
det	95.23	94.97	95.10	97.27	94.09	95.66	1,114
arg_mod	81.46	81.76	81.61	86.75	84.19	85.45	8,295
mod	71.30	77.23	74.14	77.83	79.65	78.73	3,908
ncmod	73.36	78.96	76.05	78.88	80.64	79.75	3,550
xmod	42.67	53.93	47.64	56.54	60.67	58.54	178
cmod	51.34	57.14	54.08	64.77	69.09	66.86	168
pmod	0.00	0.00	0.00	0.00	0.00	0.00	12
arg	85.76	80.01	82.78	89.79	82.91	86.21	4,387

DepBank: Parc Dependency Bank
[King et al. 2003]

Results

subj_or_dobj	86.08	83.08	84.56	91.01	85.29	88.06	3,127
subj	84.08	75.57	79.60	89.07	78.43	83.41	1,363
nesubj	83.89	75.78	79.63	88.86	78.51	83.37	1,354
xsubj	0.00	0.00	0.00	50.00	28.57	36.36	7
csubj	0.00	0.00	0.00	0.00	0.00	0.00	2
comp	86.16	81.71	83.88	89.92	84.74	87.25	3,024
obj	86.30	83.08	84.66	90.42	85.52	87.90	2,328
dobj	87.01	88.44	87.71	92.11	90.32	91.21	1,764
obj2	68.42	65.00	66.67	66.67	60.00	63.16	20
iobj	83.22	65.63	73.38	83.59	69.81	76.08	544
clausal	77.67	72.47	74.98	80.35	77.54	78.92	672
xcomp	77.69	74.02	75.81	80.00	78.49	79.24	381
ccomp	77.27	70.10	73.51	80.81	76.31	78.49	291
pcomp	0.00	0.00	0.00	0.00	0.00	0.00	24
macroaverage	65.71	62.29	63.95	71.73	65.85	68.67	
microaverage	81.95	80.35	81.14	86.86	82.75	84.76	

Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

Study materials

Course materials and homeworks are available on the following web site

`https://is.muni.cz/course/fi/autumn2011/IA161`

Outline

- Introduction to Statistical parsing methods
- Statistical Parsers
  - RASP system
  - Stanford parser
  - Collins parser
  - Charniak parser
  - Berkeley parser

1. Introduction to statistical parsing

- The main theoretical approaches behind modern statistical parsers
- Over the last 12 years statistical parsing has succeeded significantly!
- NLP researchers have produced a range of statistical parsers

$\rightarrow$ wide-coverage and robust parsing accuracy

- They continues to improve the parsers year on year.

Application domains of statistical parsing

- Question answering systems of high precision
- Named entity extraction
- Syntactically based sentence compressions
- Extraction of people’s opinion about products
- Improved interaction in computer ganes
- Helping linguists find data

NLP parsing problem and solution

- The structure of language is ambiguous!

$\rightarrow$ local and global ambiguities

- **Classical parsing problem**

$\rightarrow$ simple 10 grammar rules can generate 592 parsers

$\rightarrow$ real size wide-coverage grammar generates millions of parses

NLP parsing problem and solution

NLP parsing solution

We need mechanisms that allow us to find the most likely parses

$\rightarrow$ statistical parsing lets us work with very loose grammars that admit millions of parses for sentences but to still quickly find the best parses

Improved methodology for robust parsing

The annotated data: Penn Treebank (early 90's)

- Building a treebank seems a lot slower and less useful than building a grammar
- But it has many helpful things
  - Reusability of the labor
  - Broad coverage
  - Frequencies and distributional information
  - A way to evaluate systems

Characterization of Statistical parsing

- What the grammar which determines the set of legal syntactic structures for a sentence? How is that grammar obtained?
- What is the algorithm for determining the set of legal parses for a sentence?
- What is the model for determining the probability of different parses for a sentence?
- What is the algorithm, given the model and a set of possible parses which finds the best parse?

Characterization of Statistical parsing

$$T_{\text{best}} = \text{arg max } Score(T, S)$$

Two components:

- The **model**: a function Score which assigns scores (probabilities) to tree and sentence pairs
- The **parser**: the algorithm which implements the search for T_{best}

Characterization of Statistical parsing

Statistical parsing seen as more of a **pattern recognition**/**Machine Learning** problem plus search

The grammar is only implicitly defined by the training data and the method used by the parser for generating hypotheses

Statistical parsing models

Probabilistic approach would suggest the following for the Score function

$$Score(T, S) = P(T|S)$$

Lots of research on different probability models for Penn Treebank trees

- Generative models, log-linear (maximum entropy) models, ...

2. Statistical parsers

- Many kinds of parsers based on the statistical methods:probability, machine learning
- Different objectives: research, commercial, pedagogical
  - RASP, Stanford parser, Berkeley parser,

RASP system

Robust Accurate Statistical Parsing (2nd release):
[Briscoe&Carroll, 2002; Briscoe *et al.* 2006]

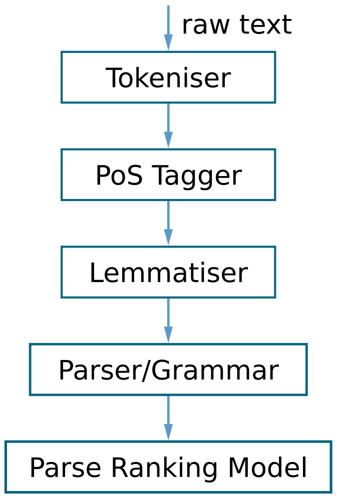
- system for syntactic annotation of free text
- Semantically-motivated output representation
- Enhanced grammar and part-of-speech tagger lexicon
- Flexible and semi-supervised training method for structural parse ranking model

