School of CIT –
Social Computing
Research Group

Technical
University
of Munich

Natural Language Processing
IN2361

Prof. Dr. Georg Groh

Chapter 17
Constituency Grammars and Constituency Parsing

- content is based on [1]
- certain elements (e.g. equations or tables) were taken over or taken over in a modified form from [1]
- citations of [1] or from [1] are omitted for legibility
- errors are fully in the responsibility of Georg Groh
- BIG thanks to Dan and James for a great book!

Constituents

Constituents - groups of words behaving as a single unit

- example: noun phrases:

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

- evidence that these are constituents: can appear before verbs

three parties from Brooklyn *arrive*...
a high-class spot such as Mindy's *attracts*...
the Broadway coppers *love*...
they *sit*

while parts thereof cannot:

*from *arrive*...	*as *attracts*...
*the *is*...	*spot *sat*...

Constituents

- other evidence that indicates constituency: different positions possible as a whole: example: prepositional phrase:

On September seventeenth, I'd like to fly from Atlanta to Denver
I'd like to fly *on September seventeenth* from Atlanta to Denver
I'd like to fly from Atlanta to Denver *on September seventeenth*

while parts thereof cannot appear in different positions individually :

*On September, I'd like to fly seventeenth from Atlanta to Denver
*On I'd like to fly September seventeenth from Atlanta to Denver
*I'd like to fly on September from Atlanta to Denver seventeenth

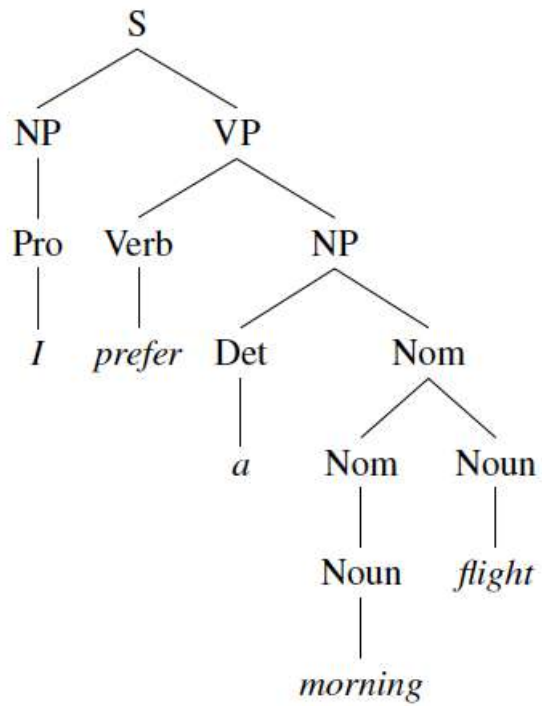
- Formal Grammars provide rules on how to construct and use constituents

Context Free Grammars - Example

Noun →	*flights* | *breeze* | *trip* | *morning* | *flight*
Verb →	*is* | *prefer* | *like* | *need* | *want* | *fly*
Adjective →	*cheapest* | *non-stop* | *first* | *latest* | *other* | *direct*
Pronoun →	*me* | *I* | *you* | *it*
Proper-Noun →	*Alaska* | *Baltimore* | *Los Angeles* | *Chicago* | *United* | *American*
Determiner →	*the* | *a* | *an* | *this* | *these* | *that*
Preposition →	*from* | *to* | *on* | *near*
Conjunction →	*and* | *or* | *but*

The lexicon for $\mathscr{L}_0$.

simple language $\mathcal{L}_0$
for flight booking



[$_S$ [$_{NP}$ [$_{Pro}$ I]] [$_{VP}$ [$_V$ prefer] [$_{NP}$ [$_{Det}$ a] [$_{Nom}$ [$_N$ morning] [$_{Nom}$ [$_N$ flight]]]]]]

Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

The grammar for $\mathscr{L}_0$, with example phrases for each rule.

CF Grammars – Formal Definitions

- Context Free (Phrase Structure) Grammar is 4-tuple of

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$,
	where A is a non-terminal,
	β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$
S	a designated **start symbol** and a member of N

here: terminal symbols are words

Notation:

Capital letters like A, B, and S	Non-terminals
S	The start symbol
Lower-case Greek letters like α, β, and γ	Strings drawn from $(\Sigma \cup N)*$
Lower-case Roman letters like u, v, and w	Strings of terminals

CF Grammars – Formal Definitions

- if $A \rightarrow \beta$ is a production of R and α and γ are any strings in the set $(\Sigma \cup N)*$, then we say that $\alpha A \gamma$ **directly derives** $\alpha\beta\gamma$, or $\alpha A \gamma \Rightarrow \alpha\beta\gamma$.

- Let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be strings in $(\Sigma \cup N)*, m \geq 1$, such that

$$\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{m-1} \Rightarrow \alpha_m$$

  We say that α_1 **derives** α_m, or $\alpha_1 \overset{*}{\Rightarrow} \alpha_m$.

- the language that a CF grammar generates:

  $\mathscr{L}_G = \{w | w \text{ is in } \Sigma* \text{ and } S \overset{*}{\Rightarrow} w\}$

- Syntactic Parsing: given sequence ("string") of words & CF grammar: construct parse tree

Examples

- declarative sentences:

$$S \rightarrow NP \ VP \qquad \text{I prefer a morning flight}$$

- imperative sentences:

$$S \rightarrow VP \qquad \text{Show the lowest fare}$$

- yes-no-questions:

$$S \rightarrow Aux \ NP \ VP \qquad \text{Do any of these flights have stops?}$$

Examples

- Wh-subject-questions:

$$S \rightarrow \textit{Wh-NP}\ \textit{VP}$$

What airlines fly from Burbank to Denver?

- Wh-non-subject-questions:

$$S \rightarrow \textit{Wh-NP}\ \textit{Aux}\ \textit{NP}\ \textit{VP}$$

What flights do you have from Burbank to Tacoma Washington?

  - Wh-NP *What flights* is not subject of sentence
  - Wh-NP has a long-range dependency to VP *have* that it relates to:
    - represent as semantic relation
    - represent as syntactic relation

The Noun Phrase

Noun phrase: Proper Nouns, pronouns, $NP \rightarrow Det\ \ Nominal$ constructions

- Determiners:

lexical determiners:

a stop	the flights	this flight
those flights	any flights	some flights

$Det \rightarrow NP\ 's$:

United's flight
United's pilot's union
Denver's mayor's mother's canceled flight

mass nouns / nouns in plural: should be / can be without determiner:

Does this flight serve dinner?
Show me *flights* from San Francisco to Denver on weekdays

The Noun Phrase

- Nominal:
  either simple (head-)noun:

  $Nominal \rightarrow Noun$

- or with pre-head noun modifiers:
  cardinal numbers, ordinal numbers, quantifiers, and adjectives or adjective phrases

  two friends one stop

  the first one the next day the second leg
  the last flight the other American flight

  many fares

  a *first-class* fare a *non-stop* flight
  the *longest* layover the *earliest* lunch flight

  the *least expensive* fare

The Noun Phrase

- or with post-head noun modifiers (postmodifiers); three common kinds:

prepositional phrases	all flights *from Cleveland*
non-finite clauses	any flights *arriving after eleven a.m.*
relative clauses	a flight *that serves breakfast*

  - prepositional phrase (PP) postmodifiers

$$Nominal \rightarrow Nominal\ PP$$

all flights *[from Cleveland] [to Newark]*
arrival *[in San Jose] [before seven p.m.]*
a reservation *[on flight six oh six] [from Tampa] [to Montreal]*

  - non-finite postmodifiers
    - gerundive (-ing) form
    - -ed form
    - infinitive form

The Noun Phrase

- gerundive (-ing) form postmodifiers

any of those *[leaving on Thursday]*
any flights *[arriving after eleven a.m.]*
flights *[arriving within thirty minutes of each other]*

$$
\begin{aligned}
\textit{Nominal} &\rightarrow \textit{Nominal GerundVP} \\
\textit{GerundVP} &\rightarrow \textit{GerundV NP} \\
&\quad | \ \textit{GerundV PP} \mid \textit{GerundV} \mid \textit{GerundV NP PP} \\
\textit{GerundV} &\rightarrow \textit{being} \mid \textit{arriving} \mid \textit{leaving} \mid \ldots
\end{aligned}
$$

- -ed form or infinitive form postmodifiers:

the last flight *to arrive in Boston*
I need to have dinner *served*
Which is the aircraft *used by this flight*?

The Noun Phrase

- postnominal relative clause (restrictive relative clause):

  a flight *that serves breakfast*
  flights *that leave in the morning*
  the one *that leaves at ten thirty five*

$$\begin{aligned} \textit{Nominal} &\rightarrow \textit{Nominal RelClause} \\ \textit{RelClause} &\rightarrow (\textit{who} \mid \textit{that})\ \textit{VP} \end{aligned}$$

