Finite-State Automata

A finite-state automaton (plural “automata”) consists of:

- A finite set of states $S$
  - Initial state $s_0 \in S$
  - Final states $F \subseteq S$
- A finite alphabet $\Sigma$
- Transitions $\delta : S \times \Sigma \rightarrow 2^S$
  - Special case: deterministic FSA defines $\delta : S \times \Sigma \rightarrow S$

A string $x \in \Sigma^n$ is recognizable by the FSA iff there is a sequence $\langle s_0, \ldots, s_n \rangle$ such that $s_n \in F$ and

$$\bigwedge_{i=1}^{n} [\{s_i \in \delta(s_{i-1}, x_i)\}]$$

This is sometimes called a path.
A regular expression can be:
- an empty string (usually denoted $\epsilon$) or a symbol from $\Sigma$
- a concatenation of regular expressions (e.g., $abc$)
- an alternation of regular expressions (e.g., $ab|cd$)
- a Kleene star of a regular expression (e.g., $(abc)^*$)

A language is a set of strings.

A regular language is a language expressible by a regular expression.

Important theorem: every regular language can be recognized by a FSA, and every FSA’s language is regular.
Proving a Language Isn’t Regular

Pumping lemma (for regular languages): if $L$ is an infinite regular language, then there exist strings $x$, $y$, and $z$, with $y \neq \epsilon$, such that $xy^nz \in L$, for all $n \geq 0$.

If $L$ is infinite and $x$, $y$, $z$ do not exist, then $L$ is not regular.
Proving a Language Isn’t Regular

Pumping lemma (for regular languages): if $L$ is an infinite regular language, then there exist strings $x$, $y$, and $z$, with $y \neq \epsilon$, such that $xy^n z \in L$, for all $n \geq 0$.

If $L$ is infinite and $x$, $y$, $z$ do not exist, then $L$ is not regular.

If $L_1$ and $L_2$ are regular, then $L_1 \cap L_2$ is regular.
Proving a Language Isn’t Regular

Pumping lemma (for regular languages): if \( L \) is an infinite regular language, then there exist strings \( x, y, \) and \( z \), with \( y \neq \epsilon \), such that \( xy^n z \in L \), for all \( n \geq 0 \).

If \( L \) is infinite and \( x, y, z \) do not exist, then \( L \) is not regular.
If \( L_1 \) and \( L_2 \) are regular, then \( L_1 \cap L_2 \) is regular.
If \( L_1 \cap L_2 \) is not regular, and \( L_1 \) is regular, then \( L_2 \) is not regular.
Claim: English is not regular.

\[ L_1 = (\text{the cat}|\text{mouse}|\text{dog})^*(\text{ate}|\text{bit}|\text{chased})^* \text{ likes tuna fish} \]
\[ L_2 = \text{English} \]
\[ L_1 \cap L_2 = (\text{the cat}|\text{mouse}|\text{dog})^n (\text{ate}|\text{bit}|\text{chased})^{n-1} \text{ likes tuna fish} \]

\[ L_1 \cap L_2 \text{ is not regular, but } L_1 \text{ is } \Rightarrow L_2 \text{ is not regular.} \]
the cat likes tuna fish

the cat the dog chased likes tuna fish

the cat the dog the mouse scared chased likes tuna fish

the cat the dog the mouse the elephant squashed scared chased likes tuna fish

the cat the dog the mouse the elephant the flea bit squashed scared chased likes tuna fish

the cat the dog the mouse the elephant the flea the virus infected bit squashed scared chased likes tuna fish
Linguistic Debate
Chomsky put forward an argument like the one we just saw.
Chomsky put forward an argument like the one we just saw.

(Chomsky gets credit for formalizing a hierarchy of types of languages: regular, context-free, context-sensitive, recursively enumerable. This was an important contribution to CS!)
Linguistic Debate

Chomsky put forward an argument like the one we just saw.

(Chomsky gets credit for formalizing a hierarchy of types of languages: regular, context-free, context-sensitive, recursively enumerable. This was an important contribution to CS!)

Some are unconvinced, because after a few center embeddings, the examples become unintelligible.
Chomsky put forward an argument like the one we just saw.

(Chomsky gets credit for formalizing a hierarchy of types of languages: regular, context-free, context-sensitive, recursively enumerable. This was an important contribution to CS!)