Useful links to RASP

`http://ilexir.co.uk/applications/rasp/download/`

`http://www.informatics.susx.ac.uk/research/groups/nlp/rasp/`

Components of system



- Input:

unannotated text or transcribed (and punctuated) speech

- 1st step:

sentence boundary detection and tokenisation modules

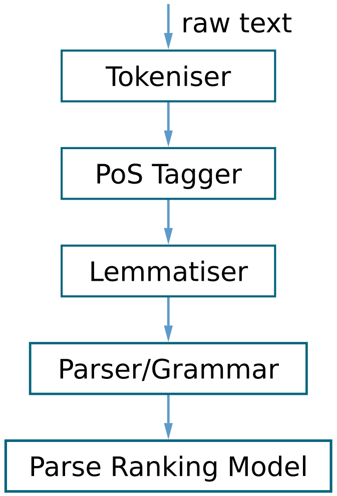
- 2nd step:

Tokenized text is tagged with one of 150 POS and punctuation labels (derived from the CLAWS tagset)

$\rightarrow$ first-order ('bigram') HMM tagger

$\rightarrow$ trained on the manually corrected tagged version of the Susanne, LOB and BNC corpora

Components of system



- 3rd step:

Morphological analyzer

- 4th step:

Manually developed wide-coverage tag sequence grammar in the parser

$\rightarrow$ 689 unification based phrase structure rules

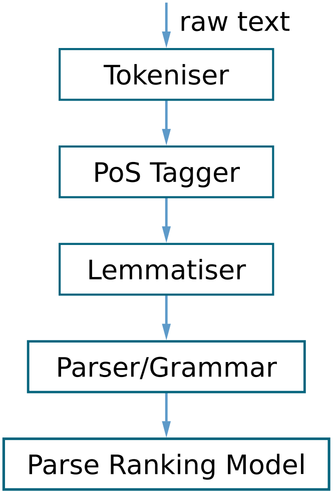
$\rightarrow$ preterminals to this grammar are the POS and punctuation tags

$\rightarrow$ terminals are featural description of the preterminals

$\rightarrow$ non-terminals project information up the tree using an X-bar scheme with 41 attributes with a maximum of 33 atomic

values

Components of system



- 5th step:

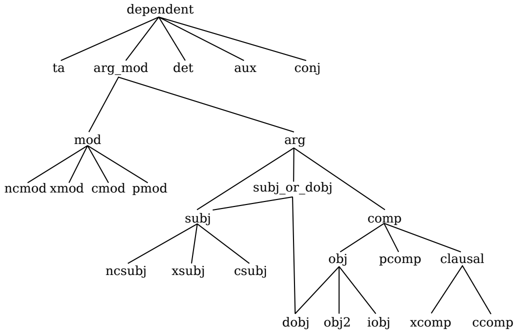
Generalized LR Parser

→ a non-deterministic LALR table is constructed automatically from CF 'backbone' compiled from the featurebased grammar

→ the parser builds a packed parse forest using this table to guide the actions it performs

→ the n-best parses can be efficiently extracted by unpacking sub-analyses, following pointers to contained subanalyses and choosing alternatives in order of probabilistic ranking

Components of system



- Output:

set of named grammatical relations (GRs)

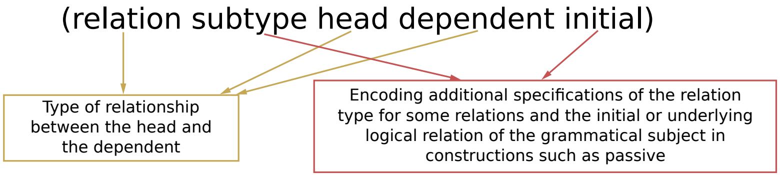
→ resulting set of ranked parses can be displayed or passed on for further processing

→ transformation of derivation trees into a set of named GRs

→ GR scheme captures those aspects of predicate-argument structure

Evaluation

- The system has been evaluated using the re-annotation of the PARC dependency bank (DepBank, King *et al.*, 2003)
- It consists of 560 sentences chosen randomly from section 23 of the WSJ with grammatical relations compatible with RASP system.
- Form of relations


(relation subtype head dependent initial)
Type of relationship between the head and the dependent
Encoding additional specifications of the relation type for some relations and the initial or underlying logical relation of the grammatical subject in constructions such as passive

Evaluation

Relation	Precision	Recall	F_1	std GRs
dependent	79.76	77.49	78.61	10696
aux	93.33	91.00	92.15	400
conj	72.39	72.27	72.33	595
ta	42.61	51.37	46.58	292
det	87.73	90.48	89.09	1114
arg_mod	79.18	75.47	77.28	8295
mod	74.43	67.78	70.95	3908
ncmod	75.72	69.94	72.72	3550
xmod	53.21	46.63	49.70	178
cmod	45.95	30.36	36.56	168
pmod	30.77	33.33	32.00	12
arg	77.42	76.45	76.94	4387
subj_or_dobj	82.36	74.51	78.24	3127
subj	78.55	66.91	72.27	1363
ncsubj	79.16	67.06	72.61	1354
xsubj	33.33	28.57	30.77	7
csubj	12.50	50.00	20.00	2
comp	75.89	79.53	77.67	3024
obj	79.49	79.42	79.46	2328
dobj	83.63	79.08	81.29	1764
obj2	23.08	30.00	26.09	20
iobj	70.77	76.10	73.34	544
clausal	60.98	74.40	67.02	672
xcomp	76.88	77.69	77.28	381
ccomp	46.44	69.42	55.55	291
pcomp	72.73	66.67	69.57	26
macroaverage	62.12	63.77	62.94	
microaverage	77.66	74.98	76.29	

Parsing accuracy on DepBank [Briscoe *et al.*, 2006]

- Micro-averaged precision, recall and F_1 score are calculated from the counts for all relations in the hierarchy
- Macro-averaged scores are the mean of the individual scores for each relation
- Micro-averaged F_1 score of 76.3% across all relations

Stanford parser

Java implementation of probabilistic natural language parsers (version 1.6.9)
: [Klein and Manning, 2003]

- Parsing system for English and has been used in Chinese, German, Arabic, Italian, Bulgarian, Portuguese
- Implementation, both highly optimized PCFG and lexicalized dependency parser, and lexicalized PCFG parser
- Useful links

`http://nlp.stanford.edu/software/lex-parser.shtml`
`http://nlp.stanford.edu:8080/parser/`

Stanford parser

- Input

various form of plain text

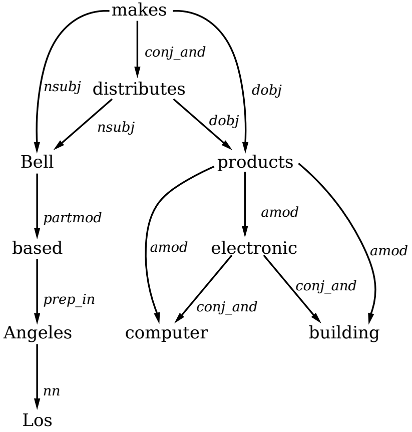
- Output

Various analysis formats

→ Stanford Dependencies (SD): typed dependencies as GRs

→ phrase structure trees

→ POS tagged text



Graphical representation of the SD for the sentence

“Bell, based in Los Angeles, makes and distributes electronic, computer and building products.”

