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

{ Directional top-down
  Directional bottom-up

{ Non-directional top-down method
  – firstly by **Unger**
  Non-directional bottom-up
  – by Cocke, Younger and Kasami (**CYK, also CKY**)

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

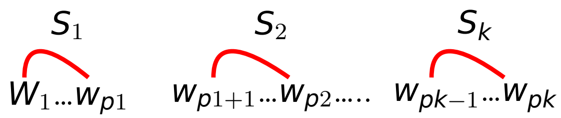
Non-directional top-down methods by Unger

- 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:
VP -> $V_{ditrans}$ NP PP_{to}
VP -> $V_{ditrans}$ NP 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 $O(n^3)$

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						
2						
3						
4						
5						
6						

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					
2						
3						
4						
5						
6						

Lecture 3

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						
3						
4						
5						
6						

IA161 Syntactic Formalisms for Parsing Natural Languages 12 / 36

Lecture 3

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					
3						
4						
5						
6						

IA161 Syntactic Formalisms for Parsing Natural Languages 12 / 36

Lecture 3

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						
4						
5						
6						

IA161 Syntactic Formalisms for Parsing Natural Languages 12 / 36

CKY example – solution

a b a a b a

Definition

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

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	$\emptyset$				
4						
5						
6						

CKY example – solution

a b a a b a

Definition

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

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						
5						
6						

CKY example – solution

a b a a b a

Definition

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

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					
5						
6						

CKY example – solution

a b a a b a

Definition

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

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				
5						
6						

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						
6						

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	$\emptyset$					
6						

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						

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					

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 → α • Y β, j), add (Y → • γ, k) to agenda for every production in the grammar with Y on the left-hand side (Y → γ).
2. Scanning
   If a is the next symbol in the input stream, for every state in agenda of the form (X → α • a β, j), add (X → α a • β, j) to agenda.
3. Completion
   For every state in agenda of the form (X → γ •, j), find states in agenda of the form (Y → α • X β, i) and add (Y → α X • β, i) to agenda.

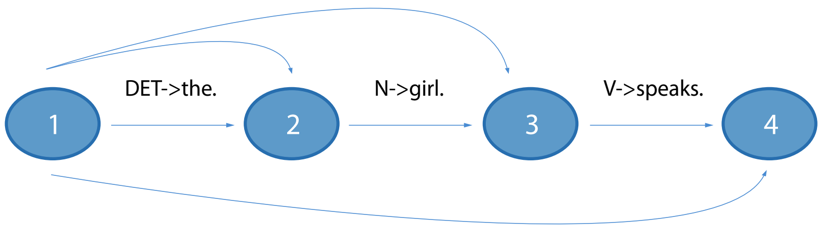
Earley algorithm

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 **The**	2 **girl**	3 **speaks**