Some are unconvinced, because after a few center embeddings, the examples become unintelligible.

Nonetheless, most agree that natural language syntax isn’t well captured by FSAs.
Noun Phrases

What, exactly makes a noun phrase? Examples (Jurafsky and Martin, 2008):

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

More general than noun phrases: **constituents** are groups of words.

Linguists characterize constituents in a number of ways, including:

- where they occur (e.g., "NPs can occur before verbs")
- where they can move in variations of a sentence
- On September 17th, I'd like to fly from Atlanta to Denver
- I'd like to fly on September 17th from Atlanta to Denver
- I'd like to fly from Atlanta to Denver on September 17th
- what parts can move and what parts can't
- *On September I'd like to fly 17th from Atlanta to Denver*
- what they can be conjoined with
- I'd like to fly from Atlanta to Denver on September 17th and in the morning
Constituents

More general than noun phrases: **constituents** are groups of words.

Linguists characterize constituents in a number of ways, including:

- where they occur (e.g., “NPs can occur before verbs”)
- where they can *move* in variations of a sentence
  - On September 17th, I’d like to fly from Atlanta to Denver
  - I’d like to fly on September 17th from Atlanta to Denver
  - I’d like to fly from Atlanta to Denver on September 17th
Constituents

More general than noun phrases: constituents are groups of words.

Linguists characterize constituents in a number of ways, including:

- where they occur (e.g., “NPs can occur before verbs”)
- where they can move in variations of a sentence
  - On September 17th, I’d like to fly from Atlanta to Denver
  - I’d like to fly on September 17th from Atlanta to Denver
  - I’d like to fly from Atlanta to Denver on September 17th
- what parts can move and what parts can’t
  - *On September I’d like to fly 17th from Atlanta to Denver
Constituents

More general than noun phrases: **constituents** are groups of words.

Linguists characterize constituents in a number of ways, including:

- where they occur (e.g., “NPs can occur before verbs”)
- where they can *move* in variations of a sentence
  - On September 17th, I’d like to fly from Atlanta to Denver
  - I’d like to fly on September 17th from Atlanta to Denver
  - I’d like to fly from Atlanta to Denver on September 17th
- what parts can move and what parts can’t
  - *On September I’d like to fly 17th from Atlanta to Denver*
- what they can be conjoined with
  - I’d like to fly from Atlanta to Denver on September 17th and in the morning
Recursion and Constituents

this is the house

this is the house that Jack built

this is the cat that lives in the house that Jack built

this is the dog that chased the cat that lives in the house that Jack built

this is the flea that bit the dog that chased the cat that lives in the house the Jack built

this is the virus that infected the flea that bit the dog that chased the cat that lives in the house that Jack built
Not Constituents
(Pullum, 1991)

- *If on a Winter’s Night a Traveler* (by Italo Calvino)
- *Nuclear and Radiochemistry* (by Gerhart Friedlander et al.)
- *The Fire Next Time* (by James Baldwin)
- *A Tad Overweight, but Violet Eyes to Die For* (by G.B. Trudeau)
- *Sometimes a Great Notion* (by Ken Kesey)
- *[how can we know the] Dancer from the Dance* (by Andrew Holleran)
A context-free grammar consists of:
- A finite set of nonterminal symbols $\mathcal{N}$
  - A start symbol $S \in \mathcal{N}$
- A finite alphabet $\Sigma$, called “terminal” symbols, distinct from $\mathcal{N}$
- Production rule set $\mathcal{R}$, each of the form “$N \rightarrow \alpha$” where
  - The lefthand side $N$ is a nonterminal from $\mathcal{N}$
  - The righthand side $\alpha$ is a sequence of zero or more terminals and/or nonterminals: $\alpha \in (\mathcal{N} \cup \Sigma)^*$
- Special case: Chomsky normal form constrains $\alpha$ to be either a single terminal symbol or two nonterminals
An Example CFG for a Tiny Bit of English