Standford typed dependencies [De Marmette and Manning, 2008]

- provide a simple description of the grammatical relationships in a sentence
- represents all sentence relationships uniformly as typed dependency relations
- quite accessible to non-linguists thinking about tasks involving information extraction from text and is quite effective in relation extraction applications.

Standford typed dependencies [De Marnette and Manning, 2008]

For an example sentence:

Bell, based in Los Angeles, makes and distributes electronic, computer and building products.

Stanford Dependencies (SD) representation is:

nsubj(makes-8, Bell-1)
nsubj(distributes-10, Bell-1)
partmod(Bell-1, based-3)
nn(Angeles-6, Los-5)
prep_in(based-3, Angeles-6)
root(ROOT-0, makes-8)

conj_and(makes-8, distributes-10)
amod(products-16, electronic-11)
conj_and(electronic-11, computer-13)
amod(products-16, computer-13)
conj_and(electronic-11, building-15)
amod(products-16, building-15)
dobj(makes-8, products-16)
dobj(distributes-10, products-16)

Output

A lineup of masseurs was waiting to take the media in hand.

POS tagged text

Parsing [sent. 4 len. 13]: [A, lineup, of, masseurs, was, waiting, to, take, the, media, in, hand, .]

CFPSG representation

```
(ROOT
  (S
    (NP
      (NP (DT A) (NN lineup))
      (PP (IN of)
        (NP (NNS masseurs))))
    (VP (VBD was)
      (VP (VBG waiting)
        (S
          (VP (TO to)
            (VP (VB take)
              (NP (DT the) (NNS media))
              (PP (IN in)
                (NP (NN hand))))))))
    (. .)))
```

Typed dependencies representation

```
det(lineup2, A1)
nsubj(waiting6, lineup2)
xsubj(take8, lineup2)
prep_of(lineup2, masseurs4)
aux(waiting6, was5)
root(ROOT0, waiting6)
aux(take8, to7)
xcomp(waiting6, take8)
det(media10, the9)
dobj(take8, media10)
prep_in(take8, hand12)
```

Berkeley parser

Learning PCFGs, statistical parser (release 1.1, version 09.2009)

: [Petrov *et al.*, 2006; Petrov and Klein, 2007]

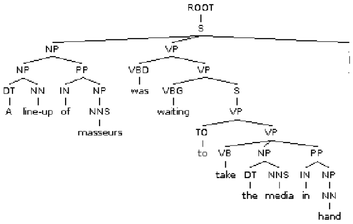
- Parsing system for English and has been used in Chinese, German, Arabic, Bulgarian, Portuguese, French
- Implementation of unlexicalized PCFG parser
- Useful links

```
http://nlp.cs.berkeley.edu/
http://tomato.banatao.berkeley.edu:
8080/parser/parser.html
http://code.google.com/p/berkeleyparser/
```

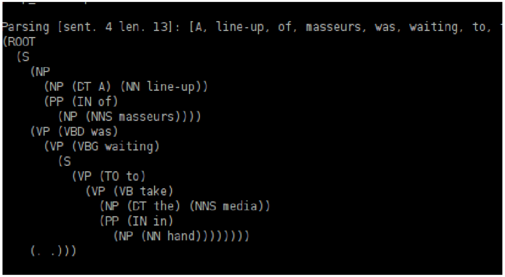
Comparison of parsing an example sentence

A lineup of masseurs was waiting to take the media in hand.

```
(ROOT
  (S
    (NP
      (NP (DT A) (NN line-up))
      (PP (IN of)
        (NP (NNS masseurs))))
    (VP (VBD was)
      (VP (VBG waiting)
        (S
          (VP (TO to)
            (VP (VB take)
              (NP (DT the) (NNS media))
              (PP (IN in)
                (NP (NN hand))))))))
    (. .)))
```



Berkeley parser



```
Parsing [sent. 4 len. 13]: [A, line-up, of, masseurs, was, waiting, to,
(ROOT
  (S
    (NP
      (NP (DT A) (NN line-up))
      (PP (IN of)
        (NP (NNS masseurs))))
    (VP (VBD was)
      (VP (VBG waiting)
        (S
          (VP (TO to)
            (VP (VB take)
              (NP (DT the) (NNS media))
              (PP (IN in)
                (NP (NN hand))))))))
    (. .)))
```

Stanford parser

charniak parser

Probabilistic LFG F-Structure Parsing
: [Charniak, 2000; Bikel, 2002]

- Parsing system for English
- PCFG based wide coverage LFG parser
- Useful links

```
http://nclt.computing.dcu.ie/demos.html
http://lfg-demo.computing.dcu.ie/lfgparser.html
```

Collins parser

Head-Driven Statistical Models for natural language parsing (Release 1.0, version 12.2002)
: [Collins, 1999]

- Parsing system for English
- Useful links

`http://www.cs.columbia.edu/~mcollins/code.html`

Bikel's parser

Multilingual statistical parsing engine (release 1.0, version 06.2008)
: [Charniak, 2000; Bikel, 2002]

- Parsing system for English, Chinese, Arabic, Korean

```
http://www.cis.upenn.edu/~dbikel/#stat-parser
http://www.cis.upenn.edu/~dbikel/software.html
```