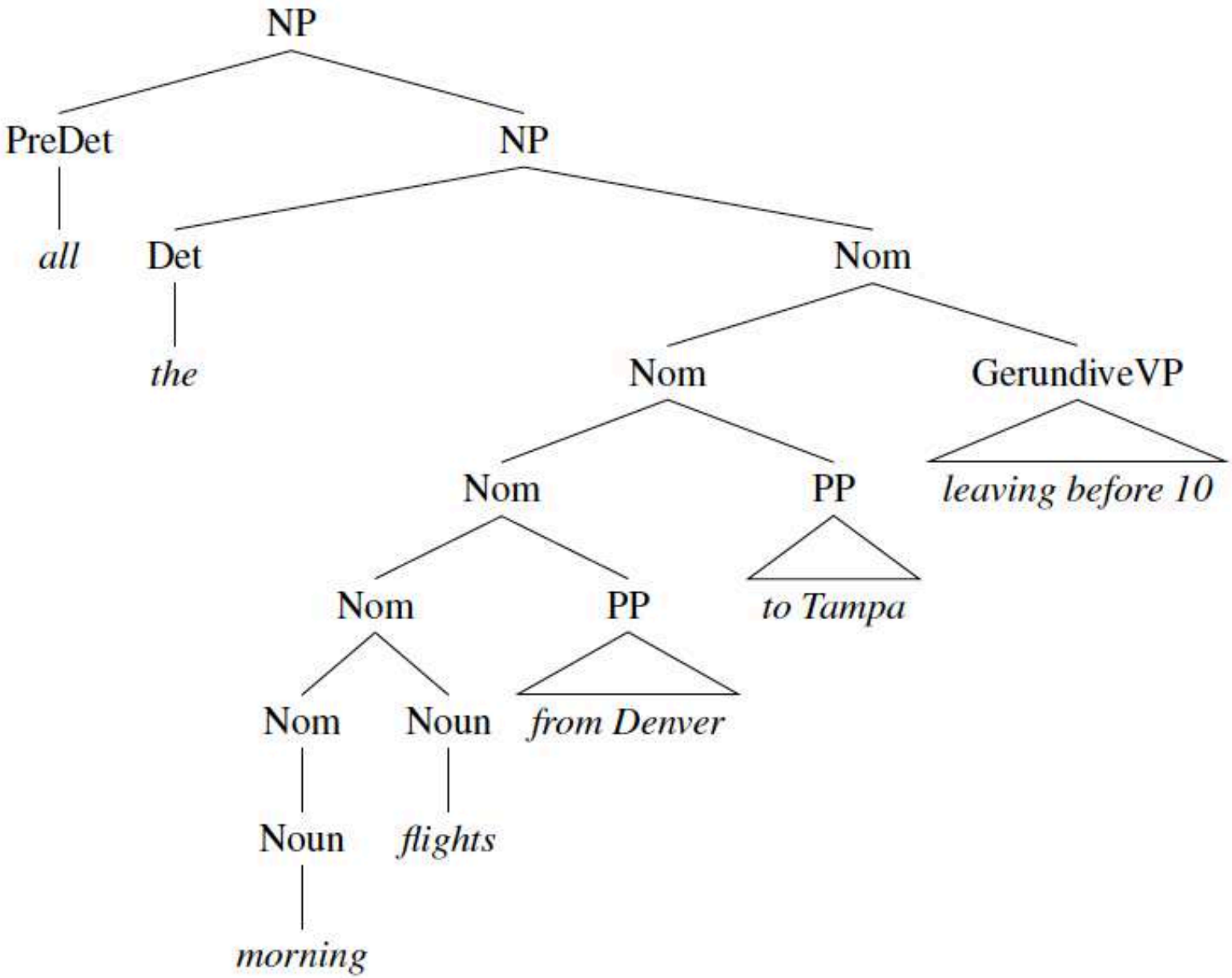
postnominal modifiers can also be combined:

a flight *[from Phoenix to Detroit] [leaving Monday evening]*
evening flights *[from Nashville to Houston] [that serve dinner]*
a friend *[living in Denver] [that would like to visit me here in Washington DC]*

- also possible: predeterminers before the whole noun phrase:

  all the flights all flights all non-stop flights

The Noun Phrase – Larger Example



The Verb Phrase

- example: simple rules

  $VP \rightarrow Verb$ disappear

  $VP \rightarrow Verb\ NP$ prefer a morning flight

  $VP \rightarrow Verb\ NP\ PP$ leave Boston in the morning

  $VP \rightarrow Verb\ PP$ leaving on Thursday

- example: also possible: sentences as complements for verbs (sentential complements)

  $VP \rightarrow Verb\ S$

  You $[_{VP}\ [_{V}$ said $[_{S}$ you had a two hundred sixty six dollar fare]]

  $[_{VP}\ [_{V}$ Tell] $[_{NP}$ me] $[_{S}$ how to get from the airport in Philadelphia to downtown]]

  I $[_{VP}\ [_{V}$ think $[_{S}$ I would like to take the nine thirty flight]]

- example: also possible: VPs as complements for verbs

  I want $[_{VP}$ to fly from Milwaukee to Orlando]

  Hi, I want $[_{VP}$ to arrange three flights]

The Verb Phrase

- idea subcategorize verbs:
  - old: transitive verbs, intransitive verbs
  - modern: over 100 subcategorization frames (complements):

Frame	Verb	Example
$\emptyset$	eat, sleep	I ate
NP	prefer, find, leave	Find [$_{NP}$ the flight from Pittsburgh to Boston]
NP NP	show, give	Show [$_{NP}$ me] [$_{NP}$ airlines with flights from Pittsburgh]
PP_{from} PP_{to}	fly, travel	I would like to fly [$_{PP}$ from Boston] [$_{PP}$ to Philadelphia]
NP PP_{with}	help, load	Can you help [$_{NP}$ me] [$_{PP}$ with a flight]
VPto	prefer, want, need	I would prefer [$_{VPto}$ to go by United airlines]
VPbrst	can, would, might	I can [$_{VPbrst}$ go from Boston]
S	mean	Does this mean [$_{S}$ AA has a hub in Boston]

not every verb supports all of these → possible:

$$\begin{aligned}
\textit{Verb-with-NP-complement} &\rightarrow \textit{find} \mid \textit{leave} \mid \textit{repeat} \mid \ldots \\
\textit{Verb-with-S-complement} &\rightarrow \textit{think} \mid \textit{believe} \mid \textit{say} \mid \ldots \\
\textit{Verb-with-Inf-VP-complement} &\rightarrow \textit{want} \mid \textit{try} \mid \textit{need} \mid \ldots
\end{aligned}$$

$$\begin{aligned}
VP &\rightarrow \textit{Verb-with-no-complement} && \text{disappear} \\
VP &\rightarrow \textit{Verb-with-NP-comp}\ NP && \text{prefer a morning flight} \\
VP &\rightarrow \textit{Verb-with-S-comp}\ S && \text{said there were two flights}
\end{aligned}$$

Coordinations

- Conjoin major phrase types (noun phrases, verb phrases etc.) with conjunctions (*and, or, but, …*) → coordinate phrases.
  Examples with and:

$NP \rightarrow NP \textit{ and } NP$

Please repeat $[_{NP}$ $[_{NP}$ the flights$]$ *and* $[_{NP}$ the costs$]]$
I need to know $[_{NP}$ $[_{NP}$ the aircraft$]$ *and* $[_{NP}$ the flight number$]]$

$Nominal \rightarrow Nominal \textit{ and } Nominal$

Please repeat the $[_{Nom}$ $[_{Nom}$ flights$]$ *and* $[_{Nom}$ costs$]]$
I need to know the $[_{Nom}$ $[_{Nom}$ aircraft$]$ *and* $[_{Nom}$ flight number$]]$

$VP \rightarrow VP \textit{ and } VP$

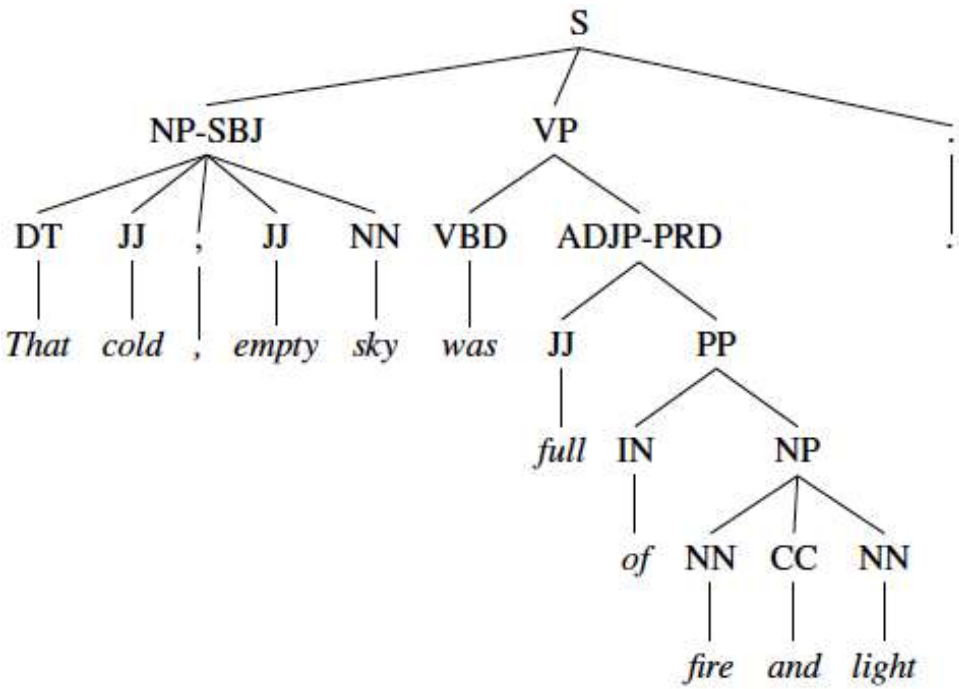
$S \rightarrow S \textit{ and } S$

What flights do you have $[_{VP}$ $[_{VP}$ leaving Denver$]$ *and* $[_{VP}$ arriving in San Francisco$]]$
$[_{S}$ $[_{S}$ I'm interested in a flight from Dallas to Washington$]$ *and* $[_{S}$ I'm also interested in going to Baltimore$]]$

Treebanks

- Corpora with parse trees for each sentence. Example: Penn Treebank

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```



- different treebank or different parts of a treebank may use slightly different notation or non-terminals

Treebanks as Grammars

- From the following two sentences, the following rules may be derived

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```

```
( (S (`` ``)
    (S-TPC-2
      (NP-SBJ-1 (PRP We) )
      (VP (MD would)
        (VP (VB have)
          (S
            (NP-SBJ (-NONE- *-1) )
            (VP (TO to)
              (VP (VB wait)
                (SBAR-TMP (IN until)
                  (S
                    (NP-SBJ (PRP we) )
                    (VP (VBP have)
                      (VP (VBN collected)
                        (PP-CLR (IN on)
                          (NP (DT those)(NNS assets))))))))))))
    (, ,) (`` ``)
    (NP-SBJ (PRP he) )
    (VP (VBD said)
      (S (-NONE- *T*-2) ))
    (. .) ))
```