From Jurafsky and Martin (2008)

\[
\begin{align*}
S & \rightarrow NP \ VP \\
S & \rightarrow Aux \ NP \ VP \\
S & \rightarrow VP \\
NP & \rightarrow Pronoun \\
NP & \rightarrow Proper-Noun \\
NP & \rightarrow Det \ Nominal \\
Nominal & \rightarrow Noun \\
Nominal & \rightarrow Nominal Noun \\
Nominal & \rightarrow Nominal PP \\
VP & \rightarrow Verb \\
VP & \rightarrow Verb \ NP \\
VP & \rightarrow Verb \ NP \ PP \\
VP & \rightarrow Verb PP \\
VP & \rightarrow VP PP \\
PP & \rightarrow Preposition \ NP
\end{align*}
\]

\[
\begin{align*}
Det & \rightarrow that \mid this \mid a \\
Noun & \rightarrow book \mid flight \mid meal \mid money \\
Verb & \rightarrow book \mid include \mid prefer \\
Pronoun & \rightarrow I \mid she \mid me \\
Proper-Noun & \rightarrow Houston \mid NWA \\
Aux & \rightarrow does \\
Preposition & \rightarrow from \mid to \mid on \mid near \\
& \rightarrow through
\end{align*}
\]
The phrase-structure tree represents both the syntactic structure of the sentence and the derivation of the sentence under the grammar. E.g., \( VP \rightarrow \text{Verb} \ \text{NP} \).
The First Phrase-Structure Tree
(Chomsky, 1956)

Sentence
  NP
  the man
  VP
  V
  took
  NP
  the book
Where do natural language CFGs come from?

As evidenced by the discussion in Jurafsky and Martin (2008), building a CFG for a natural language by hand is really hard.
Where do natural language CFGs come from?

As evidenced by the discussion in Jurafsky and Martin (2008), building a CFG for a natural language by hand is really hard.

- Need lots of categories to make sure all and only grammatical sentences are included.
Where do natural language CFGs come from?

As evidenced by the discussion in Jurafsky and Martin (2008), building a CFG for a natural language by hand is really hard.

▶ Need lots of categories to make sure all and only grammatical sentences are included.
▶ Categories tend to start exploding combinatorially.
Where do natural language CFGs come from?

As evidenced by the discussion in Jurafsky and Martin (2008), building a CFG for a natural language by hand is really hard.

- Need lots of categories to make sure all and only grammatical sentences are included.
- Categories tend to start exploding combinatorially.
- Alternative grammar formalisms are typically used for manual grammar construction; these are often based on constraints and a powerful algorithmic tool called unification.
Where do natural language CFGs come from?

As evidenced by the discussion in Jurafsky and Martin (2008), building a CFG for a natural language by hand is really hard.

- Need lots of categories to make sure all and only grammatical sentences are included.
- Categories tend to start exploding combinatorially.
- Alternative grammar formalisms are typically used for manual grammar construction; these are often based on constraints and a powerful algorithmic tool called unification.

Standard approach today:

1. Build a corpus of annotated sentences, called a treebank. (Memorable example: the Penn Treebank, Marcus et al., 1993.)
2. Extract rules from the treebank.
3. Optionally, use statistical models to generalize the rules.
Example from the Penn Treebank

[S
  NP-SBJ
    NP
      NNP Pierre
      NNP Vinken
    ADJP
      NP
        CD 61
        NNS years
      JJ old
    MD will
  VP
    VB join
    NP
      DT the
      NN board
    PP-CLR
      IN as
      NP
        DT a
        JJ nonexecutive
        NN director
  NP-TMP
    NNP Nov.
    CD 29
]
LISP Encoding in the Penn Treebank

((S
  (NP-SBJ-1
    (NP (NNP Rudolph) (NNP Agnew) )
    (, ,)
    (UCP
      (ADJP
        (NP (CD 55) (NNS years) )
        (JJ old) )
      (CC and)
      (NP
        (NP (JJ former) (NN chairman) )
        (PP (IN of)
          (NP (NNP Consolidated) (NNP Gold) (NNP Fields) (NNP PLC) ))))))
  (, ,) )
(VP (VBD was)
  (VP (VBN named)
    (S
      (NP-SBJ (-NONE- *-1) )
      (NP-PRD
        (NP (DT a) (JJ nonexecutive) (NN director) )
        (PP (IN of)
          (NP (DT this) (JJ British) (JJ industrial) (NN conglomerate) ))))))
  (, .) ))
Some Penn Treebank Rules with Counts