Comparing parser speed on section 23 of WSJ Penn Treebank

Parser	Time (min.)
Collins	45
Charniak	28
Sagae	11
CCG	1.9

Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

Study materials

- Course materials and homeworks are available on the following web site:
  https://is.muni.cz/course/fi/autumn2011/IA161
- Refer to *Dependency Parsing*, Synthesis: Lectures on Human Language Technologies, S. kübler, R. McDonald and J. Nivre, 2009

Outline

- Introduction to Dependency parsing methods
- Dependency Parsers

Introduction to Dependency parsing

- **Motivation**

  a. dependency-based syntactic representation seem to be useful in many applications of language technology: machine translation, information extraction

     $\rightarrow$ transparent encoding of predicate-argument structure

  b. dependency grammar is better suited than phrase structure grammar for language with free or flexible word order

     $\rightarrow$ analysis of diverse languages within a common framework

  c. leading to the development of accurate syntactic parsers for a number of languages

     $\rightarrow$ combination with machine learning from syntactically annotated corpora (e.g. treebank)

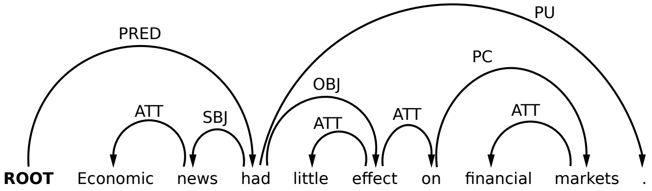
Introduction to Dependency parsing

- **Dependency parsing**

  "Task of automatically analyzing the dependency structure of a given input sentence"

- **Dependency parser**

  "Task of producing a labeled dependency structure of the kind depicted in the follow figure, where the words of the sentence are connected by typed dependency relations"



Definitions of dependency graphs and dependency parsing

Dependency graphs: syntactic structures over sentences

Def. 1.: A sentence is a sequence of tokens denoted by

$$S = w_0 w_1 \ldots w_n$$

Def. 2.: Let $R = \{r_1, \ldots, r_m\}$ be a finite set of *possible dependency relation types* that can hold between any two words in a sentence. A relation type $r \in R$ is additionally called an *arc label*.

Definitions of dependency graphs and dependency parsing

Dependency graphs: syntactic structures over sentences

Def. 3.: A dependency graph $G = (V, A)$ is a labeled directed graph, consists of nodes, V, and arcs, A, such that for sentence $S = w_0 w_1 \dots w_n$ and label set R the following holds:

1. $V \subseteq \{w_0 w_1 \dots w_n\}$
2. $A \subseteq V \times R \times V$
3. if $(w_i, r, w_j) \in A$ then $(w_i, r', w_j) \notin A$ for all $r' \neq r$

Approach to dependency parsing

a. **data-driven**
it makes essential use of machine learning from linguistic data in order to parse new sentences

b. **grammar-based**
it relies on a formal grammar, defining a formal language, so that it makes sense to ask whether a given input is in the language defined by the grammar or not.

$\rightarrow$ **Data-driven have attracted the most attention in recent years.**

Data-driven approach

according to the *type of parsing model* adopted,
the algorithms used to learn the model from data
the algorithms used to parse new sentences with the model

a. **transition-based**
start by defining a transition system, or state machine, for mapping a sentence to its dependency graph.

b. **graph-based**
start by defining a space of candidate dependency graphs for a sentence.

Data-driven approach

a. **transition-based**
- **learning problem:** induce a model for predicting the next state transition, given the transition history
- **parsing problem:** construct the optimal transition sequence for the input sentence, given induced model

b. **graph-based**
- **learning problem:** induce a model for assigning scores to the candidate dependency graphs for a sentence
- **parsing problem:** find the highest-scoring dependency graph for the input sentence, given induced model

Transition-based Parsing

- Transition system consists of a set C of parser configurations and of a set D of transitions between configurations.
- **Main idea:** a sequence of valid transitions, starting in the *initial configuration* for a given sentence and ending in one of several *terminal configurations*, defines a valid dependency tree for the input sentence.

$$D_{1'm} = d_1(c_1), \dots, d_m(c_m)$$

Transition-based Parsing

- **Definition**
  Score of $D_{1'm}$ factors by configuration-transition pairs (c_i, d_i):

$$s(D_{1'm}) = \sum_{i=1}^{m} s(c_i, d_i)$$

- **Learning**
  Scoring function $s(c_i, d_i)$ for $d_i(c_i) \in D_{1'm}$
- **Inference**
  Search for highest scoring sequence $D^*_{1'm}$ given $s(c_i, d_i)$

Transition-based Parsing

Inference for transition-based parsing

- **Common inference strategies:**
  - Deterministic [Yamada and Matsumoto 2003, Nivre et al. 2004]
  - Beam search [Johansson and Nugues 2006, Titov and Henderson 2007]
  - Complexity given by upper bound on transition sequence length
- **Transition system**
  - Projective O(n) [Yamada and Matsumoto 2003, Nivre 2003]
  - Limited non-projective O(n) [Attardi 2006, Nivre 2007]
  - Unrestricted non-projective O(n2) [Nivre 2008, Nivre 2009]

Transition-based Parsing

Learning for transition-based parsing

- **Typical scoring function:**
  - $s(c_i, d_i) = w \cdot f(c_i, d_i)$ where $f(c_i, d_i)$ is a feature vector over configuration c_i and transition d_i and w is a weight vector $[w_i = \text{ weight of feature} f_i(c_i, d_i)]$
- **Transition system**
  - Projective O(n) [Yamada and Matsumoto 2003, Nivre 2003]
  - Limited non-projective O(n) [Attardi 2006, Nivre 2007]
  - Unrestricted non-projective O(n2) [Nivre 2008, Nivre 2009]
- **Problem**
  - Learning is local but features are based on the global history