Grammar	Lexicon
$S \rightarrow NP\ VP\ .$	$PRP \rightarrow we \mid he$
$S \rightarrow NP\ VP$	$DT \rightarrow the \mid that \mid those$
$S \rightarrow$ "S", $NP\ VP$.	$JJ \rightarrow cold \mid empty \mid full$
$S \rightarrow$ -NONE-	$NN \rightarrow sky \mid fire \mid light \mid flight \mid tomorrow$
$NP \rightarrow DT\ NN$	$NNS \rightarrow assets$
$NP \rightarrow DT\ NNS$	$CC \rightarrow and$
$NP \rightarrow NN\ CC\ NN$	$IN \rightarrow of \mid at \mid until \mid on$
$NP \rightarrow CD\ RB$	$CD \rightarrow eleven$
$NP \rightarrow DT\ JJ,\ JJ\ NN$	$RB \rightarrow a.m.$
$NP \rightarrow PRP$	$VB \rightarrow arrive \mid have \mid wait$
$NP \rightarrow$ -NONE-	$VBD \rightarrow was \mid said$
$VP \rightarrow MD\ VP$	$VBP \rightarrow have$
$VP \rightarrow VBD\ ADJP$	$VBN \rightarrow collected$
$VP \rightarrow VBD\ S$	$MD \rightarrow should \mid would$
$VP \rightarrow VBN\ PP$	$TO \rightarrow to$
$VP \rightarrow VB\ S$	
$VP \rightarrow VB\ SBAR$	
$VP \rightarrow VBP\ VP$	
$VP \rightarrow VBN\ PP$	
$VP \rightarrow TO\ VP$	
$SBAR \rightarrow IN\ S$	
$ADJP \rightarrow JJ\ PP$	
$PP \rightarrow IN\ NP$	

Treebanks as Grammars

- whole treebank: 17500 rule types

```
VP → VBD PP
VP → VBD PP PP
VP → VBD PP PP PP
VP → VBD PP PP PP PP
VP → VB ADVP PP
VP → VB PP ADVP
VP → ADVP VB PP
...
```

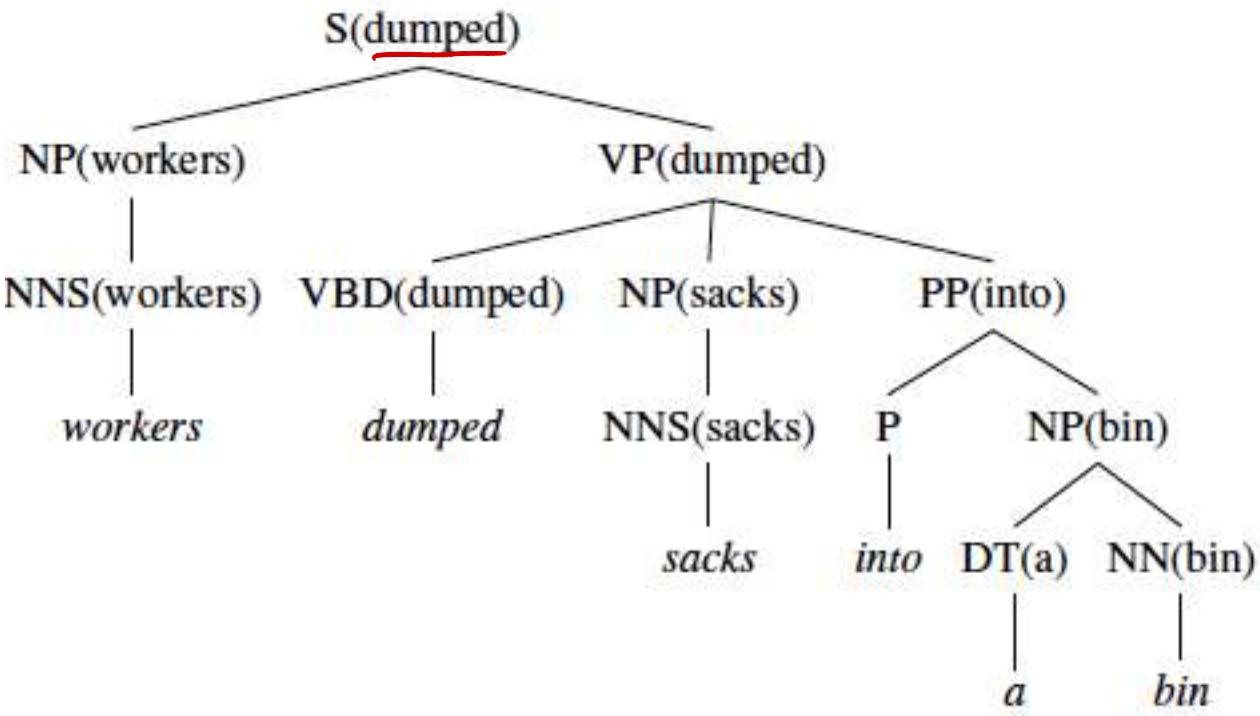
```
NP → DT JJ NN
NP → DT JJ NNS
NP → DT JJ NN NN
NP → DT JJ JJ NN
NP → DT JJ CD NNS
NP → RB DT JJ NN NN
NP → RB DT JJ JJ NNS
NP → DT JJ JJ NNP NNS
NP → DT NNP NNP NNP NNP JJ NN
NP → DT JJ NNP CC JJ JJ NN NNS
NP → RB DT JJS NN NN SBAR
NP → DT VBG JJ NNP NNP CC NNP
NP → DT JJ NNS , NNS CC NN NNS NN
NP → DT JJ JJ VBG NN NNP NNP FW NNP
NP → NP JJ , JJ ‘‘ SBAR ’’ NNS
...
```

we do not want to do this

Heads

- each constituent: associate with a lexical head (word):
  For a NP it is a N, for a VP it is a V

- augment rules and thus parse trees with head words:



Heads

- Finding heads, if heads are not explicitly specified in rules: non-trivial: parse sentence and apply head rules

- example: find head of an NP:

  - If the last word is tagged POS, return last-word.
  - Else search from right to left for the first child which is an NN, NNP, NNPS, NX, POS, or JJR.
  - Else search from left to right for the first child which is an NP.
  - Else search from right to left for the first child which is a $, ADJP, or PRN.
  - Else search from right to left for the first child which is a CD.
  - Else search from right to left for the first child which is a JJ, JJS, RB or QP.
  - Else return the last word

Grammar Equivalence and Normal Form

- two grammars are weakly equivalent if they generate the same language
- two grammars are strongly equivalent if they generate the same language and the same phrase structure (sequence of non-terminals up to renaming)

- a CF grammar is in Chomsky Normal Form (CNF) if it is ϵ-free and every rule is of the form A → BC or A → a. → Non-terminal → terminal
  ↳ non-terminal → non-terminal(s)

$$\left.\begin{array}{l} A \rightarrow B\ C\ D \end{array}\right\} \Longrightarrow \begin{array}{l} A \rightarrow B\ X \\ X \rightarrow C\ D \end{array}$$

$$\left.\begin{array}{l} \mathrm{VP} \rightarrow \mathrm{VBD}\ \mathrm{NP}\ \mathrm{PP} \\ \mathrm{VP} \rightarrow \mathrm{VBD}\ \mathrm{NP}\ \mathrm{PP}\ \mathrm{PP} \\ \mathrm{VP} \rightarrow \mathrm{VBD}\ \mathrm{NP}\ \mathrm{PP}\ \mathrm{PP}\ \mathrm{PP} \\ \mathrm{VP} \rightarrow \mathrm{VBD}\ \mathrm{NP}\ \mathrm{PP}\ \mathrm{PP}\ \mathrm{PP}\ \mathrm{PP} \\ \ldots \end{array}\right\} \Longrightarrow \begin{array}{l} \mathrm{VP} \rightarrow \mathrm{VBD}\ \mathrm{NP}\ \mathrm{PP} \\ \mathrm{VP} \rightarrow \mathrm{VP}\ \mathrm{PP} \end{array}$$

Lexicalized Grammars: Combinatory Categorial Grammars

- CF grammars may have too little an emphasis on the lexicon and may have too many actual non-terminal productions (e.g. compare subcategorization frames)

- → place emphasis on lexicon. Various approaches exist.

- example: Combinatory Categorial Grammar (CCG)

Combinatory Categorial Grammars

- define $\mathscr{C}$, a set of categories for a grammar
  - $\mathscr{A} \subseteq \mathscr{C}$, where $\mathscr{A}$ is a given set of atomic elements (e.g. NP)
  - $(X/Y), (X\backslash Y) \in \mathscr{C}$, if $X, Y \in \mathscr{C}$

- Rules:

X expects Y on the righ

$X/Y \;\; Y \;\Rightarrow\; X$ $\quad$ X/Y : function that seeks its argument to the right

$Y \;\; X\backslash Y \;\Rightarrow\; X$ $\quad$ $X\backslash Y$: function that seeks its argument to the left

