1 Tips

- ϵ is not a terminal symbol

2 Context-free Grammars

Left-associative - $(1 \oplus 2) \oplus 3$

$$E \to E \oplus T \mid T$$
$$T \to N$$

Right-associative - $1 \oplus (2 \oplus 3)$

$$E \to T \oplus E \mid T$$
$$T \to N$$

$$E \to TE'$$
$$E' \to \epsilon \mid \oplus TE'$$
$$T \to N$$

3 Left-factoring and left-recursion removal

3.1 Removing left recursion

$$E \to E \oplus T \mid T$$

is transformed into

$$E \to TE'$$
$$E' \to \epsilon \mid \oplus TE'$$

This transformers a left-associative grammar into a right-associative grammar

4 First, Follow, and LL(1)

4.1 Calculating First sets

- If production of form $N \to \epsilon$, add ϵ to first set for N to indicate nullability
- If production of form $N \to S_1 S_2 \dots S_n$, then if $\forall i \in 1..n, \forall j \in 1..i-1 \cdot S_j$ is nullable, we add current first set for S_i to first set for N
- If every construct $S_1, \dots, S_n$ is nullable, add ϵ to first set for N

Perform for all productions, repeating the process until no sets are modified

5 Shift/reduce parsing

5.1 LR(x) parsing action conflicts

There is no such thing as a *shift/shift* conflict

5.2 LR(1) parsing algorithm

1. Perform state transition, after dequeueing start symbol of the *Input* queue
   (a) If state transition was *shift*, put dequeued symbol on the *Parsing stack*
   (b) If state transition was *reduce*, pop start symbol of the **RHS** of the reduction, and all stack elements above the start symbol, off the stack. Transition to state indicated by number currently on top of stack. Put reduced symbol on the stack. *If LR(1), choose production s.t.* $queue_0 \in T$*, where* T *is look-ahead set.* Follow transition path of current state, based on the reduced symbol.
   (c) If performed state transition was *accept*, do nothing
2. Put number indicating current state on the stack

5.3 Subsection Header

5.3.1 Subsubsection Header