Graph-based Parsing

- For a input sentence S we define a graph $G_s = (V_s, A_s)$ where $V_s = \{w_0, w_1, \dots, w_n\}$ and $A_s = \{(w_i, w_j, l) | w_i, w_j \in V \text{ and } l \in L\}$
- Score of a dependency tree T factors by subgraphs $G_s, \dots, Gs$:

$$s(T) = \sum_{i-1}^{m} s(G_i)$$

- Learning: **Scoring function** $s(G_i)$ for a subgraph $G_i \in T$
- Inference: Search for maximum spanning tree scoring sequence T^* of G_s given $s(G_i)$

Graph-based Parsing

Learning graph-based models

- **Typical scoring function:**
  - $s(G_i) = w \cdot f(G_i)$ where $f(G_i)$ is a high-dimensional feature vector over subgraphs and w is a weight vector
    $[w_j =$ weight of feature $f_j(G_i)]$
- **Structured learning** [McDonald et al. 2005a, Smith and Johnson 2007]:
  - Learn weights that maximize the score of the correct dependency tree for every sentence in the training set
- **Problem**
  - Learning is global (trees) but features are local (subgraphs)

Grammar-based approach

a. **context-free dependency parsing**
exploits a mapping from dependency structures to CFG structure representations and reuses parsing algorithms originally developed for CFG $\rightarrow$ chart parsing algorithms

b. **constraint-based dependency parsing**

- parsing viewed as a constraint satisfaction problem
- grammar defined as a set of constraints on well-formed dependency graphs
- finding a dependency graph for a sentence that satisfies all the constraints of the grammar (having the best score)

Grammar-based approach

a. **context-free dependency parsing**

Advantage: Well-studied parsing algorithms such as CKY, Earley's algorithm can be used for dependency parsing as well.

$\rightarrow$ need to convert dependency grammars into efficiently parsable context-free grammars; (e.g. bilexical CFG, Eisner and Smith, 2005)

b. **constraint-based dependency parsing**

defines the problem as constraint satisfaction

- Weighted constraint dependency grammar (WCDG, Foth and Menzel, 2005)
- Transformation-based CDG

Dependency parsers

- **Trainable parsers**
  - Probabilistic dependency parser (Eisner, 1996, 2000)
  - MSTParser (McDonald, 2006)-graph-based
  - MaltParser (Nivre, 2007, 2008)-transition-based
  - K-best Maximum Spanning Tree Dependency Parser (Hall, 2007)
  - Vine Parser
  - ISBN Dependency Parser
- **Parsers for specific languages** defines the problem as constraint satisfaction
  - Minipar (Lin, 1998)
  - WCDG Parser (Foth *et al.*, 2005)
  - Pro3Gres (Schneider, 2004)
  - Link Grammar Parser (Lafferty *et al.*, 1992)
  - CaboCha (Kudo and Matsumoto, 2002)

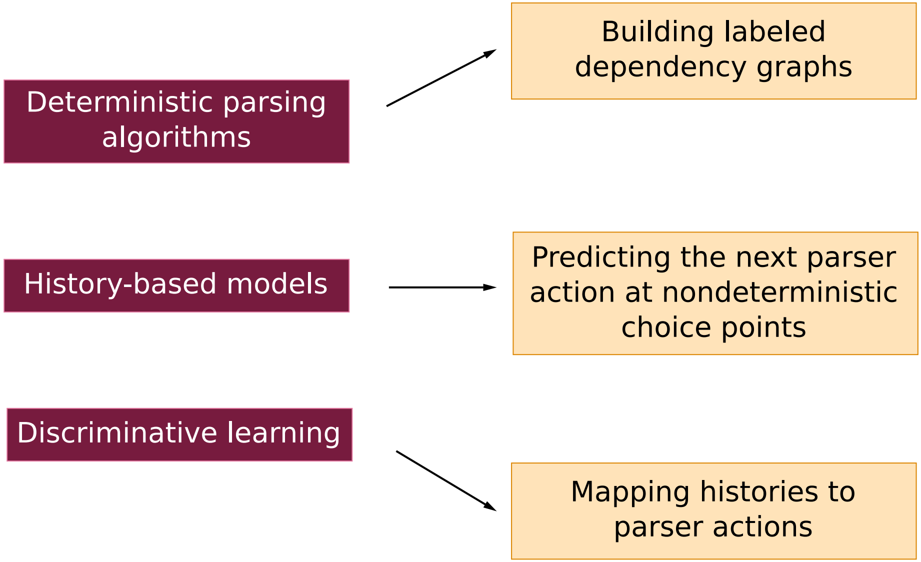
MaltParser

Data-driven dependency parsing system (Last version, 1.6.1, J. Hall, J. Nilsson and J. Nivre)

- Transition-based parsing system
- Implementation of inductive dependency parsing
- Useful for inducing a parsing model from treebank data
- Useful for parsing new data using an induced model

Useful links
http://maltparser.org

Components of system


Deterministic parsing algorithms
Building labeled dependency graphs
History-based models
Predicting the next parser action at nondeterministic choice points
Discriminative learning
Mapping histories to parser actions

MSTParser

Running system

- Input: part-of-speech tags or word forms

1	Den	_	PO	PO	DP	2	SS	_	_
2	blir	_	V	BV	PS	0	ROOT	_	_
3	gemensam	_	AJ	AJ	_	2	SP	_	_
4	für	_	PR	PR	_	2	OA	_	_
5	alla	_	PO	PO	TP	6	DT	_	_
6	inkomsttagare	_	N	NN	HS	4	PA	_	_
7	oavsett	_	PR	PR	_	2	AA	_	_
8	civilständ	_	N	NN	SS	7	PA	_	_
9	.	_	P	IP	_	2	IP	_	_

- Output: column containing a dependency label

MSTParser

Minimum Spanning Tree Parser (Last version, 0.2, R. McDonald et al., 2005, 2006)

- Graph-based parsing system

Useful links
http://www.seas.upenn.edu/ strctlrn/MSTParser/MSTParser.html

MSTParser

Running system

- Input data format:

w1	w2	...	wn
p1	p2	...	pn
l1	l2	...	ln
d1	d2	...	d2

Where,

- w1 ... wn are the n words of the sentence (tab deliminated)
- p1 ... pn are the POS tags for each word
- l1 ... ln are the labels of the incoming edge to each word
- d1 ... dn are integers representing the postition of each words parent