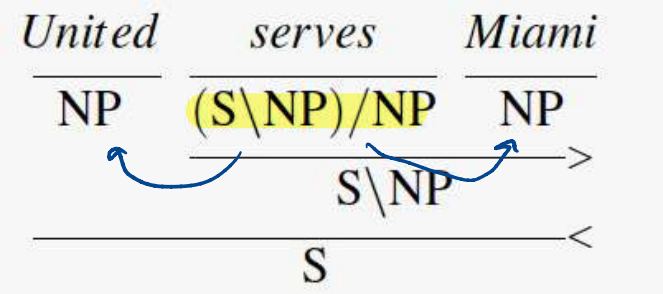
X expects Y on the left

- Lexicon: assignment of words to categories

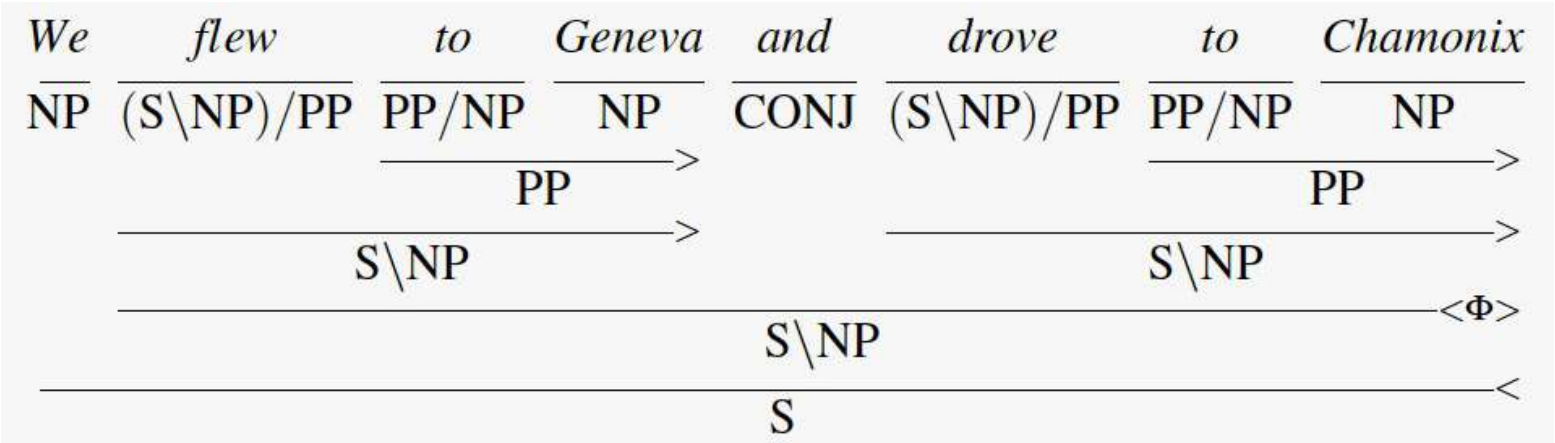
flight :	N	
Miami :	NP	
cancel :	$(S\backslash NP)/NP$	transitive verb (one object)
give :	$((S\backslash NP)/NP)/NP$	bitransitive verb (two objects)

Combinatory Categorial Grammars

- example:



- coordination (meta-)rule : $X \ CONJ \ X \Rightarrow X$ <Φ>



Combinatory Categorial Grammars

- more expressive power than CF by:
  - function composition:

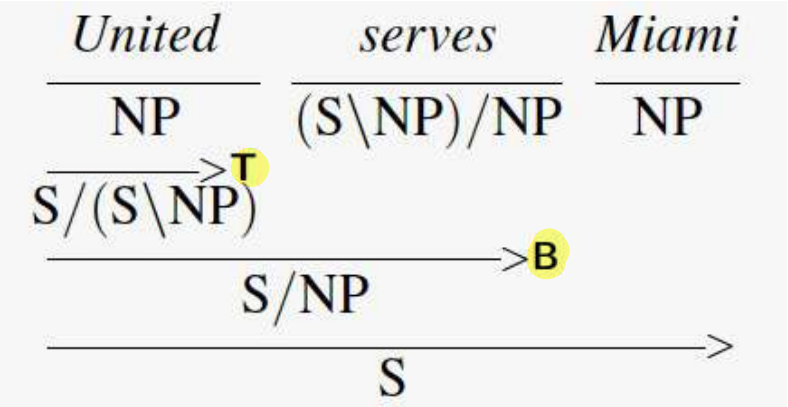
$$X/Y \;\; Y/Z \;\Rightarrow\; X/Z$$
$$Y\backslash Z \;\; X\backslash Y \;\Rightarrow\; X\backslash Z$$

  - type raising:

$$X \;\Rightarrow\; T/(T\backslash X)$$
$$X \;\Rightarrow\; T\backslash(T/X)$$

with T any category

- example for type raising (**T**) and function composition (**B**):
  type-raise NP: $NP \;\Rightarrow\; S/(S\backslash NP)$



→ left to right parsing (human style) is possible

→ *United serves* as S/NP (not normally considered a constituent)

Combinatory Categorial Grammars

- <u>from previous slide:</u> type raising allows to derive constituents not normally considered a constituent *United serves* as S/NP

$$\dfrac{\dfrac{\dfrac{\textit{flew}}{(\mathrm{VP/PP})/\mathrm{NP}} \quad \dfrac{\textit{IcelandAir}}{\mathrm{NP}}}{\mathrm{VP/PP}}\,> \quad \dfrac{\dfrac{\textit{to}}{\mathrm{PP/NP}} \quad \dfrac{\textit{Geneva}}{\mathrm{NP}}}{\mathrm{PP}}\,>}{\mathrm{VP}}\,>$$

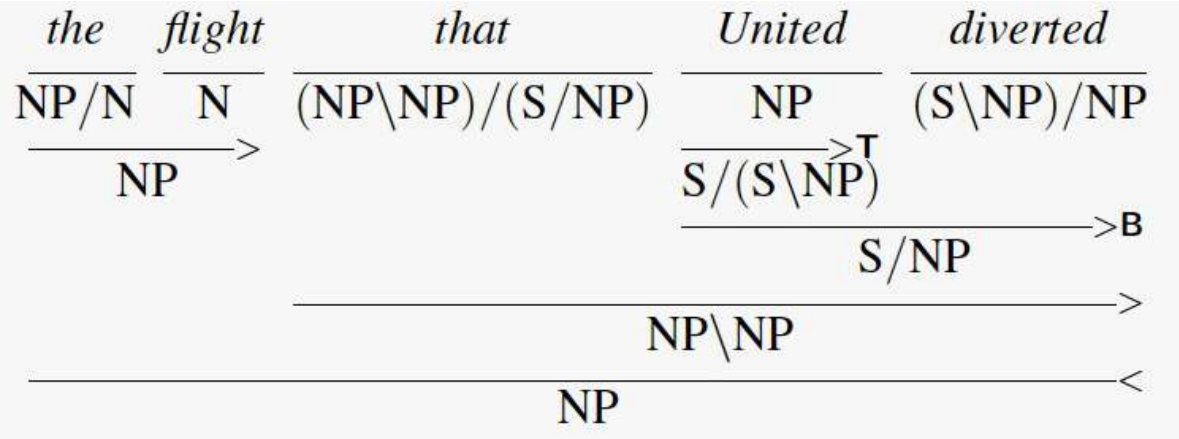
$$\dfrac{\dfrac{\dfrac{\text{flew}}{(VP/PP)/NP} \quad \dfrac{\dfrac{\text{IcelandAir}}{NP}}{(VP/PP)\backslash((VP/PP)/NP)}\,{<}\mathbf{T}}{VP\backslash((VP/PP)/NP)}\,{<}\mathbf{B} \quad \dfrac{\dfrac{\text{to}}{PP/NP} \quad \dfrac{\text{Geneva}}{NP}}{PP}\,{>}}{VP}\,{<}$$

- now possible: coordinate such constituents:

$$\dfrac{\dfrac{\dfrac{\textit{flew}}{(VP/PP)/NP} \quad \dfrac{\dfrac{\textit{IcelandAir}}{NP}}{(VP/PP)\backslash((VP/PP)/NP)}\,{<}\mathbf{T} \quad \dfrac{\dfrac{\textit{to}}{PP/NP} \quad \dfrac{\textit{Geneva}}{NP}}{PP}\,{>}}{VP\backslash((VP/PP)/NP)}\,{<}\mathbf{T} \quad \dfrac{\textit{and}}{CONJ} \quad \dfrac{\dfrac{\textit{SwissAir}}{NP}}{(VP/PP)\backslash((VP/PP)/NP)}\,{<}\mathbf{T} \quad \dfrac{\dfrac{\textit{to}}{PP/NP} \quad \dfrac{\textit{London}}{NP}}{PP}\,{>}}{VP\backslash((VP/PP)/NP)}\,{<}\mathbf{T}}{VP\backslash((VP/PP)/NP)}\,{<}\mathbf{\Phi}{>}}{VP}\,{<}$$

Combinatory Categorial Grammars

- modelling of long range dependencies is possible



(NP\NP) / (S/NP):
seeks a sentence missing an (NP)-argument to its right,
and transforms it into an NP seeking a missing (NP)-element to its left

Syntactic Parsing

- Syntactic parsing: sentence → parse tree

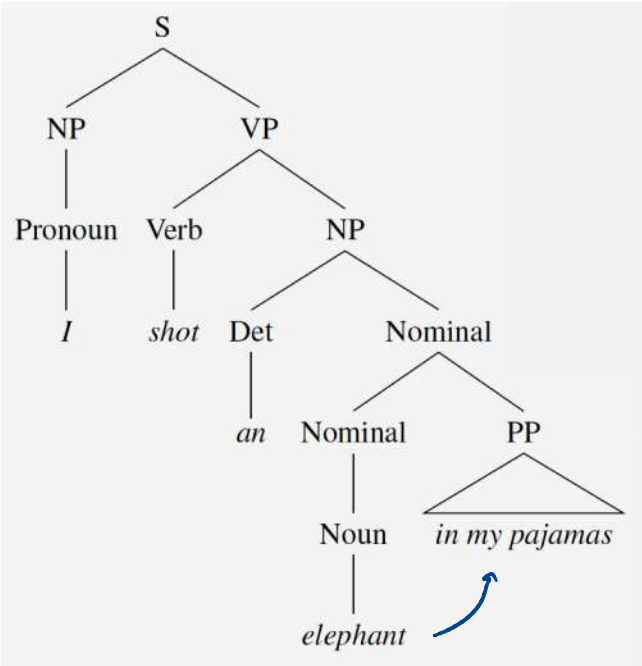
- applications:
  - grammar checking (e.g. in Word)
  - information extraction + question answering
    What books were written by British women authors before 1800?
  - etc

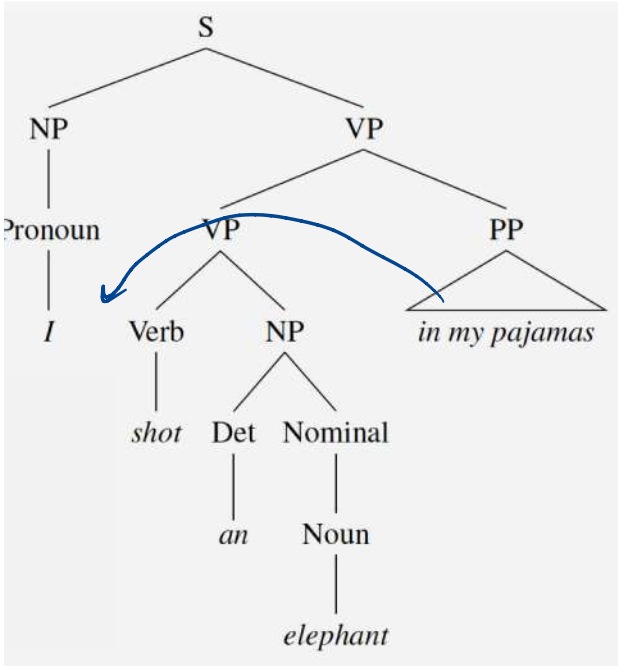
    ↳ systems nowadays do not use this

- classic "conflict":
  - symbolic reasoning (syntactic + semantic parsing + rules, logic) vs
  - sub-symbolic reasoning (deep learning)

