Introduction (1)
Top-Down Parsing: Scan and Predict with
\[
\text{Predict: } \begin{bmatrix} [A\alpha, i] \\ [\gamma\alpha, i] \end{bmatrix} A \rightarrow \gamma \in P
\]
Problem: in general highly non-deterministic.
Better if grammar in GNF but still non-deterministic.
Goal: find grammars that allow for deterministic top-down parsing.

Introduction (2)
Idea: Use the next terminal symbol(s) as lookahead to determine which production to predict.
Example: 1 lookahead.
Productions \( A \rightarrow X\beta \) and \( A \rightarrow Y\gamma \) such that \( X \overset{*}{\Rightarrow} b\beta' \) and \( Y \overset{*}{\Rightarrow} c\gamma' \).
Then:
\[
\begin{array}{c|c|c}
\text{stack} & \text{input} & \text{remaining input} \\
\hline
A\Gamma & b \ldots & A\Gamma c \ldots \\
X\beta\Gamma & b \ldots & Y\gamma\Gamma c \ldots \\
\end{array}
\]
Deterministic, if neither \( X \overset{*}{\Rightarrow} c \ldots \) nor \( Y \overset{*}{\Rightarrow} b \ldots \)
**LL(1) grammars (1)**

Intuition: A CFG is LL(1) if it allows for a deterministic top-down parsing with 1 lookahead.

In order to define LL(1), we define First and Follow.

Let $\alpha \in (N \cup T)^*$.

$$\text{First}(\alpha) = \{ a \mid \alpha \Rightarrow a\beta, a \in T, \beta \in (N \cup T)^* \} \cup \{ \varepsilon \mid \alpha \Rightarrow \varepsilon \}$$

Let $A \in N$.

$$\text{Follow}(A) = \{ a \mid S \Rightarrow \alpha A \beta, a \in T, \alpha, \beta \in (N \cup T)^* \} \cup \{ S \Rightarrow \alpha A, \alpha \in (N \cup T)^* \}$$

where $\$ \$ is a new symbol marking the end of the input.

---

**LL(1) grammars (2)**

Examples:

1. $G_1: S \rightarrow ab \mid aSb$
   $$\text{First}(ab) = \text{First}(aSb) = \{ a \}$$
   $$\text{Follow}(S) = \{ b, \$ \}$$
2. $G_2: S \rightarrow aB \mid bA, A \rightarrow a \mid aS \mid bAA, B \rightarrow b \mid bS \mid aBB$
   $$\text{First}(aB) = \{ a \}, \text{First}(bA) = \{ b \}$$
   $$\text{First}(a) = \text{First}(aS) = \{ a \}, \text{First}(bAA) = \{ b \}$$
   $$\text{Follow}(S) = \{ a, b, \$ \}$$
3. $G_3: S \rightarrow aT, T \rightarrow b \mid Sb$
   $$\text{First}(S) = \{ a \}, \text{First}(b) = \{ b \}, \text{First}(Sb) = \{ a \}$$

---

**LL(1) grammars (3)**

A CFG $G$ is a LL(1)-grammar if for all $A \in N$:

Let $A \rightarrow \alpha_1 \ldots \alpha_n$ be all $A$-productions in $G$.

- $\text{First}(\alpha_1), \ldots, \text{First}(\alpha_n)$ are pairwise disjoint, and
- if $\varepsilon \in \text{First}(\alpha_j)$ for some $j \in \{1, \ldots, n\}$, then
  $$\text{Follow}(A) \cap \text{First}(\alpha_i) = \emptyset$$
  for all $1 \leq i \leq n, j \neq i$.

$G_1$ and $G_2$ are not LL(1), $G_3$ is LL(1).

There are CFLs that cannot be generated by a LL(1)-grammar. Example: $\{a^n b^n \mid n \geq 0\} \cup \{a^n d^n b^n \mid n \geq 0\}$

---

**LL(1) grammars (4)**

Transformations that can help to obtain an equivalent LL(1) grammar:

- Elimination of left-recursion.
- Left-factoring: elimination of $A$-productions whose rhs have the same prefix:
  Replace $A \rightarrow \alpha_1 \ldots, A \rightarrow \alpha_n \mid (\alpha \in (N \cup T)^+)$ with
  $A \rightarrow \alpha A', A' \rightarrow \beta_1, \ldots, A' \rightarrow \beta_n$ where $A'$ is a new non-terminal.

Example: Transformation from $G_1$ to $G_3$. 

---

Note: The page numbers and dates are consistent with the extracted content.
Computing First and Follow (1)

1. For all \( X \in N \cup T \): \( \text{First}(X) = \emptyset \).
   - If \( X \in T \), then add \( X \) to \( \text{First}(X) \).
   - If \( X \rightarrow \epsilon \in P \), then add \( \epsilon \) to \( \text{First}(X) \).
2. Do the following repeatedly until the \( \text{First} \)-sets do not change any more:
   - For each production \( X \rightarrow X_1 \ldots X_n \) with \( n \geq 1 \), add \( a \in T \) to \( \text{First}(X) \) if there is an \( i \in [1..n] \) such that
     - \( a \in \text{First}(X_i) \), and
     - \( \epsilon \in \text{First}(X_j) \) for all \( 1 \leq j < i \).
     - If \( \epsilon \in \text{First}(X_j) \) for all \( 1 \leq j < n \), then add \( \epsilon \) to \( \text{First}(X) \).
3. For all \( \alpha \in (N \cup T)^* \): Add a new nonterminal \( X_\alpha \) and a production \( X_\alpha \rightarrow \alpha \) and then compute \( \text{First}(\alpha) = \text{First}(X_\alpha) \).
4. \( \text{First}(\epsilon) = \{ \epsilon \} \).