- Example:

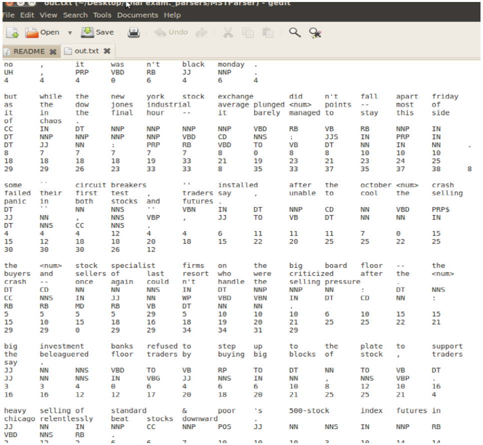
For example, the sentence ”John hit the ball” would be:

John	hit	the	ball
N	V	D	N
SBJ	ROOT		MOD OBJ
2	0	4	2

MSTParser

Running system

- Output: column containing a dependency label


out.txt (~/Desktop/final exam._parsers/MSTParser) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
README
out.txt

```
no      ,       it      was     n't     black   monday  .
UH      ,       PRP     VBD     RB      JJ      NNP     .
4       4       4       0       6       4       6       4

but     while   the     new     york    stock   exchange        did     n't     fall    apart   friday
as      the     dow     jones   industrial      average plunged <num>   points  --      most    of
it      in      the     final   hour    --      it      barely  managed to      stay    this    side
of      chaos   .
CC      IN      DT      NNP     NNP     NNP     NNP     VBD     RB      VB      RB      NNP     IN
DT      NNP     NNP     NNP     NNP     VBD     CD      NNS     :       JJS     IN      PRP     IN
DT      JJ      NN      :       PRP     RB      VBD     TO      VB      DT      NN      IN      NN      .
8       7       7       7       7       7       8       0       8       8       10      10      10
18      18      18      18      19      33      21      19      23      21      23      24      25
29      29      26      23      33      33      8       35      33      37      35      37      38      8

some    ``      circuit breakers        ''      installed       after   the     october <num>   crash
failed  their   first   test    ,       traders say     ,       unable  to      cool    the     selling
panic   in      both    stocks  and     futures .
DT      ``      NN      NNS     ''      VBN     IN      DT      NNP     CD      NN      VBD     PRP$
JJ      NN      ,       NNS     VBP     ,       JJ      TO      VB      DT      NN      NN      IN
DT      NNS     CC      NNS     .
4       4       4       12      4       4       6       11      11      11      7       0       15
15      12      18      18      20      18      15      22      20      25      25      22      25
30      30      30      26      12

the     <num>   stock   specialist      firms   on      the     big     board   floor   --      the
buyers  and     sellers of      last    resort  who     were    criticized      after   the     <num>
crash   --      once    again   could   n't     handle  the     selling pressure        .
DT      CD      NN      NN      NNS     IN      DT      NNP     NNP     NN      :       DT      NNS
CC      NNS     IN      JJ      NN      WP      VBD     VBN     IN      DT      CD      NN      :
RB      RB      MD      RB      VB      DT      NN      NN      .
5       5       5       5       29      5       10      10      10      6       10      15      15
15      10      15      18      16      18      19      20      21      25      25      22      21
29      29      0       29      29      34      34      31      29

big     investment      banks   refused to      step    up      to      the     plate   to      support
the     beleaguered     floor   traders by      buying  big     blocks  of      stock   ,       traders
say     .
JJ      NN      NNS     VBD     TO      VB      RP      TO      DT      NN      TO      VB      DT
JJ      NN      NNS     IN      VBG     JJ      NNS     IN      NN      ,       NNS     VBP     .
3       3       4       0       6       4       6       6       10      8       12      10      16
16      16      12      12      17      20      18      20      21      25      25      21      4

heavy   selling of      standard        &       poor    's      500-stock       index   futures in
chicago relentlessly    beat    stocks  downward        .
JJ      NN      IN      NNP     CC      NNP     POS     JJ      NN      NNS     IN      NNP     RB
VBD     NNS     RB      .
2       12      2       6       6       7       10      10      10      3       10      14      14
```

Comparing parsing accuracy

Graph-based Vs. Transition-based MST Vs. Malt

Language	MST	Malt
Arabic	**66.91**	66.71
Bulgarian	**87.57**	87.41
Chinese	85.90	**86.92**
Czech	**80.18**	78.42
Danish	**84.79**	84.77
Dutch	**79.19**	78.59
German	**87.34**	85.82
Japanese	90.71	**91.65**
Portuguese	86.82	**87.60**
Slovene	**73.44**	70.30
Spanish	**82.25**	81.29
Swedish	82.55	**84.58**
Turkish	63.19	**65.68**
Average	**80.83**	80.75

Presented in *Current Trends in Data-Driven Dependency Parsing* by Joakim Nivre, 2009

Link Parser

Syntactic parser of English, based on the Link Grammar (version, 4.7.4, Feb. 2011, D. Temperley, D, Sleator, J. Lafferty, 2004)

- Words as blocks with connectors + or -
- Words rules for defining the connection between the connectors
- Deep syntactic parsing system

Useful links
http://www.link.cs.cmu.edu/link/index.html
http://www.abisource.com/

Link Parser

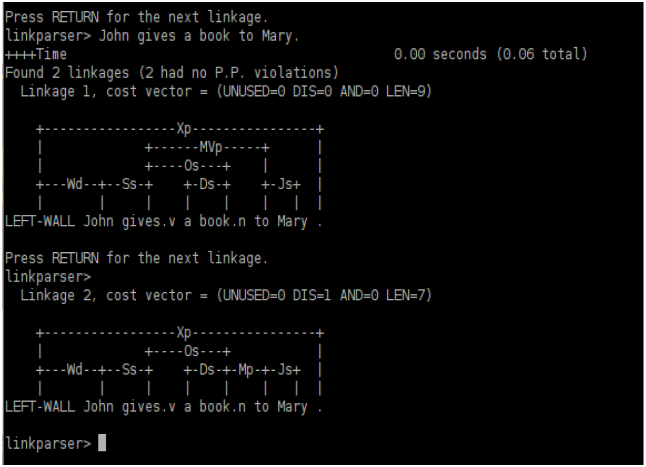
- Example of a parsing in the Link Grammar:

  let’s test our proper sentences!