Ambiguity

- (structural) Ambiguity: assign more than one parse tree to a sentence
  - attachment ambiguity: a particular constituent can be attached to the parse tree at more than one place





  - coordination ambiguity: join different sets of phrases by *and*

[old [men and women]] [old men] and [women]

Disambiguation

- often some parse trees are semantically or statistically or contextually more probable than others

President Kennedy today pushed aside other White House business to devote all his time and attention to working on the Berlin crisis address he will deliver tomorrow night to the American people over nationwide television and radio

→ integrate statistical info into parsers to produce most plausible parse tree (see old slide-set on probabilistic parsing)

CYK Algorithm

- Cocke Younger Kasami dynamic programming algorithm for parsing CF grammars
  - dynamic programming (as in MinEditDistance, Viterbi, Forward etc. before): systematically fill in tables of solutions to sub-problems, at the end: assemble solution form these
  - two parts: recognizer + actual parser
  - typically: convert to Chomsky Normal Form (CNF) first

CF Grammars: Conversion to CNF

- Convert into Chomsky Normal Form (CNF) : $A \rightarrow B\,C$

  $A \rightarrow w$

1. eliminate terminals from non-pure right hand sides

$$INF\text{-}VP \rightarrow \underline{to}\ VP \quad \Longrightarrow \quad \begin{cases} INF\text{-}VP \rightarrow \underline{TO}\ VP \\ \underline{TO} \rightarrow \underline{to} \end{cases}$$

2. eliminate unit productions $A \rightarrow B$

   if $\underline{A \overset{*}{\Rightarrow} B}$ by a chain of one or more unit productions and $B \rightarrow \gamma$ is a non-unit production in our grammar, then we add $A \rightarrow \gamma$ for each such rule in the grammar and discard all the intervening unit productions

3. iteratively shorten long productions

$$A \rightarrow B\,C\,\gamma \quad \Longrightarrow \quad \begin{cases} A \rightarrow \underline{X1}\ \gamma \\ \underline{X1} \rightarrow \underline{B\,C} \end{cases}$$

CF Grammars: Conversion to CNF

$\mathscr{L}_1$ **Grammar**	$\mathscr{L}_1$ **in CNF**
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

Lexicon
$Det \rightarrow that \mid this \mid the \mid a$
$Noun \rightarrow book \mid flight \mid meal \mid money$
$Verb \rightarrow book \mid include \mid prefer$
$Pronoun \rightarrow I \mid she \mid me$
$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$Aux \rightarrow does$
$Preposition \rightarrow from \mid to \mid on \mid near \mid through$

CYK Recognizer

- organize possible rule applications into upper triangle (i.e. excluding the diagonal) of an (n+1) x (n+1) matrix (n: sentence length)

- first super-diagonal (cells [0,1], [1,2], [2,3], …, [n-1,n]) contain all possible POS of the words of the sentence. (words are the terminals)

- other than first super-diagonal: cell [i,j] contains the set of all non-terminals that represent all possible constituents spanning positions i through j of the sentence (position-notation: *0 Book 1 that 2 flight 3 tomorrow 4*)

- **work** for each cell (**bottom-up** and **left to right**) from first super diagonal to ultimately the right upper corner (cell [0,n]):
  for each current cell [i,j]: fill it with all A from rules A → B C where B ∈ cell [i,k] (to the **left**) and C ∈ cell [k,j] (**below**) for all possible k (i<k<j).
  (for actual parser: also store k, B, C together with all possible multiple occurrences A_n of A in cell [i,j])

function CKY-PARSE(*words*, *grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do** iterate over all columns, left to right

  for all $\{A \mid A \rightarrow words[j] \in grammar\}$ fill element of first superdiagonal with all possible POS A→w for the word

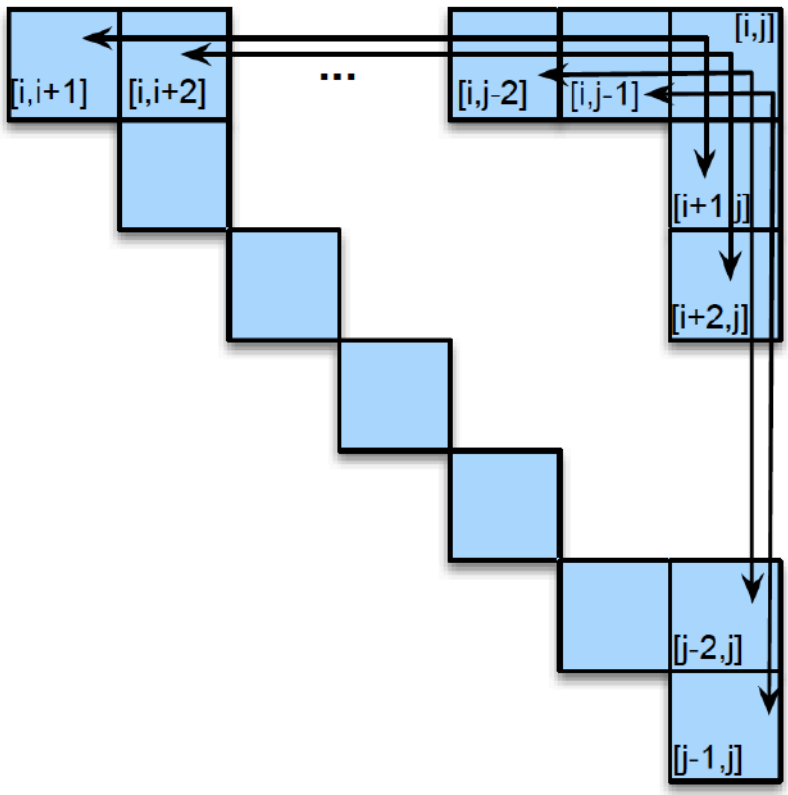
    $table[j-1, j] \leftarrow table[j-1, j] \cup A$

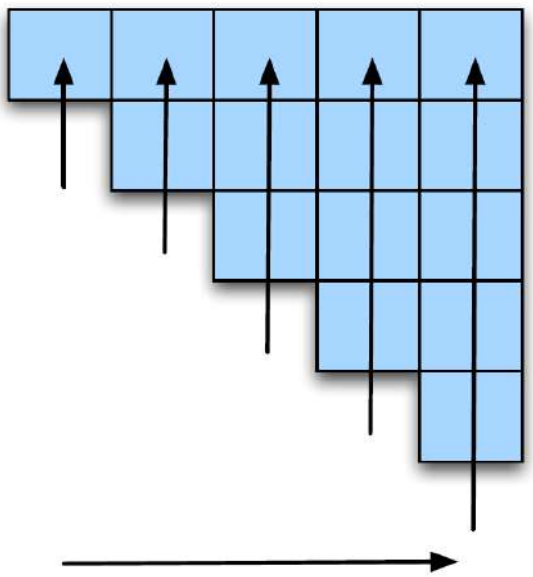
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do** iterate over the rows (the elements in the current column), bottom up

    for $k \leftarrow i+1$ **to** $j-1$ **do** for current (i,j), consider all possible k's to the left & down for non-terminal rule applications

      for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$

        $table[i,j] \leftarrow table[i,j] \cup A$





Filling of the last column bottom up, after all previous columns to the left have already been filled

	Book	the	flight	through	Houston
	S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	[0,5]
		Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
			Nominal, Noun [2,3]	[2,4]	[2,5]
				Prep [3,4]	PP [3,5]
					NP, Proper-Noun [4,5]

diagonal filled with $\mathcal{L}_1$ and Lexicon

	Book	the	flight	through	Houston
	S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	[0,5]
		Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
			Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
				Prep [3,4]	PP [3,5]
					NP, Proper-Noun [4,5]

	Book	the	flight	through	Houston
	S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	[0,5]
		Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
			Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
				Prep [3,4]	PP [3,5]
					NP, Proper-Noun [4,5]

	Book	the	flight	through	Houston
	S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S, VP, X2 [0,3]	[0,4]	S_1, VP, X2 / S_2, VP / S_3
		Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
			Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
				Prep [3,4]	PP [3,5]
					NP, Proper-Noun [4,5]

$\mathcal{L}_1$ in CNF
$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

Lexicon
$Det \rightarrow that \mid this \mid the \mid a$
$Noun \rightarrow book \mid flight \mid meal \mid money$
$Verb \rightarrow book \mid include \mid prefer$
$Pronoun \rightarrow I \mid she \mid me$
$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$Aux \rightarrow does$
$Preposition \rightarrow from \mid to \mid on \mid near \mid through$

CYK Parser

- For every instance S_i of S in cell [0,n]: recursively repeat:
  - trace back all possible rules (productions) S → B C that led to this S_i
  - for each such rule, trace back all instances of B in its corresponding cell and trace back all instances of C in its corresponding cell
  - and so forth

- For every instance S_i of S in cell [0,n] we may get an exponential number of possible parse trees

