$LL(1)$ Grammars

Recap: Parsing $LL(k)$ Grammars

- We have seen that CFGs can be parsed with a Nondeterministic Top-Down Automaton (NTA) and that a k-lookahead allows to make the parsing deterministic if the grammar is $LL(k)$
- The CFG $G =< \Sigma, N, P, S >$ is an $LL(k)$ grammar for a given $k \in \mathbb{N}$ if for all leftmost derivations of the form

$$S \Rightarrow_l^* wA\alpha \begin{cases} \Rightarrow_l w\beta\alpha \Rightarrow_l^* wx \\ \Rightarrow_l w\gamma\alpha \Rightarrow_l^* wy \end{cases} \text{such that } \beta \neq \gamma$$

it follows that $first_k(x) \neq first_k(y)$

- Lemma: $G =< \Sigma, N, P, S >$ is an $LL(k)$ grammar if and only if for all leftmost derivations of the form

$$S \Rightarrow_l^* wA\alpha \begin{cases} \Rightarrow_l w\beta\alpha \Rightarrow_l^* wx \\ \Rightarrow_l w\gamma\alpha \Rightarrow_l^* wy \end{cases} \text{such that } \beta \neq \gamma$$

it follows that $first_k(\beta\alpha) \cap first_k(\gamma\alpha) = \emptyset$

Parsing $LL(k)$

- Using the property $first_k(\beta\alpha) \cap first_k(\gamma\alpha) = \emptyset$ for deterministic expansion decisions in the NTA requires to compute $first_k(\beta\alpha)$ and $first_k(\gamma\alpha)$ for all $\beta\alpha$ and $\gamma\alpha$ that can appear during the parsing.

- Implementing an efficient parser for $LL(k > 1)$ is not easy
  - See for example the ANTLR parser generator tool http://www.antlr.org/
- Fortunately, for many interesting languages, an $LL(1)$ grammar can be given, and we will see that parsing for $LL(1)$ is rather straight-forward

$first_1$ and $follow_1$ set

- Consider $G =< \Sigma, N, P, S >$, $X = N \cup \Sigma$
- We have already seen the $first_1$ set:
  For every $\alpha \in X^*$ we define

$$first_1(\alpha) = \{a \in \Sigma \mid \exists w \in \Sigma^*: \alpha \Rightarrow^* aw\} \cup \{\varepsilon \mid \alpha \Rightarrow^* \varepsilon\}$$

- We now define the $follow_1$ set. It is the set of all terminal symbols (or ε) that can *follow* a non-terminal symbol $A \in N$:

$$follow_1(A) = \{x \in first_1(\alpha) \mid \exists w \in \Sigma^*, \alpha \in X^*: S \Rightarrow_l^* wA\alpha\}$$

- In the following, we will write fi and fo for $first_1$ and $follow_1$

- We also define the $first_1$ set for a set $\Gamma \subseteq X^*$:

$$fi(\Gamma) = \bigcup_{\gamma \in \Gamma} fi(\gamma)$$

Lookahead sets and $LL(1)$

- Given a rule $A \to \beta \in P$, its *lookahead set* is defined as:

$$la(A \to \beta) = fi\big(\beta \cdot fo(A)\big)$$

  where $\beta \cdot \{a, b, c, \dots\}$ means $\{\beta a, \beta b, \beta c, \dots\}$

- Obviously, $a \in la(A \to \beta)$ if and only if

$$a \in fi(\beta) \text{ or } (\beta \Rightarrow^* \varepsilon \text{ and } a \in fo(A))$$

  and $\varepsilon \in la(A \to \beta)$ if and only if

$$\beta \Rightarrow^* \varepsilon \text{ and } \varepsilon \in fo(A)$$

- **Important Theorem** (not proven here):
  A grammar is $LL(1)$ if and only if for all rules $A \to \beta \mid \gamma$ (with $\beta \neq \gamma$)

$$la(A \to \beta) \cap la(A \to \gamma) = \emptyset$$

  - Note that $la(\cdot)$ can be easily computed for all rules
  - Does not generally hold for $LL(k > 1)$

Computing lookahead set $la(A \to \beta) = fi(\beta \cdot fo(A))$

- $fi(\alpha)$ for $\alpha \in X^*$ is the least set such that
  - $fi(a) = \{a\}$ for $a \in \Sigma$
  - $a \in fi(A)$ for $A \to \beta$ if $a \in fi(\beta)$
  - $a \in fi(Y_1 \ldots Y_n)$ for $Y_i \in X$ if $\varepsilon \in fi(Y_1) \cap \cdots \cap fi(Y_{k-1})$ and $a \in fi(Y_k)$ for some $k \leq n$
  - $\varepsilon \in fi(Y_1 \ldots Y_n)$ for $Y_i \in X$ if $\varepsilon \in fi(Y_1) \cap \cdots \cap fi(Y_n)$
- $fo(A)$ for $A \in N$ is the least set such that
  - $\varepsilon \in fo(A)$ if A is the start symbol of the grammar
  - $a \in fo(A)$ if there is a rule $B \to \alpha A \beta$ and $a \in fi(\beta)$
  - $a \in fo(A)$ if there is a rule $B \to \alpha A \beta$ with $\varepsilon \in fi(\beta), a \in fo(B)$
- Some useful insights:
  - $A \to a\beta \ \Rightarrow \ a \in fi(A)$
  - $A \to B\alpha$ and $a \in fi(B) \ \Rightarrow \ a \in fi(A)$
  - $A \to \varepsilon \ \Rightarrow \ \varepsilon \in fi(A)$
  - $a \in fi(A) \ \Rightarrow \ a \in fi(A\alpha)$