http://www.link.cs.cmu.edu/link/submit-sentence-4.html

Link Parser

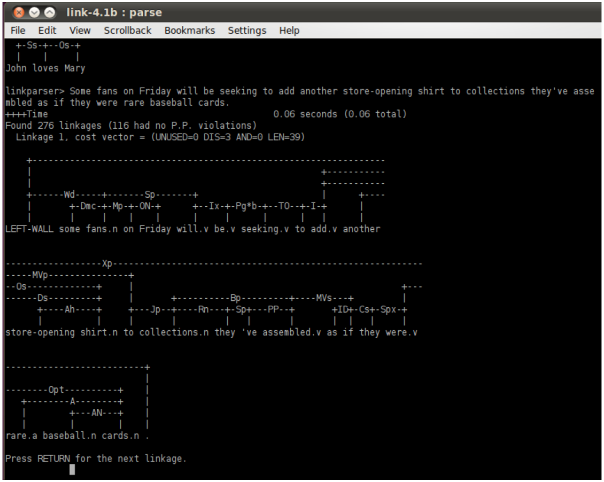
John gives a book to Mary.



```
Press RETURN for the next linkage.
linkparser> John gives a book to Mary.
++++Time                                          0.00 seconds (0.06 total)
Found 2 linkages (2 had no P.P. violations)
  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=9)

    +-----------------Xp---------------+
    |             +------MVp-----+     |
    |             +----Os---+    |     |
    +---Wd--+--Ss-+    +-Ds-+    +-Js+ |
    |       |     |    |    |    |   | |
LEFT-WALL John gives.v a book.n to Mary .

Press RETURN for the next linkage.
linkparser>
  Linkage 2, cost vector = (UNUSED=0 DIS=1 AND=0 LEN=7)

    +-----------------Xp---------------+
    |             +----Os---+          |
    +---Wd--+--Ss-+    +-Ds-+-Mp-+-Js+ |
    |       |     |    |    |    |   | |
LEFT-WALL John gives.v a book.n to Mary .

linkparser> █
```

Link Parser

Some fans on Friday will be seeking to add another store-opening shirt to collections they’ve assembled as if they were rare baseball cards.



```
  +-Ss-+--Os-+
  |    |     |
John loves Mary

linkparser> Some fans on Friday will be seeking to add another store-opening shirt to collections they've asse
mbled as if they were rare baseball cards.
++++Time                                          0.06 seconds (0.06 total)
Found 276 linkages (116 had no P.P. violations)
  Linkage 1, cost vector = (UNUSED=0 DIS=3 AND=0 LEN=39)

    +------------------------------------------------------------------
    |                                                      +-----------
    |                                                      +-----------
    +------Wd-----+-------Sp-------+                       |      +----
    |       +-Dmc-+-Mp-+-ON-+      +--Ix-+-Pg*b-+--TO--+-I-+      |
    |       |     |    |    |      |     |      |      |   |      |
LEFT-WALL some fans.n on Friday will.v be.v seeking.v to add.v another


------------------Xp----------------------------------------------------------
-----MVp---------------+
--Os-------------+     |                                                  +---
------Ds---------+     |       +----------Bp---------+----MVs---+         |
      +----Ah----+     +---Jp--+----Rn---+-Sp+---PP--+       +ID+-Cs+-Spx-+
      |          |     |       |         |   |       |       |  |   |     |
store-opening shirt.n to collections.n they 've assembled.v as if they were.v


--------------------------+
                          |
--------Opt----------+    |
   +--------A--------+    |
   |        +---AN---+    |
   |        |        |    |
rare.a baseball.n cards.n .

Press RETURN for the next linkage.
```

WCDG parser

Weighted Constraint Dependency Grammar Parser (version, 0.97-1, May, 2011, W. Menzel, N. Beuck, C. Baumgärtner)

- incremental parsing
- syntactic predictions for incomplete sentences
- Deep syntactic parsing system

Useful links
http://nats-www.informatik.uni-hamburg.de/view/CDG/ParserDemo

Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

Parsing Evaluation

Parsing Results

- usually some complex (i.e. non-scalar) structure, mostly a tree or a graph-like structure
- crucial question: how to measure the “goodness” of the result?

Extrinsic vs. Intrinsic Evaluation

- Intrinsic
  - by comparing to a “gold”, i.e. correct, representation
- Extrinsic
  - by exploiting the result in a 3rd party task and evaluating its results
- Which is better?

Intrinsic Evaluation – Phrase-Structure Syntax

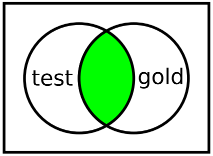
- i.e. compare two phrase-structure trees and tell a number
- PARSEVAL metric
- LAA (Leaf-ancestor assessment) metric

PARSEVAL metric

- basic idea: penalize crossing brackets in the tree
- i.e. compare all constituents in the test tree to the gold tree
- $\Rightarrow$ parsing viewed as classification problem

Precision, recall

- for classification problems in NLP, the standard evaluation is by means of precision and recall



$$\text{precision} = \frac{|\text{test} \cap \text{gold}|}{|\text{test}|} \quad \text{recall} = \frac{|\text{test} \cap \text{gold}|}{|\text{gold}|}$$

- two numbers, we just want to have one – F-score

$$\text{F}_1 \text{ score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

F-score

- also F-measure
- general form: F$_\beta$ score

$$\text{F}_\beta \text{ score} = (1+\beta^2) \cdot \frac{\text{precision}\cdot\text{recall}}{(\beta^2+\text{precision})+\text{recall}}$$