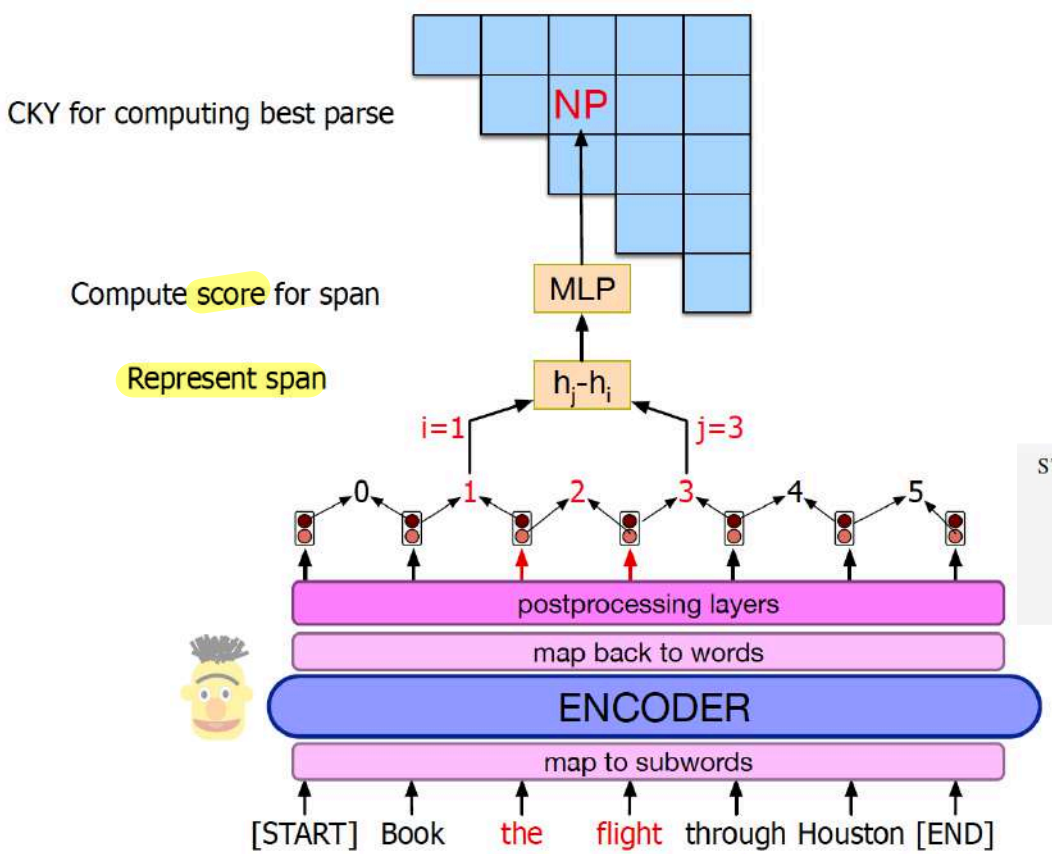
  ↳ incorporate probability
  ↳ beam search

- resulting CNF parse trees may be linguistically unwieldy → convert back into original grammar or user adapted parsing algorithm that can parse non-CNF CF grammars

Span-Based Neural Constituency Parsing

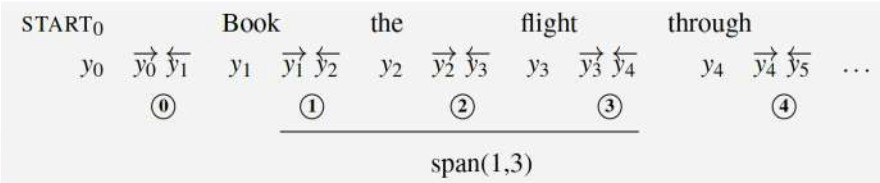
- CYK produces a lot of possible trees → disambiguate? → PCFGs (old slide-set) *or* (span-based) neural approaches ☺

- neural span-based constituency parsing (neural CKY): assign scores to possible trees



score for each labeled constituent span $s(i, j, l)$

$$v(i,j) = [\overrightarrow{y_j} - \overrightarrow{y_i} \,;\, \overleftarrow{y_{j+1}} - \overleftarrow{y_{i+1}}]$$



- In [2] (postprocessing layers are Transformers) : just do a dimensional split:

If $y_2 = \begin{pmatrix} 2 \\ 4 \\ -1 \\ 0 \\ 3 \\ 9 \end{pmatrix}$ *we have* $\overrightarrow{y_2} = \begin{pmatrix} 2 \\ 4 \\ -1 \end{pmatrix}$ *and* $\overleftarrow{y_2} = \begin{pmatrix} 0 \\ 3 \\ 9 \end{pmatrix}$

- In [3] (postprocessing is BiLSTM): for $\overrightarrow{y_i}$ and $\overleftarrow{y_i}$ use BiLSTM hidden states (forward and backward)

Span-Based Neural Constituency Parsing

- integrating labelled span-scores $s(i,j,l)$ into parse tree via:

$$T = \{(i_t, j_t, l_t) : t = 1, \ldots, |T|\}$$

$$s(T) = \sum_{(i,j,l) \in T} s(i,j,l)$$

$$\hat{T} = \operatorname*{argmax}_T s(T)$$

- to actually compute argmax:
  - use greedy: choose best scoring span each (these spans might not correctly „combine“ to a parse tree)
  - or use CYK algorithm idea:
    (no grammar needed
    (grammar is implicitly
    learned by NN via finding
    good intermediate representations)):

$$s_{\text{best}}(i, i+1) = \max_l s(i, i+1, l)$$

$$s_{\text{best}}(i,j) = \max_l s(i,j,l) + \max_k [s_{\text{best}}(i,k) + s_{\text{best}}(k,j)]$$

Performance Evaluation of Classifiers - PARSEVAL

$$\textbf{labeled recall:} = \sum_s \frac{\#\text{ of correct constituents in hypothesis parse of } s}{\#\text{ of correct constituents in reference parse of } s}$$

$$\text{or} \quad \frac{\sum_s \#\text{ of correct constituents in hypothesis parse of } s}{\sum_s \#\text{ of correct constituents in reference parse of } s}$$

$$\textbf{labeled precision:} = \sum_s \frac{\#\text{ of correct constituents in hypothesis parse of } s}{\#\text{ of total constituents in hypothesis parse of } s}$$

$$\text{or} \quad \frac{\sum_s \#\text{ of correct constituents in hypothesis parse of } s}{\sum_s \#\text{ of total constituents in hypothesis parse of } s}$$

$$F_1 = \frac{2PR}{P+R}$$

cross bracketing errors: e.g. predicted: (A(BC)) but correct ((AB)C)

- modern parsers: F1 > 0.9 & < 0.01 cross bracketing errors
- For some applications (and e.g. for lexical parsers such as CCG): more interested in extracting appropriate predicate-argument relations or grammatical dependencies, rather than a specific derivation → better metric of parser usefulness for further semantic processing → use alternative evaluation metrics

Chunking (Partial Parsing)

- Chunking: identifying and classifying flat, non-overlapping segments of a sentence (chunks)

  chunks constitute the basic, non-recursive phrases corresponding to the major content bearing POS (noun phrases, verb phrases, adjective phrases, and prepositional phrases)

  [$_{NP}$ The morning flight] [$_{PP}$ from] [$_{NP}$ Denver] [$_{VP}$ has arrived.]

- criteria for a chunk:

  not nested

  - chunks do not contain other chunks of the same (or other) type
  - boundaries of a chunk: headword + pre-head modifiers;
    ignore post-head modifiers → assignment ambiguities due to overlap
    aka post-head(i-1) ∩ pre-head(i) are vastly reduced

Chunking as Classification Task

- Chunking: sequence classification (sequence labeling) task
- chunking: find limits for a chunk and classify it → BIO notation
  - for each of the n original labels: introduce
    - B version (beginning) and
    - I version (internal);
  - additional label: O (outside)
  - n original labels → 2n+1 new labels

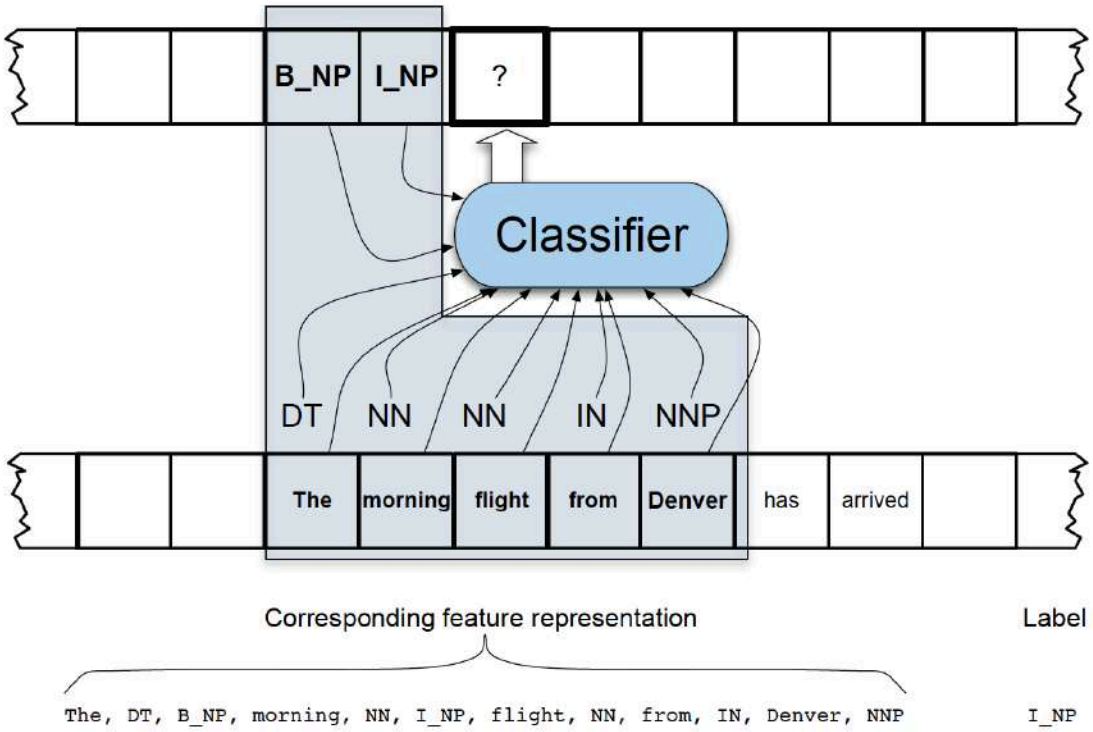
The	*morning*	*flight*	*from*	*Denver*	*has*	*arrived*
B_NP	I_NP	I_NP	B_PP	B_NP	B_VP	I_VP

The	*morning*	*flight*	*from*	*Denver*	*has*	*arrived.*
B_NP	I_NP	I_NP	O	B_NP	O	O

for an NP-only chunker

Chunking as Classification Task

- 2n+1 class labels;
  features per word in the sequence: from frame surrounding the word: e.g.
  - the word itself plus the two preceding words,
  - their parts-of-speech
  - the chunk tags of the preceding inputs in the window
  - ….