40717 PP $\rightarrow$ IN NP
33803 S $\rightarrow$ NP-SBJ VP
22513 NP-SBJ $\rightarrow$ -NONE-
21877 NP $\rightarrow$ NP PP
20740 NP $\rightarrow$ DT NN
14153 S $\rightarrow$ NP-SBJ VP .
12922 VP $\rightarrow$ TO VP
11881 PP-LOC $\rightarrow$ IN NP
11467 NP-SBJ $\rightarrow$ PRP
11378 NP $\rightarrow$ -NONE-
11291 NP $\rightarrow$ NN

...  
989 VP $\rightarrow$ VBG S
985 NP-SBJ $\rightarrow$ NN
983 PP-MNR $\rightarrow$ IN NP
983 NP-SBJ $\rightarrow$ DT
969 VP $\rightarrow$ VBN VP

100 VP $\rightarrow$ VBD PP-PRD
100 PRN $\rightarrow$ : NP :
100 NP $\rightarrow$ DT JJS
100 NP-CLR $\rightarrow$ NN
99 NP-SBJ-1 $\rightarrow$ DT NNP
98 VP $\rightarrow$ VBN NP PP-DIR
98 VP $\rightarrow$ VBD PP-TMP
98 PP-TMP $\rightarrow$ VBG NP
97 VP $\rightarrow$ VBD ADVP-TMP VP

...  
10 WHNP-1 $\rightarrow$ WRB JJ
10 VP $\rightarrow$ VP CC VP PP-TMP
10 VP $\rightarrow$ VP CC VP ADVP-MNR
10 VP $\rightarrow$ VBZ S , SBAR-ADV
10 VP $\rightarrow$ VBZ S ADVP-TMP
Penn Treebank Rules: Statistics

32,728 rules in the training section (not including 52,257 lexicon rules)
4,021 rules in the development section
overlap: 3,128
(Phrase-Structure) Recognition and Parsing

Given a CFG \( (\mathcal{N}, S, \Sigma, \mathcal{R}) \) and a sentence \( x \), the recognition problem is:

Is \( x \) in the language of the CFG?

Related problem: parsing:

Show one or more derivations for \( x \), using \( \mathcal{R} \).
Given a CFG \((\mathcal{N}, S, \Sigma, \mathcal{R})\) and a sentence \(x\), the **recognition** problem is:

Is \(x\) in the language of the CFG?

The proof is a derivation.

Related problem: **parsing**:

Show one or more derivations for \(x\), using \(\mathcal{R}\).
(Phrase-Structure) Recognition and Parsing

Given a CFG \((\mathcal{N}, S, \Sigma, \mathcal{R})\) and a sentence \(x\), the **recognition** problem is:

Is \(x\) in the language of the CFG?

The proof is a derivation.

Related problem: **parsing**:

Show one or more derivations for \(x\), using \(\mathcal{R}\).

With reasonable grammars, the number of parses is exponential in \(|x|\).
Ambiguity

S
  NP    VP
    shot   NP
       an  Nominal
          Nominal  PP
            elephant  in my pajamas

S
  NP    VP
    shot   PP
       an  Nominal
          Nominal  in my pajamas
            elephant
Parser Evaluation

Represent a parse tree as a collection of tuples \( \langle \ell_1, i_1, j_1 \rangle, \langle \ell_2, i_2, j_2 \rangle, \ldots, \langle \ell_n, i_n, j_n \rangle \), where

\[\begin{align*}
\bullet & \quad \ell_k \text{ is the nonterminal labeling the } k\text{th phrase} \\
\bullet & \quad i_k \text{ is the index of the first word in the } k\text{th phrase} \\
\bullet & \quad j_k \text{ is the index of the last word in the } k\text{th phrase}
\end{align*}\]

Example:

\[
\rightarrow \langle \langle S, 1, 6 \rangle, \langle NP, 2, 3 \rangle, \langle VP, 4, 6 \rangle, \langle NP, 5, 6 \rangle \rangle
\]

Convert gold-standard tree and system hypothesized tree into this representation, then estimate precision, recall, and \( F_1 \).
Tree Comparison Example

only in left tree

only in right tree

in both trees
Two Views of Parsing

1. Incremental search: the state of the search is the partial structure built so far; each action incrementally extends the tree.

   ▶ Often greedy, with a statistical classifier deciding what action to take in every state.