Chart parsing

- The Earley parser can be modified to work bottom-up or head-corner
- ⇒ 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;
```


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:

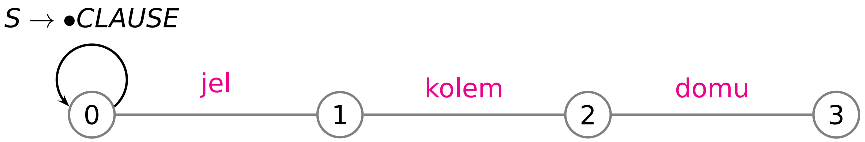
$$\begin{array}{rcl} S & \rightarrow & CLAUSE \\ CLAUSE & \rightarrow & V\ OPTPREP\ N \\ OPTPREP & \rightarrow & \epsilon \\ OPTPREP & \rightarrow & PREP \\ V & \rightarrow & jel \\ PREP & \rightarrow & kolem \\ N & \rightarrow & domu \\ N & \rightarrow & kolem \end{array}$$

Sentence:

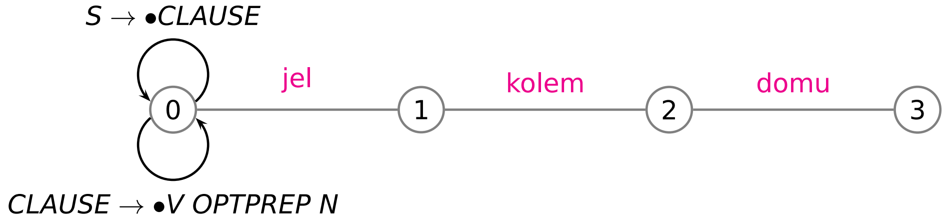
”jel kolem domu” (a_1=*jel*, a_2=*kolem*, a_3=*domu*).

Example – chart after top-down analysis

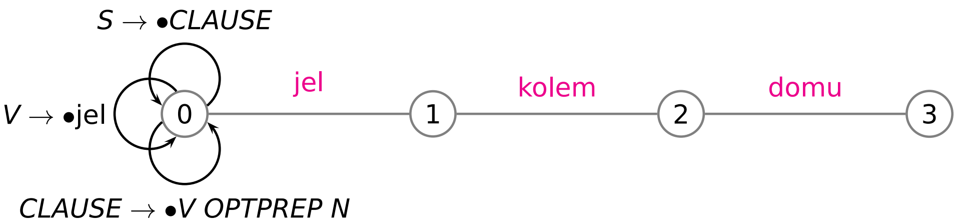
Example – chart after top-down analysis



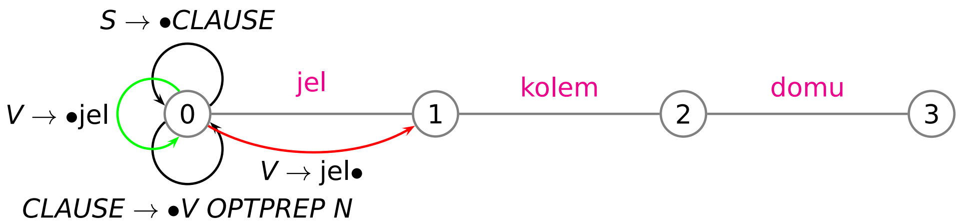
Example – chart after top-down analysis



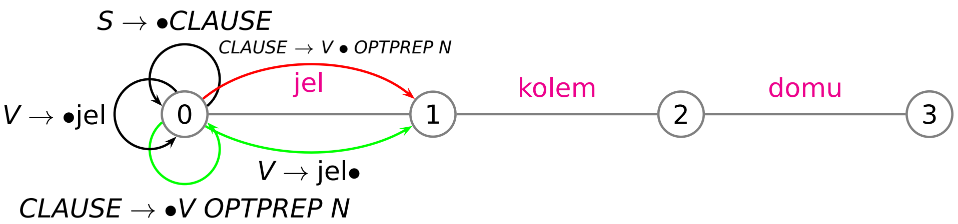
Example – chart after top-down analysis



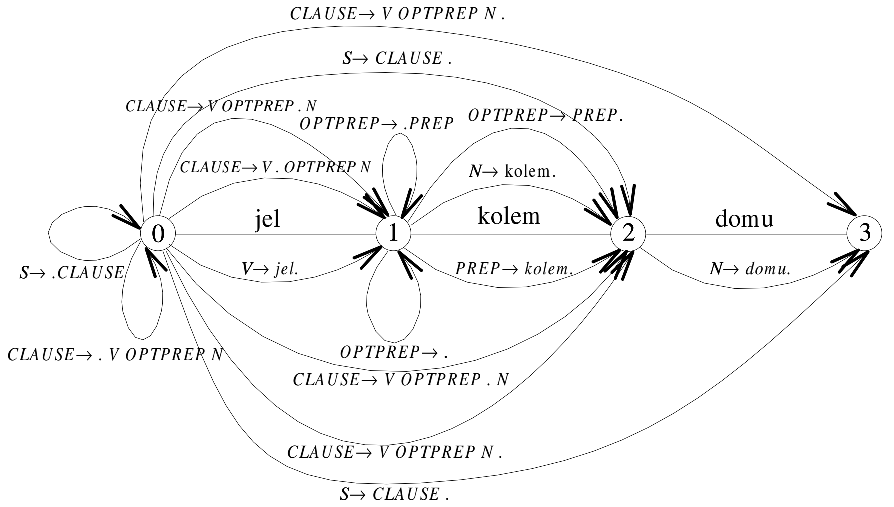
Example – chart after top-down analysis



Example – chart after top-down analysis



Example – chart after top-down analysis



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*].

Head-driven chart parsing

- Rule head – any particular right-hand side non-terminal E.g. in the rule *CLAUSE* → *V* <u>*OPTPREP*</u> *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

Lecture 3

PCFG

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

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

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

Lecture 3

Statistical parsing

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

Lecture 3

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.