Search: Breadth-first or Depth-first

Handling ambiguity: Dynamic programmiing vs. parallel

Parameters that define different parsing algorithms:

Flexibility: a good algorithm can be easily modified

Efficiency

Completeness: the algorithm must produce all the results

General: the algorithm must be applicable to any

General requirements for a parsing algorithm:

Parsings: If \( m \in L(G) \), produce the (tree) structure that is

assigned by \( G \) to \( m \).

Parsings: If \( m \in L(G) \), determine whether \( m \in L(G) \).

Recognition: Given a (context-free) grammar \( G \) and a string
before constructing larger ones

The idea: build all constituents up to the i-th position before constructing the i + 1 position; build smaller constituents

if the start symbol S is in the \( \{0\} \cup \) entity of the chart

Consequently, the chart is triangular. A word \( w \) is recognized

chart iff \( A \in \{m \ldots m+1\} \)

\( m \cdot 1\) dimensional matrix of size \( n \times (m+1) \)

To recognize a string of length \( n \), uses a chart: a bi-

in CNF

Bottom-up, chart-based recognition algorithm for grammars

THE CYK ALGORITHM

THE CYK ALGORITHM

0 the I car 2 in 3 the car 5

between the input string's words:

A set of indices \( \{0, 1, \ldots, l\} \) is defined to point to positions

The string to recognize is \( m \cdot 1\)

\( m = n \cdot 1\)

\( n \cdot 1 \)

terminals (or the form \( A \rightarrow a \) where \( a \) is a terminal)

is either of the form \( A \rightarrow B \cdot C \) where \( A, B, C \) are non-

The grammar is given in Chomsky Normal Form; each rule

Assumptions:

An example Grammar

Example Sentences:

\[ \begin{align*}
  p & \rightarrow \text{in} \\
  dp & \rightarrow N \\
  dp & \rightarrow pN \\
  dp & \rightarrow pD \\
  d & \rightarrow \text{the} \\
  d & \rightarrow N \\
  d & \rightarrow dp \\
  d & \rightarrow dp \\
\end{align*} \]
Given a grammar $G = \langle \Sigma, \Gamma, S, \delta \rangle$ and a string $w = n$

\textbf{Parsing schema: CKK} 

<table>
<thead>
<tr>
<th>a set of goal items</th>
<th>a set of deduction rules</th>
<th>a set of axioms</th>
<th>a set of items</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>General context-free grammars (not just CKK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for e-rules</td>
</tr>
<tr>
<td>Parsing in addition to recognition</td>
</tr>
<tr>
<td>Extensions</td>
</tr>
</tbody>
</table>

\textbf{CKK Parsing schema:} deduction example

1. The I cat 2 in 3 the 4 hat 5
   - $p$ 
   - $N$ 
   - $D$ 

\textbf{Inference rules:}

\textbf{Axioms:}

\[ A \rightarrow \alpha \quad \text{when} \quad A \in \Sigma \]

\textbf{Items:}

\[ n \leq \hat{f}(\alpha) \leq 0 \text{ for } A \in \Lambda \text{ and } \alpha \in \Sigma \]

\textbf{Implementation:

- Given a grammar $G = \langle \Sigma, \Gamma, S, \delta \rangle$ and a string $w = n$

- The CKK algorithm

\textbf{CKK Parsing schema:} deduction example

- $0$ 
- $p$ 
- $D$ 
- $N$ 

\textbf{CyK Parsing schema: deduction example

- $0$ 
- $p$ 
- $D$ 
- $N$ 

\textbf{Inference rules:}

\textbf{Goals:}

\[ [0, 5^n] \]

\textbf{Axioms:}

\[ A \rightarrow \alpha \quad \text{when} \quad A \in \Sigma \]

\textbf{Items:}

\[ n \leq \hat{f}(\alpha) \leq 0 \text{ for } A \in \Lambda \text{ and } \alpha \in \Sigma \]

\textbf{Implementation:

- Given a grammar $G = \langle \Sigma, \Gamma, S, \delta \rangle$ and a string $w = n$

- The CKK algorithm

\textbf{CKK Parsing schema:} deduction example

- $0$ 
- $p$ 
- $D$ 
- $N$ 

\textbf{Inference rules:}

\[ [0, 5^n] \]

\textbf{Axioms:}

\[ A \rightarrow \alpha \quad \text{when} \quad A \in \Sigma \]

\textbf{Items:}

\[ n \leq \hat{f}(\alpha) \leq 0 \text{ for } A \in \Lambda \text{ and } \alpha \in \Sigma \]
Input: 0 the 1 cat 2 in 3 the 4 hat 5

Top-down deduction: example

Inference rules:

Goal: $\bar{\cdot}v$

Axioms: $\nothing$

Item form: $\\text{if } \bar{\cdot}a \Rightarrow [f, \bar{\cdot}]$

Parse: top-down schema

Reduce

\[ \bar{\cdot} \leftarrow B \quad \frac{[f, \bar{\cdot} \omega]}{[f, \bar{\cdot} \omega]} \]

\[ [f, \bar{\cdot} \omega] \]

\[ [t + f, \bar{\cdot} t + f \omega] \]

\[ [f, \bar{\cdot} \omega] \]