2. Discrete optimization: define a scoring function and seek the tree with the highest score.

   ▶ Today: scores are defined using the rules.

   \[
   \text{predict}(x) = \arg\max_t \prod_{r \in R} s(r) c_t(r) = \arg\max_t \sum_{r \in R} c_t(r) \log s(r)
   \]

   where \( t \) is constrained to include grammatical trees with \( x \) as their yield. Denote this set \( T_x \).
Two Views of Parsing

1. Incremental search: the state of the search is the partial structure built so far; each action incrementally extends the tree.
Two Views of Parsing

1. Incremental search: the state of the search is the partial structure built so far; each action incrementally extends the tree.
   ▶ Often greedy, with a statistical classifier deciding what action to take in every state.
Two Views of Parsing

1. Incremental search: the state of the search is the partial structure built so far; each action incrementally extends the tree.
   ▶ Often greedy, with a statistical classifier deciding what action to take in every state.

2. Discrete optimization: define a scoring function and seek the tree with the highest score.

\[
predict(x) = \arg\max_t \prod_{r \in R} s(r) c_t(r) = \arg\max_t \sum_{r \in R} c_t(r) \log s(r)
\]

where \( t \) is constrained to include grammatical trees with \( x \) as their yield. Denote this set \( T_x \).
Two Views of Parsing

1. Incremental search: the state of the search is the partial structure built so far; each action incrementally extends the tree.
   ▶ Often greedy, with a statistical classifier deciding what action to take in every state.

2. Discrete optimization: define a scoring function and seek the tree with the highest score.
   ▶ Today: scores are defined using the rules.

\[
predict(x) = \arg\max_t \prod_{r \in \mathcal{R}} s(r)^{c_t(r)} = \arg\max_t \sum_{r \in \mathcal{R}} c_t(r) \log s(r)
\]

where \( t \) is constrained to include grammatical trees with \( x \) as their yield. Denote this set \( \mathcal{T}_x \).
A **probabilistic context-free grammar** consists of:

- A finite set of nonterminal symbols $\mathcal{N}$
  - A start symbol $S \in \mathcal{N}$
- A finite alphabet $\Sigma$, called “terminal” symbols, distinct from $\mathcal{N}$
- Production rule set $\mathcal{R}$, each of the form “$N \rightarrow \alpha$” where
  - The lefthand side $N$ is a nonterminal from $\mathcal{N}$
  - The righthand side $\alpha$ is a sequence of zero or more terminals and/or nonterminals: $\alpha \in (\mathcal{N} \cup \Sigma)^*$
    - Special case: **Chomsky normal form** constrains $\alpha$ to be either a single terminal symbol or two nonterminals
- For each $N \in \mathcal{N}$, a probability distribution over the rules where $N$ is the lefthand side, $p(\ast | N)$. 

Probabilistic Context-Free Grammar
PCFG Example

Write down the start symbol. Here: $S$

Score:

1
Choose a rule from the “S” distribution. Here: $S \rightarrow \text{Aux NP VP}$

Score:

$$p(\text{Aux NP VP} \mid S)$$
PCFG Example

Choose a rule from the “Aux” distribution. Here: $\text{Aux} \rightarrow \text{does}$

Score:

$$p(\text{Aux NP VP} \mid S) \cdot p(\text{does} \mid \text{Aux})$$
Choose a rule from the “NP” distribution. Here: NP → Det Noun

Score:

\[ p(\text{Aux NP VP} \mid S) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \]
Choose a rule from the “Det” distribution. Here: Det → this

Score:

\[ p(\text{Aux NP VP} | \text{S}) \cdot p(\text{does} | \text{Aux}) \cdot p(\text{Det Noun} | \text{NP}) \cdot p(\text{this} | \text{Det}) \]
PCFG Example

Choose a rule from the “Noun” distribution. Here: Noun → flight

Score:

\[ p(\text{Aux NP VP} \mid S) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det}) \cdot p(\text{flight} \mid \text{Noun}) \]
PCFG Example

Choose a rule from the “VP” distribution. Here: \( \text{VP} \rightarrow \text{Verb NP} \)

Score:

\[
p(\text{Aux NP VP} \mid \text{S}) \cdot p(\text{does} \mid \text{Aux}) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det}) \\
\cdot p(\text{flight} \mid \text{Noun}) \cdot p(\text{Verb NP} \mid \text{VP})
\]
PCFG Example