Computing First and Follow (2)

Computing items \([A, t]\) with \( A \in N \cup T, t \in T \cup \{ \epsilon \} \) such that \([A, t]\) iff \( t \in \text{First}(A) \)

- Terminals: \([X, X] \rightarrow X \in T \)
- \( \epsilon \)-productions: \([A, \epsilon] \rightarrow A \rightarrow \epsilon \in P \)
- Bottom-up propagation:
  - \([B, X], [X_1, \epsilon], \ldots, [X_k, \epsilon] \rightarrow [A, X] \rightarrow X_1 \ldots X_k B \beta \in P, X \neq \epsilon \text{ or } \beta = \epsilon \)

Computing First and Follow (3)

Computing \( \text{Follow} \): Let \$ be a new symbol (the end marker).

1. For every \( A \in N \): \( \text{Follow}(A) = \emptyset \).
2. Add \$ to \( \text{Follow}(S) \).
3. Do the following until the \( \text{Follow} \)-sets do not change any more:
   - For each \( A \rightarrow \alpha B \beta \in P \) with \( \alpha, \beta \in (N \cup T)^*, B \in N \):
     - add \( \text{First}(\beta) \cap T \) to \( \text{Follow}(B) \).
     - if \( \epsilon \in \text{First}(\beta) \), then add \( \text{Follow}(A) \) to \( \text{Follow}(B) \).

(We assume all \( A \in N \) to be reachable.)

Computing First and Follow (4)

Computing items \([A, t]\) with \( A \in N, t \in T \cup \{ \$ \} \) such that \([A, t]\) iff \( t \in \text{Follow}(A) \)

- Axiom: \([S, \$] \)
- Right-to-left propagation:
  - \([B, a] \rightarrow A \rightarrow \alpha B X_1 \ldots X_k C \beta \in P, \{X_1, \ldots, [X_k, \epsilon], [C, a] \in \text{First} \}
- Top-down propagation:
  - \([A, X] \rightarrow A \rightarrow \alpha B X_1 \ldots X_k \in P, \{X_1, \ldots, [X_k, \epsilon] \in \text{First} \}

LL(\(k\)) Parsing 21 November 2012
LL(1) parsing (1)

If a CFG is a LL(1) grammar, then it allows for a deterministic top-down parsing where the next input symbol as lookahead determines the predict step to take.

We construct a parsing table that tells us, depending on
- the topmost stack symbol and
- the next input symbol,
which production we have to predict.

LL(1) parsing (2)

Example: \( G_3: S \rightarrow aT, T \rightarrow b | Sb \)

<table>
<thead>
<tr>
<th></th>
<th>( S )</th>
<th>( T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>( S \rightarrow aT )</td>
<td>( T \rightarrow Sb )</td>
</tr>
<tr>
<td>( b )</td>
<td>( - )</td>
<td>( T \rightarrow b )</td>
</tr>
</tbody>
</table>

LL(1) parsing (3)

Construction of the parsing table \( M \):

For each production \( A \rightarrow \alpha \):
- For every \( a \in T \) with \( a \in \text{First}(\alpha) \): \( M(A, a) = A \rightarrow \alpha \).
- If \( \epsilon \in \text{First}(\alpha) \), then for each \( b \in \text{Follow}(A) \): \( M(A, b) = A \rightarrow \alpha \).

LL(1) parsing (4)

Example: \( S \rightarrow ABC, A \rightarrow aA | \epsilon, B \rightarrow cB | bB | \epsilon, C \rightarrow d \)

Parsing table:

<table>
<thead>
<tr>
<th></th>
<th>( S )</th>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>( S \rightarrow ABC )</td>
<td>( A \rightarrow aA )</td>
<td>( - )</td>
<td>( - )</td>
</tr>
<tr>
<td>( b )</td>
<td>( S \rightarrow ABC )</td>
<td>( A \rightarrow \epsilon )</td>
<td>( B \rightarrow bB )</td>
<td>( - )</td>
</tr>
<tr>
<td>( c )</td>
<td>( S \rightarrow ABC )</td>
<td>( A \rightarrow \epsilon )</td>
<td>( B \rightarrow cB )</td>
<td>( - )</td>
</tr>
<tr>
<td>( d )</td>
<td>( S \rightarrow ABC )</td>
<td>( A \rightarrow \epsilon )</td>
<td>( B \rightarrow \epsilon )</td>
<td>( C \rightarrow d )</td>
</tr>
</tbody>
</table>
**LL(k) parsing**

If more than one symbol as lookahead is used, namely up to \( k \) symbols, the technique is called LL(\( k \)) parsing.

The definitions of First and Follow must be extended to contain terminal strings of up to \( k \) symbols.

The parse table gets much larger of course.

A CFG is LL(\( k \)) if it allows for deterministic top-down parsing with \( k \) lookahead symbols.

---

**Conclusion**

- LL(1) grammars allow for a deterministic top-down parsing.
- The next terminal in the remaining input (the lookahead) determines the predict step to take.
- First and Follow and the parse table can be precompiled.
- The set of languages generated by LL(1) grammars is a proper subset of CFL.