Shirt

Inference rules:

Goal: $\bar{\cdot}v$

Axioms: $\nothing$

Item form: $\\text{if } \bar{\cdot}a \Rightarrow [f, \bar{\cdot}]$

Parse: bottom-up schema (Shift–Reduce)

Bottom-up deduction: example

Parse: bottom-up schema (Shift–Reduce)

Inference rules:

Goal: $\bar{\cdot}v$

Axioms: $\nothing$

Item form: $\\text{if } \bar{\cdot}a \Rightarrow [f, \bar{\cdot}]$

Parse: bottom-up schema (Shift–Reduce)
An example grammar

Noun ← Nominal Noun
Nominal ← Nominal Noun
Nominal ← Noun Nominal
Prep ← Prep Nominal
TP ← Prep from to on
Prep ← Prep
Prep ← to verb NP
Prep ← verb NP
Prep ← to
Noun ← verb NP
S ← NP
S ← verb NP
S ← NP

Top-down vs. Bottom-up Parsing

If parse(a, b) then accept else reject

If parse(5, 0) then return false
If parse(x, y) then return true
For every rule g ∈ R
If x = g then if parse(1, g + 1) then return parse(x, y)
Top-down vs. Bottom-up Parsing

An example derivation tree

The flight includes a meal

An example derivation tree

does the flight from Houston include a meal

An example derivation tree

A book that the flight
Subtrees

Top-down parsing: repeated generation of

If Parse(s, 0) then accept else reject
Return false
If Parse(s, y) \{ f', b', y \} then return true
For every rule B \in P
If f is a left-corner of B then
else if f = b' \{ f' \} then return Parse(\{ f' \}, f + 1)
If f = b' \{ f' \} + 1 \cdot m = b' \{ f' \} \cdot m then Parse(\{ f' \}, m)

Top-down parsing with bottom-up filtering

Constituents are computed over and over again many steps.
Even when parsing terminates, it might take exponentially

NP \leftarrow NP pp.
Let recursive rules can cause non-termination:

The following problems:
Even with bottom-up filtering, top-down parsing suffers from

Top-down vs. Bottom-up parsing

To reduce "blind" search, add bottom-up filtering.

Top-down vs. Bottom-up parsing

Definition: A word w is a left-corner of a non-terminal B iff
the parser succeeds only if B \leftarrow m \cdot f', f' where f' = B'.

Observation: when trying to parse(\{ f' \}), where f' = B'.
Early's parsing algorithm

Basic concepts:

1. Reduplication of effort
2. E-rules can cause performance degradation
3. When they can never be used, they can be used for parallelism
4. All possible analyses of every substring are generated, even if they are not.

Worst-case complexity: O(n^3)

- E-rules are handled correctly
- Left-recursion is handled correctly
- No reduplication of computation
- Combined top-down predictions with bottom-up scanning
- Dynamic programming: partial results are stored in a chart

Top-down vs. Bottom-up Parsing
Inference rules:

**Parsings: Early deduction**

\[ [f \cdot \cdot B \cdot \cdot  \rightarrow A^* \cdot i'] \]
\[ [f \cdot \cdot \cdot \rightarrow A^* \cdot i'] \]
\[ [f \cdot \cdot \cdot \rightarrow A^* \cdot i'] \]

**Complete**

**Predict**

\[ [f \cdot \cdot \cdot \rightarrow A^* \cdot i'] \]
\[ [f \cdot \cdot \cdot \rightarrow A^* \cdot i'] \]
\[ [f \cdot \cdot \cdot \rightarrow A^* \cdot i'] \]

**Scan**

**Inference rules:**

**Parsings: Early deduction**

\[ [f \cdot \cdot B \cdot \cdot \rightarrow A^* \cdot i'] \]
\[ [f \cdot \cdot B \cdot \cdot \rightarrow A^* \cdot i'] \]
\[ [f \cdot \cdot B \cdot \cdot \rightarrow A^* \cdot i'] \]

**Combination:**

**Complete:** when a complete edge is added to the chart.

**Predict:** when an active edge is added to the chart.

**Scan:** when an input word and a corresponding complete edge

**Earley's parsing algorithm**
Earley's Parsing Algorithm

for k such that edge ∈ C[k,t] do
  for edge ∈ registers(edge) do
    if edge is passive then /* complete */
      execute(edge)
    endif
  endfor
  for edge ∈ registers(edge) do
    if edge is active then /* predict */
      for j ∈ C[j,t] do
        { edge } ∩ { j } = ∅
      endfor
    endif
    if edge ⊈ C[t,t] then /* occurs check */
      execute(edge,t)
    endif
  endfor
if S' ∈ S ∈ C[0,n] then accept else reject

Earley's Parsing Algorithm

for every rule a in S do
  for j = 1 to n do
    execute([j,S',S,j])
  endfor
  parse