Choose a rule from the “Verb” distribution. Here: Verb → include

Score:

\[
p(Aux \ NP \ VP \mid S) \cdot p(\text{does} \mid Aux) \cdot p(\text{Det Noun} \mid \text{NP}) \cdot p(\text{this} \mid \text{Det}) \\
\cdot p(\text{flight} \mid \text{Noun}) \cdot p(\text{Verb NP} \mid \text{VP}) \cdot p(\text{include} \mid \text{Verb})
\]
PCFG Example

Choose a rule from the “NP” distribution. Here: \( \text{NP} \rightarrow \text{Det Noun} \)

Score:

\[
p(Aux \ \text{NP} \ \text{VP} | \ S) \cdot p(\text{does} | \ Aux) \cdot p(\text{Det Noun} | \ \text{NP}) \cdot p(\text{this} | \ \text{Det}) \\
\cdot p(\text{flight} | \ \text{Noun}) \cdot p(\text{Verb NP} | \ \text{VP}) \cdot p(\text{include} | \ \text{Verb}) \\
\cdot p(\text{Det Noun} | \ \text{NP})
\]
PCFG Example

Choose a rule from the “Det” distribution. Here: Det → a

Score:

\[
p(\text{Aux NP VP | S}) \cdot p(\text{does | Aux}) \cdot p(\text{Det Noun | NP}) \cdot p(\text{this | Det}) \\
\cdot p(\text{flight | Noun}) \cdot p(\text{Verb NP | VP}) \cdot p(\text{include | Verb}) \\
\cdot p(\text{Det Noun | NP}) \cdot p(\text{a | Det})
\]
PCFG Example

Choose a rule from the “Noun” distribution. Here: Noun → meal

Score:

\[
p(Aux \ NP \ VP \mid S) \cdot p(\text{does} \mid Aux) \cdot p(\text{Det Noun} \mid NP) \cdot p(\text{this} \mid Det) \cdot p(\text{flight} \mid Noun) \cdot p(\text{Verb NP} \mid VP) \cdot p(\text{include} \mid \text{Verb}) \cdot p(\text{Det Noun} \mid NP) \cdot p(a \mid Det) \cdot p(\text{meal} \mid \text{Noun})
\]
PCFG as a Noisy Channel

The PCFG defines the source model.

The channel is deterministic: it erases everything except the tree’s leaves (the yield).

Decoding:

$$\arg\max_{t} p(t) \cdot \begin{cases} 1 & \text{if } t \in \mathcal{T}_x \\ 0 & \text{otherwise} \end{cases} = \arg\max_{t \in \mathcal{T}_x} p(t)$$
Probabilistic Parsing with CFGs

- How to set the probabilities $p(\text{righthand side} \mid \text{lefthand side})$?
- How to decode/parse?
Probabilistic CKY
(Cocke and Schwartz, 1970; Kasami, 1965; Younger, 1967)

Input:
- a PCFG \((\mathcal{N}, S, \Sigma, \mathcal{R}, p(\ast | \ast))\), in Chomsky normal form
- a sentence \(x\) (let \(n\) be its length)

Output: \(\arg\max_{t \in \mathcal{T}_x} p(t \mid x)\) (if \(x\) is in the language of the grammar)
Probabilistic CKY

Base case: for $i \in \{1, \ldots, n\}$ and for each $N \in \mathcal{N}$:

$$s_{i:i}(N) = p(x_i \mid N)$$

For each $i, k$ such that $1 \leq i < k \leq n$ and each $N \in \mathcal{N}$:

$$s_{i:k}(N) = \max_{L, R \in \mathcal{N}, j \in \{i, \ldots, k-1\}} p(L, R \mid N) \cdot s_{i:j}(L) \cdot s_{(j+1):k}(R)$$

Solution:

$$s_{1:n}(S) = \max_{t \in \mathcal{T}_x} p(t)$$
Parse Chart

\[
\begin{array}{cccccc}
\hline
& & & & & \\
& x_1 & & & & \\
& & x_2 & & & \\
& & & x_3 & & \\
& & & & x_4 & \\
& & & & & x_5 \\
\hline
\end{array}
\]
Parse Chart

\[
\begin{array}{ccc}
 & s_{1:1}(&) & \\
\times_1 & & s_{2:2}(*)& \\
\times_2 & s_{3:3}(*)& \\
\times_3 & & s_{4:4}(&) \\
\times_4 & s_{5:5}(&) \\
\end{array}
\]
### Parse Chart