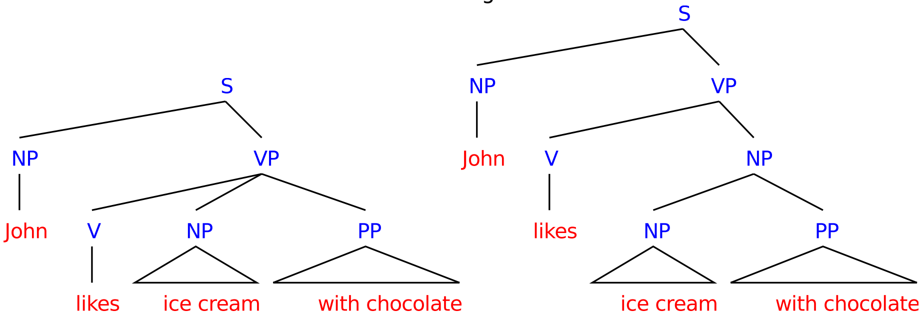
- special case of $\beta = 1$ corresponds to the harmonic mean of precision and recall
- β can be used for favouring precision over recall (for $\beta < 1$) or vice versa (for $\beta > 1$)

PARSEVAL metric

- basic idea: penalize crossing brackets in the tree
- i.e. compare all constituents in the test tree to the gold tree
- $\Rightarrow$ parsing viewed as classification problem
- $\Rightarrow$ F-score on correct bracketings/constituents
- might even disregard non-terminal names
- sort of standardized tool available: the `evalb` script at `http://nlp.cs.nyu.edu/evalb/`

PARSEVAL metric – example

test vs. gold

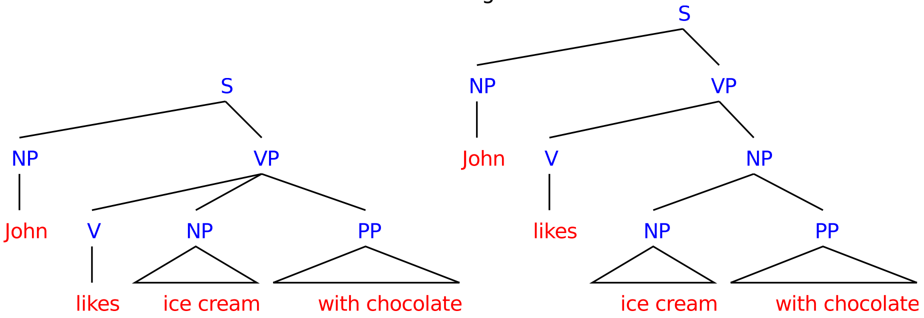


```
test:[S [NP John][VP [V likes][NP ice cream] [PP with chocolate]]]
gold:[S [NP John][VP [V likes][NP [NP ice cream] [PP with chocolate]]]]
```

precision = 6/6 = 1.0, recall = 6/7 = 0.86, F-score = 0.92

PARSEVAL metric

test vs. gold



```
test:[S [NP John][VP [V likes][NP ice cream] [PP with chocolate]]]
gold:[S [NP John][VP [V likes][NP [NP ice cream] [PP with chocolate]]]]
```

precision = 6/6 = 1.0, recall = 6/7 = 0.86, F-score = 0.92

PARSEVAL metric

- often subject to criticism (see e.g. Sampson, 2000)
- Sampson proposed another metric, the leaf-ancestor assessment (LAA)

LAA metric

- basic idea: for each leaf (word), compare the path to the root of the tree, compute the edit distance between both paths, finally take the average of all words
- in the previous example, the paths (lineages) are:
  - (John) NP S vs. (John) NP S
  - (likes) V VP S vs. (likes) V VP S
  - (ice cream) NP VP S vs. (ice cream) NP NP VP S
  - (with chocolate) PP VP S vs. (with chocolate) PP NP VP S

Intrinsic Evaluation – Dependency Syntax

- much easier
- just precision, labeled or unlabeled (as the number of correct dependencies)

Intrinsic Evaluation – Building Treebanks

- treebank = a syntactically annotated text corpus
- manual annotation according to some guidelines
- from the evaluation point of view: inter-annotator agreement (IAA) is a crucial property

Measuring IAA

- naïve approach: count how many times people agreed on
- problem: it does not account for agreement by chance

Chance-corrected coefficients for IAA

- S (Benett, Alpert and Goldstein, 1954)
- π (Scott, 1955)
- κ (Cohen, 1960)
- (there is lot of terminology confusion, we follow Ron Artstein, Massimo Poesio: Inter-coder Agreement for Computational Linguistics, 2008)
- A_o – observed agreement
- A_e – expected (chance) agreement
- for all coefficients, they compute:

$$S, \pi, \kappa = \frac{A_o - A_e}{1 - A_e}$$

Chance-corrected coefficients for IAA

- S (Benett, Alpert and Goldstein, 1954)
  - assumes that all categories and all annotators have uniform probability distribution
- π (Scott, 1955)
  - assumes that different categories have different distributions shared across annotators
- κ (Cohen, 1960)
  - assumes that different categories and different annotators have different distributions
- devised for 2 annotators, various modifications for more than 2 annotators available

Intrinsic Evaluation – Conclusions

- generally not easy
- builds on the assumption of having THE correct parse
- there is evidence that it does not correlate with extrinsic evaluation, i.e. how good the tool is for some particular job

Extrinsic Evaluation

- = evaluation on a particular task/application
- advantages: measures direct fitness for that task
- disadvantages: may not generalize for other tasks

- leads to crucial question: what can be parsing used for?

What can parsing be used for?

- in theory, (full) parsing is suitable/appropriate/necessary for many NLP tasks
- practically it turns out to be:
  - often not accurate enough
  - often too complicated to exploit
  - sometimes just an overkill compared to shallow parsing or yet simpler approaches

What can parsing be used for?

- in theory, (full) parsing is suitable/appropriate/necessary for many NLP tasks
    - information extraction
    - information retrieval
    - machine translation
    - corpus linguistics
    - computer lexicography
    - question answering
    - ...

Where is parsing actually used now?

- prototype systems
- academia work
- production systems ???

What to evaluate parsing on

Sample (more or less well defined) applications

- (partial) morphological disambiguation
- text correcting systems
- word sketches
- phrase extraction
- simple treebank of high IAA