Ground Truth for Chunking

- from parse trees in Treebanks:
  - concentrate on the non-terminals corresponding to the chunks that we are interested in (e.g. NP, VP)
  - boundary determination: in the respective phrase: find the head word (with head word detection rules) + all words preceding it; ignore words after head word

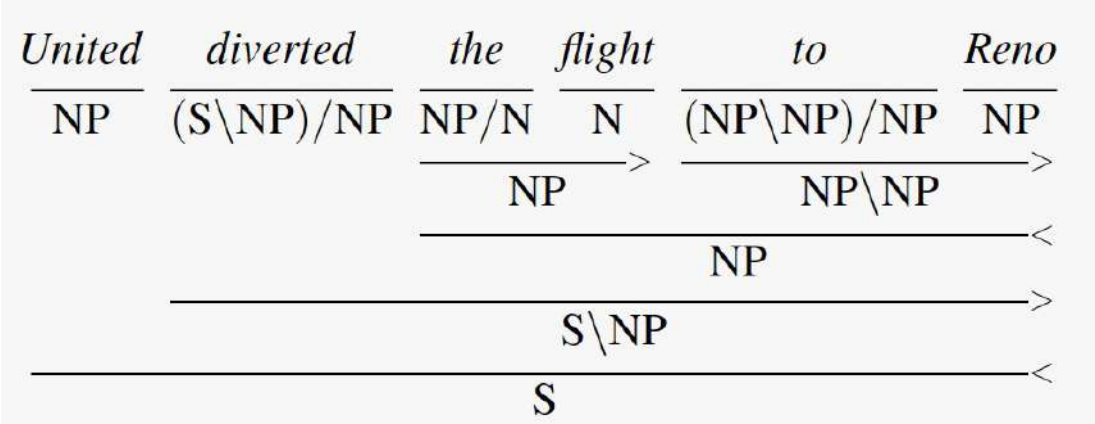
CCG Parsing

- ambiguity example: *United diverted the flight to Reno* :

traditional CFG: choice of internal rule determines parse

$Nominal \rightarrow Nominal\ PP$	United diverted [the flight to Reno]
$VP \rightarrow VP\ PP$	United [diverted to Reno] the flight
$VP \rightarrow Verb\ NP\ PP$	United [diverted the flight to Reno]

traditional CCG (lexicalized grammar): choice of lexical rule determines parse



Repetition: Combinatory Categorial Grammars

- $X/Y \;\; Y \;\Rightarrow\; X$ $\qquad$ X/Y : function that seeks its argument to the right

  $Y \;\; X\backslash Y \;\Rightarrow\; X$ $\qquad$ $X\backslash Y$: function that seeks its argument to the left

- Lexicon:

  $flight: \quad N$

  $Miami: \quad NP$

  $cancel: \quad (S\backslash NP)/NP$

  (transitive verb (one object))

$$\begin{array}{ccc} \textit{United} & \textit{serves} & \textit{Miami} \\ \hline \text{NP} & (\text{S}\backslash\text{NP})/\text{NP} & \text{NP} \\ & \multicolumn{2}{c}{\overrightarrow{\qquad\qquad\qquad}} \\ & \multicolumn{2}{c}{\text{S}\backslash\text{NP}} \\ \multicolumn{3}{c}{\overleftarrow{\qquad\qquad\qquad\qquad\qquad}} \\ \multicolumn{3}{c}{\text{S}} \end{array}$$

- function composition B:

  $X/Y \;\; Y/Z \;\Rightarrow\; X/Z$

  $Y\backslash Z \;\; X\backslash Y \;\Rightarrow\; X\backslash Z$

- type raising T:

  $X \;\Rightarrow\; T/(T\backslash X)$

  $X \;\Rightarrow\; T\backslash(T/X)$

  with T any category

$$\begin{array}{ccc} \textit{United} & \textit{serves} & \textit{Miami} \\ \hline \text{NP} & (\text{S}\backslash\text{NP})/\text{NP} & \text{NP} \\ \overrightarrow{\qquad}^{\text{T}} & & \\ \text{S}/(\text{S}\backslash\text{NP}) & & \\ \multicolumn{2}{c}{\overrightarrow{\qquad\qquad\qquad}^{\text{B}}} & \\ \multicolumn{2}{c}{\text{S}/\text{NP}} & \\ \multicolumn{3}{c}{\overrightarrow{\qquad\qquad\qquad\qquad\qquad}} \\ \multicolumn{3}{c}{\text{S}} \end{array}$$

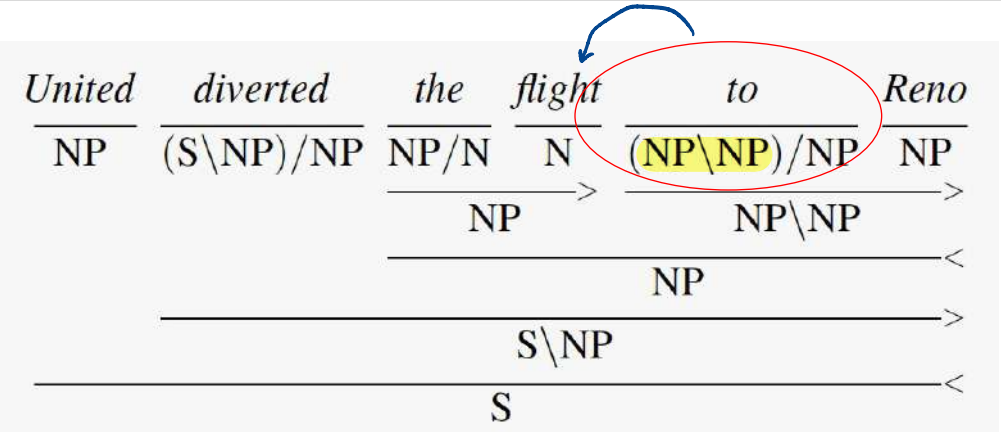
type-raising

CCG Parsing

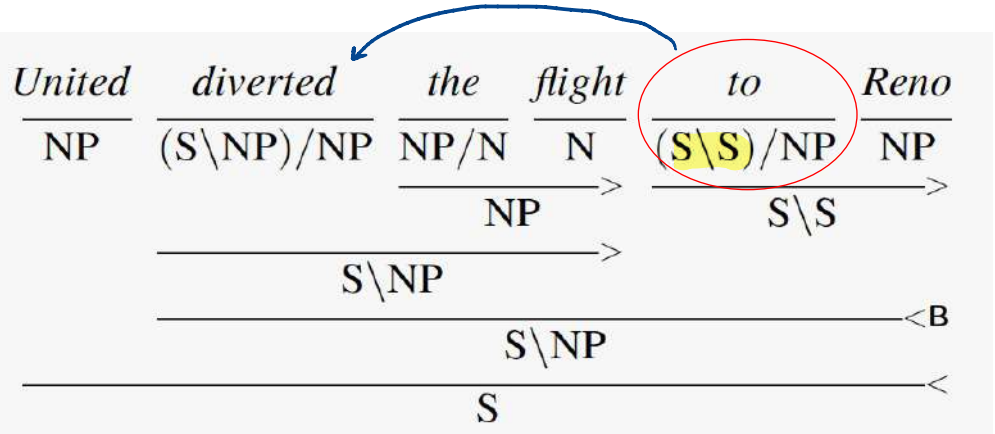
- ambiguity example (ctd.): *United diverted the flight to Reno* :

  traditional CCG (lexicalized grammar): choice of lexical rule determines parse (ctd.)

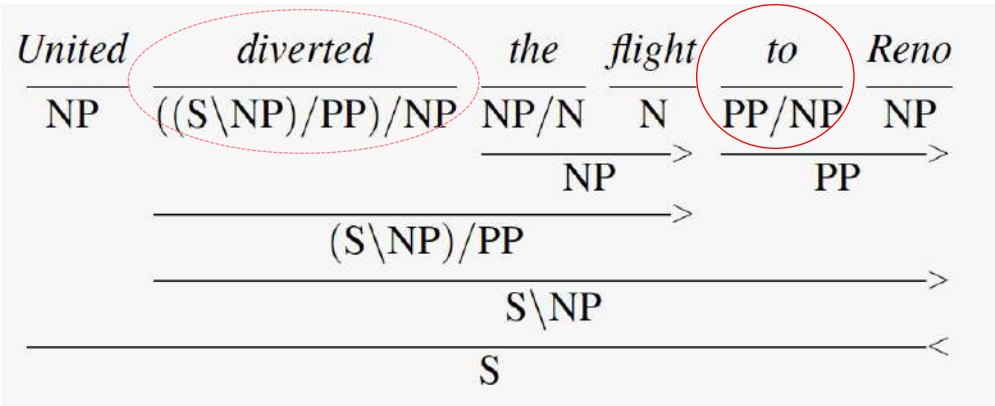
to Reno modifies *flight*



to Reno modifies the verb-phrase: *to* starting a subordinate sentence



divert as di-transitive verb and *to Reno* as simple prepositional phrase argument



Supertagging

- CYK for CCGs: a lot of possible lexical categories (lexical rules) for each word AND very few but very generally applicable internal rules → combinatorial explosion: each cell [i,j] will exist in a lot of variations [i,j,$A^{(n)}$] (most of them useless (zombie constituents)) → CYK does not work!