<table>
<thead>
<tr>
<th></th>
<th>$s_{1:1}($</th>
<th>$s_{1:2}($</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_{2:2}($</td>
<td></td>
<td></td>
<td>$s_{2:3}($</td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_{3:3}($</td>
<td></td>
<td></td>
<td>$s_{3:4}($</td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_{4:4}($</td>
<td></td>
<td></td>
<td>$s_{4:5}($</td>
<td></td>
</tr>
<tr>
<td>$x_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_{5:5}($</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Parse Chart

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_{1:1}$ (*)</td>
<td>$s_{1:2}$ (*)</td>
<td>$s_{1:3}$ (*)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td>$s_{2:2}$ (*)</td>
<td>$s_{2:3}$ (*)</td>
<td>$s_{2:4}$ (*)</td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td>$s_{3:3}$ (*)</td>
<td>$s_{3:4}$ (*)</td>
<td>$s_{3:5}$ (*)</td>
<td></td>
</tr>
<tr>
<td>$x_4$</td>
<td>$s_{4:4}$ (*)</td>
<td>$s_{4:5}$ (*)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Parse Chart

\[
\begin{array}{cccc}
  & s_{1:1}(\ast) & s_{1:2}(\ast) & s_{1:3}(\ast) & s_{1:4}(\ast) \\
\hline
x_1 & \ & s_{2:2}(\ast) & s_{2:3}(\ast) & s_{2:4}(\ast) & s_{2:5}(\ast) \\
\hline
x_2 & \ & \ & s_{3:3}(\ast) & s_{3:4}(\ast) & s_{3:5}(\ast) \\
\hline
x_3 & \ & \ & \ & s_{4:4}(\ast) & s_{4:5}(\ast) \\
\hline
x_4 & \ & \ & \ & \ & s_{5:5}(\ast) \\
\hline
x_5 & \ & \ & \ & \ & \\
\end{array}
\]
## Parse Chart

<table>
<thead>
<tr>
<th></th>
<th>$s_{1:1}(\ast)$</th>
<th>$s_{1:2}(\ast)$</th>
<th>$s_{1:3}(\ast)$</th>
<th>$s_{1:4}(\ast)$</th>
<th>$s_{1:5}(\ast)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td></td>
<td>$s_{2:2}(\ast)$</td>
<td>$s_{2:3}(\ast)$</td>
<td>$s_{2:4}(\ast)$</td>
<td>$s_{2:5}(\ast)$</td>
</tr>
<tr>
<td>$x_2$</td>
<td></td>
<td></td>
<td>$s_{3:3}(\ast)$</td>
<td>$s_{3:4}(\ast)$</td>
<td>$s_{3:5}(\ast)$</td>
</tr>
<tr>
<td>$x_3$</td>
<td></td>
<td></td>
<td></td>
<td>$s_{4:4}(\ast)$</td>
<td>$s_{4:5}(\ast)$</td>
</tr>
<tr>
<td>$x_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$s_{5:5}(\ast)$</td>
</tr>
<tr>
<td>$x_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Remarks

- Space and runtime requirements?
Remarks

- Space and runtime requirements? $O(|N|n^2)$ space, $O(|R|n^3)$ runtime.
Remarks

- Space and runtime requirements? $O(|N|n^2)$ space, $O(|R|n^3)$ runtime.
- Recovering the best tree?
Remarks

- Space and runtime requirements? $O(|N|n^2)$ space, $O(|R|n^3)$ runtime.
- Recovering the best tree? Backpointers.
Remarks

- Space and runtime requirements? $O(|N|n^2)$ space, $O(|R|n^3)$ runtime.
- Recovering the best tree? Backpointers.
- Probabilistic **Earley’s** algorithm does not require the grammar to be in Chomsky normal form.
1. Initialize every item’s value in the chart to the “default” (zero).
2. Place all initializing updates onto the agenda.
3. While the agenda is not empty or the goal is not reached:
   ▶ Pop the highest-priority update from the agenda (item $I$ with value $v$)
   ▶ If $I = \text{goal}$, then return $v$.
   ▶ If $v > \text{chart}(I)$:
     ▶ $\text{chart}(I) \leftarrow v$
     ▶ Find all combinations of $I$ with other items in the chart, generating new possible updates; place these on the agenda.

Any priority function will work! But smart ordering will save time.

This idea can also be applied to other algorithms (e.g., Viterbi).