Example

- Is this gramma $LL(1)$?

$$E \to E + T \mid T$$
$$T \to T * F \mid F$$
$$F \to (E) \mid a \mid b$$

- Let's check:
  - From $F \to a$ follows that $a \in fi(F)$
  - Because of $T \to F$ we also have $a \in fi(T)$
  - Because of $T \to T * F$ we also have $a \in fi(T * F)$
- Remember that

$$la(A \to \beta) = fi\big(\beta \cdot fo(A)\big)$$

- Therefore

$$a \in la(T \to F) \ \text{ and } \ a \in la(T \to T * F)$$

- We conclude that the $LL(1)$ property does not hold:

$$la(T \to F) \cap la(T \to T * F) \neq \emptyset$$

and therefore the grammar is not $LL(1)$

  - Because if the parsing automaton is in the state $(aw, T\beta, \ldots)$ it cannot decide whether T should be expanded to F or $T * F$

What was the problem in our example?

- In our example, the problem are rules like this one:
$$T \rightarrow T * F \mid F$$
- Everything that appears in the $first$ set of F will be also in the $first$ set of $T * F$ because $T * F$ can be expanded to $F * F$
- This will also happen if there is an intermediate rule, e.g.,
$$T \rightarrow X \mid F$$
$$X \rightarrow T * F \ldots$$
- It's even worse if we only have the rule
$$T \rightarrow T * F$$

  The automaton would expand in an infinite loop
$$(aw, T\alpha, \ldots) \Rightarrow (aw, T * F\alpha, \ldots) \Rightarrow (aw, T * F * F\alpha, \ldots) \Rightarrow \cdots$$
- In general, CFG is called *left recursive* if there is a $A \in N$ such that $A \Rightarrow^{+} A\alpha$
  - Corollary: if a CFG is left recursive with $A \Rightarrow^{+} A\alpha$ then there exists $\beta \in X^{*}$ with $A \Rightarrow_{l}^{+} A\beta$
- It hold: **a left recursive grammar is not $\boldsymbol{LL(k)}$ for any $\boldsymbol{k}$**

Eliminating Left Recursion

- *Direct left recursion* of the form
$$A \to A\alpha_1 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \beta_n$$
can be replaced by right recursion
$$A \to \beta_1 A' \mid \ldots \mid \beta_n A'$$
$$A' \to \alpha_1 A' \mid \ldots \mid \alpha_m A' \mid \varepsilon$$
without changing the language.
- Example:
$$E \to E + T \mid T$$
$$T \to T * F \mid F$$
$$F \to (E) \mid a \mid b$$
can be transformed to
$$E \to TE'$$
$$E' \to \; +TE' \mid \varepsilon$$
$$T \to FT'$$
$$T' \to * FT' \mid \varepsilon$$
$$F \to (E) \mid a \mid b$$

Eliminating Left Recursion, part 2

- *Indirect left recursion* of the form

$$A \to A_1\alpha_1$$
$$A_1 \to A_2\alpha_2$$
$$\dots$$
$$A_{k-1} \to A_k\alpha_k$$
$$A_k \to A\beta$$

can be removed by

- replacing A_1 in the rules where it appears by $A_2\alpha_2$
- then replacing A_2 in the rules where it appears by $A_3\alpha_3$
- etc.
- and as last step: eliminate the resulting direct recursion $A \to A\gamma$

A much easier problem...

- Obviously, a grammar with rules like
$$A \rightarrow \alpha\beta \mid \alpha\gamma$$
is not $LL(k)$ if the "length" of α is $\geq k$
- But this can be fixed easily by replacing those rules by
$$A \rightarrow \alpha B$$
$$B \rightarrow \beta \mid \gamma$$
- This is called *left factorization*

Note how we avoid the dangling else problem (ambiguous grammar) by using an "endif"

- Example:
$$S \rightarrow if\ C\ then\ S\ else\ S\ endif \mid\ if\ C\ then\ S\ endif$$
can be turned to
$$S \rightarrow if\ C\ then\ S\ S'$$
$$S' \rightarrow else\ S\ endif \mid\ endif$$

Important remarks

- The transformations shown here preserve the language generated by the CFG **but not the syntax tree**!
- Not every language can be generated by an $LL(1)$ grammar. In fact, there are context free languages that cannot be generated by any $LL(k)$ grammar.
  - Example: $\{a^n bc^n \mid n \geq 1\} \cup \{a^n dc^n \mid n \geq 1\}$

    This would require an infinite lookahead to read beyond a^n
- Even worse: for an arbitrary CFG it is undecidable whether there is an $LL(k)$ grammar that generates the same language
- Use tools like http://mdaines.github.io/grammophone/ to check whether a CFG is $LL(1)$. It can also transform to $LL(1)$
- Transformations are also used by parser generator tools like ANTLR https://www.antlr.org/