- → possible remedy: accurately assess and exploit only the most likely lexical categories for each word: Supertagging as a previous step before parsing (→less combinatorial explosion) and / or applying heuristic search algorithms (e.g. A*)

Supertagging

- in CCG bank of CCG parses: over 1000 lexical categories: keep only those that occur more than 10 times → 425 supertags (lexical categories) (compared to 45 POS tags in the Penn Treebank)

- Supertagging (analogous to POS-tagging): sequence labeling problem: approach as before: discriminative CRF model: train model with MLE; for argmax: Viterbi

$$
\begin{aligned}
\hat{T} &= \operatorname*{argmax}_T P(T|W) \\
&= \operatorname*{argmax}_T \prod_i P(t_i | w_{i-l}^{i+l}, t_{i-k}^{i-1}) \\
&= \operatorname*{argmax}_T \prod_i \frac{\exp\left(\sum_i w_i f_i(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}{\sum_{t' \in \text{tagset}} \exp\left(\sum_i w_i f_i(t', w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}
\end{aligned}
$$

e.g. use symmetric window of l=k=2

old notation: $w_i^j == w_{i:j}$

Supertagging

- However: even with additional features f_i : large number of supertags and high per-word ambiguity → too many errors by Viterbi

- Instead of most likely supertag-sequence (by Viterbi) we actually need a probability distribution over possible supertags for each word

United	serves	Denver
N/N: 0.4	$(S\backslash NP)/NP$: 0.8	NP: 0.9
NP: 0.3	N: 0.1	N/N: 0.05
S/S: 0.1	...	...
$S\backslash S$: .05		
...		

→ compute it with a modified version of forward backward algorithm (see chapter Appendix A) or with RNNs (see later)

RNNs for Supertagging

- Recurrent Neural Networks (RNNs) also do well for this task:
  - using vector-valued features for words (word-embeddings)
  - use entire sequence for assessing a word instead of sliding window
  - avoid high-level features (such as POS tags) because of error propagation

CCG Parsing using the A* Algorithm

- A*-algorithm: complete (finds a solution if ex.), optimal (finds best solution), optimally complex (no less complex alg. with same heuristic), informed (using heuristic) search algorithm
  - iteratively choose next state n with optimal estimated cost f(n)= g(n)+h(n), where g(n) current actual cost to n from start node and h(n) non-overestimating(!) estimate to target state from n
  - each state: either in closed list (treated completely, g correct), or open list (known but not yet fully treated, g may shrink), or unknown

CCG Parsing using the A* Algorithm

```
function CCG-ASTAR-PARSE(words) returns table or failure

  supertags ← SUPERTAGGER(words)
  for i ← from 1 to LENGTH(words) do
      for all {A | (words[i], A, score) ∈ supertags}
          edge ← MAKEEDGE(i − 1, i, A, score)
          table ← INSERTEDGE(table, edge)
          agenda ← INSERTEDGE(agenda, edge)
  loop do
      if EMPTY?(agenda) return failure
      current ← POP(agenda)
      if COMPLETEDPARSE?(current) return table
      table ← INSERTEDGE(chart, edge)
      for each rule in APPLICABLERULES(edge) do
          successor ← APPLY(rule, edge)
          if successor not ∈ in agenda or chart
              agenda ← INSERTEDGE(agenda, successor)
          else if successor ∈ agenda with higher cost
              agenda ← REPLACEEDGE(agenda, successor)
```

edges represent spans of symbols (==completed constituents)
$w_{i,j}$ (words from i to j)

agenda ←→ open-list

table: (current) parse table

chart: (current) upper triangle of n+1 x n+1 PCYK matrix

CCG Parsing using the A* Algorithm

how to compute heuristics f(n)?

- simplifying assumption: ignoring usage of other rules, we assume that probability of a derivation D of a sentence $S = (w_1, w_2, \dots)$ = product of probabilities of supertags $T = (t_1, t_2, \dots)$ for words chosen in D

$$\begin{aligned} P(D,S) &= P(T,S) \\ &= \prod_{i=1}^{n} P(t_i|w_i) \end{aligned}$$

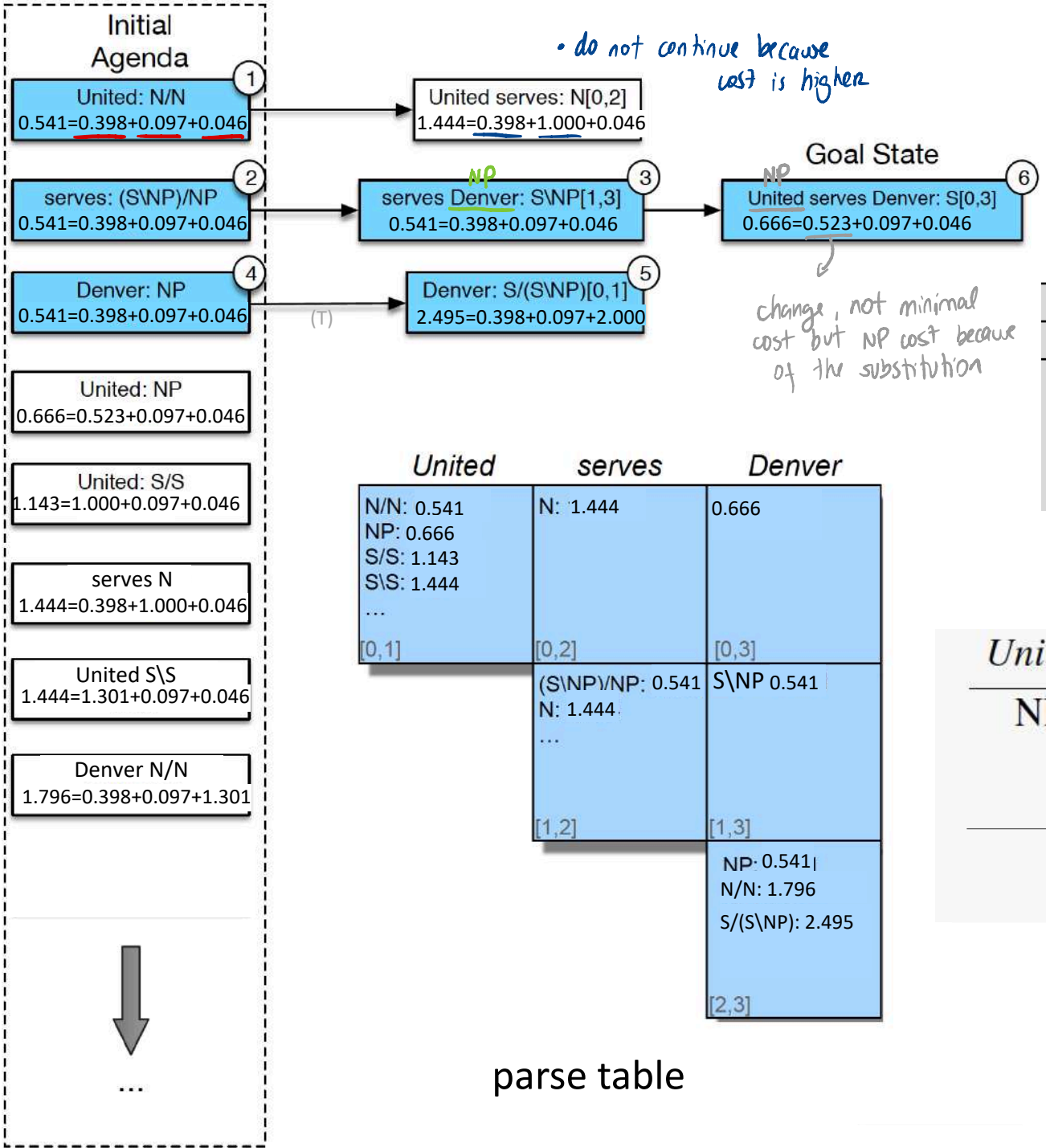
taking negative log → convert probability to a cost:

$$\begin{aligned} P(D,S) &= P(T,S) \\ &= \sum_{i=1}^{n} -\log P(t_i|w_i) \end{aligned}$$

CCG Parsing using the A* Algorithm

- → $f(n) = g(n) + h(n)$ = inside cost + outside cost:
  - g(n) cost of current span (sub-sequence ("sub-string") of words currently considered)
  - h(n) estimated cost for words outside of span: assume that each of these outside words is assigned most probable supertag (→ never overestimate costs)
- →

$$
\begin{aligned}
f(w_{i,j}, t_{i,j}) &= g(w_{i,j}) + h(w_{i,j}) \\
&= \sum_{k=i}^{j} -\log P(t_k|w_k) + \\
&\quad \sum_{k=1}^{i-1} \min_{t \in tags} (-\log P(t|w_k)) + \sum_{k=j+1}^{N} \min_{t \in tags} (-\log P(t|w_k))
\end{aligned}
$$



United	serves	Denver
N/N: 0.4	$(S\backslash NP)/NP$: 0.8	NP: 0.9
NP: 0.3	N: 0.1	N/N: 0.05
S/S: 0.1	...	...
$S\backslash S$: .05		S/(S\NP) 0.01
...		

$-\log_{10} P$		
United	serves	Denver
N/N: 0.398	(S\NP)/NP 0.097	NP: 0.046
NP: 0.523	N: 1.000	N/N: 1.301
S/S: 1.000		S/(S\NP) 2.000
S\S: 1.301		

United	*serves*	*Denver*
N/N: 0.541 NP: 0.666 S/S: 1.143 S\S: 1.444 … [0,1]	N: 1.444 [0,2]	0.666 [0,3]
	(S\NP)/NP: 0.541 N: 1.444 … [1,2]	S\NP 0.541 [1,3]
		NP: 0.541 N/N: 1.796 S/(S\NP): 2.495 [2,3]

parse table

United	*serves*	*Denver*
NP	(S\NP)/NP	NP
	S\NP (>)	
S (<)		

Bibliography

(1) Dan Jurafsky and James Martin: Speech and Language Processing (3rd ed. draft, version Jan 2023); Online: https://web.stanford.edu/~jurafsky/slp3/ (URL, Oct 2023); this slide-set mainly related to chapter 17

(2) Nikita Kitaev and Dan Klein. 2018. Constituency Parsing with a Self-Attentive Encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

(3) Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A Minimal Span-Based Neural Constituency Parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Recommendations for Studying

- minimal approach:

  work with the slides and understand their contents! Think beyond instead of merely memorizing the contents

- standard approach:

  minimal approach + read the corresponding pages in Jurafsky [1]

- interested students

  == standard approach