Overview of Parsing Methods

The FIRST set

- If X is a terminal symbol then FIRST$(X) = \{X\}$.
- If X is an intermediate symbol and $X \to Y_1 Y_2 \cdots Y_k$ is a rule and ϵ is in FIRST$(Y_1) \cdots$ FIRST(Y_k) then ϵ is in FIRST(X). (Note that rule $X \to \epsilon$ is a special case of this case.)
- If X is a nonterminal symbol and $X \to Y_1 Y_2 \cdots Y_k$ is a rule and ϵ is in FIRST$(Y_1) \cdots$ FIRST(Y_i) and a terminal symbol a is in FIRST(Y_{i+1}) then a is in FIRST(X).

The FOLLOW set

- If S is the starting symbol then \$ (end of file) is in FOLLOW(S).
- If $A \to \alpha B \beta$ is a rule then everything but ϵ that is in FIRST(β) is also in FOLLOW(B).
- If $A \to \alpha B \beta$ is a rule and ϵ is in FIRST(β) then everyting in FOLLOW(A) is also in FOLLOW(B).

LL(1) parsing tables

- If $A \to \alpha$ is a rule and b is a terminal symbol in FIRST(α) then $A \to \alpha$ is in $M[A, b]$.
- If $A \to \alpha$ is a rule and ϵ is in FIRST(α) and a terminal symbol b is in FOLLOW(A) then $A \to \alpha$ is in $M[A, b]$.

Loop invariant in an LL(1) parser Before and after each step in a table-driven LL(1) parser the concatenated string $\alpha\beta$, wheer α is the string of the terminal symbols that have been read and β is the string of terminal and nonterminal symbols that are on the prediction stack, is a string that is derivable from the starting symbol with left derivation.

In other words: $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha\beta$ is always true.

The sequence of strings $\alpha\beta$, that one gets with LL(1) parsing of a given string is therefore a left dereivation of that string according to the given grammar.

LR(0) state machines

Starting state.

The starting state is

$$I_0 = \text{closure}(\{[S' \to \cdot S]\})$$

where S is the old starting symbol and S' is the new one.

The closure function.

- closure(I) contains I as a subset.
- If $[A \to \alpha \cdot B\beta]$ is in closure(I) and $B \to \gamma$ is a rule then $[B \to \cdot\gamma]$ is in closure(I).

The goto function.

- goto$(I, X) =$ closure(goto(I, X))
- If $[A \to \alpha \cdot X\beta]$ is in I then $[A \to \alpha X \cdot \beta]$ is in goto(I, X).

SLR(1) parsing table

- If a is a terminal symbol and s is a state number and goto$(I_s, a) = I_k$ is not empty then *shift* k is in $M[s, a]$.
- If s is a state number and $[A \to \alpha\cdot]$ is in I_s and a is in FOLLOW(A) then *reduce* $A \to \alpha$ is in $M[s, a]$.

Loop invariant of LR parser

Before and after each step in an LR parser the following holds:

- There exists an ω such that $S' \overset{*}{\underset{rm}{\Rightarrow}} \gamma\omega$, where γ is the string on the stack. In other words, γ is a viable prefix.
- $\gamma \overset{*}{\underset{rm}{\Rightarrow}} \alpha$, where α is the string of the terminal symbols that have been read.

Or, in one fell swoop: $S' \overset{*}{\underset{rm}{\Rightarrow}} \gamma\omega \overset{*}{\underset{rm}{\Rightarrow}} \alpha\omega$

That γ is a viable prefix means that

- there exists an ω such that $S' \overset{*}{\underset{rm}{\Rightarrow}} \gamma\omega$, and
- no handle that can be considered for a *reduce* operation has disappeared on the stack(i.e. symbols have not been pushed on top of such a handle, which would then prevent the *reduce* operation to be applied).

LR(0) kernel items

- $[S' \to \cdot S]$ is a kernel item, where S is the old starting symbol and S' is the new one.
- $[A \to \alpha \cdot \beta]$ is a kernel item if α is not ϵ.

LR(1) kernel items are defined similarly. Two LR(0) or LR(1) states are equivalent if and only if they contain the same kernel items.

LR(1) state machines and item sets

Staring state.

The starting state is

$$I_0 = \text{closure}(\{[S' \to \cdot S, \$]\})$$

where S is the old starting symbol and S' is the new one.

The closure function.

- closure(I) contains I as a subset.
- If $[A \to \alpha \cdot B\beta, a]$ is in closure(I) and $B \to \gamma$ is a rule then $[B \to \cdot\gamma, b]$ is in closure(I), for all b in FIRST(βa).

The goto function.

- goto$(I, X) =$ closure(goto(I, X))
- If $[A \to \alpha \cdot X\beta, a]$ is in I then $[A \to \alpha X \cdot \beta, a]$ is in goto(I, X).

LR(1) and LALR(1) parsing tables

- If a is a terminal symbol and s is a state number and goto$(I_s, a) = I_k$ is not empty then *shift* k is in $M[s, a]$.
- If s is a state number, a is a terminal and $[A \to \alpha\cdot, a]$ is in I_s then *reduce* $A \to \alpha$ is in $M[s, a]$.

LALR(1) item sets

For simplicity we assume that we start out by computing the set of LR(0) items for the grammar. Assume that this set is $\{I_0, \ldots, I_n\}$. Then we can compute the LALR(1) item sets, $\{I'_0, \ldots, I'_n\}$ according to the following rules:

- $[S' \to \cdot S, \$]$ is in I'_0.
- $I'_k =$ closure(I'_k).
- If $[A \to \alpha \cdot X\beta, a]$ is in I'_k then $[A \to \alpha X \cdot \beta, a]$ is in goto(I'_k, X).
- goto$(I'_k, X) = I'_s$ if and only if goto$(I_k, X) = I_s$ (in other words, each LALR(1) state corresponds to exactly one LR(0) state).

This is exactly as when the LR(1) item sets are computed, except that we decide beforehand that certain sets are equivalent because the goto function is determined beforehand according to that rule.