Canonical (general) LR

- General LR(k) machine construction:

  method 1: find out closure/goto.. with LR(k) items (textbook)

  method 2: find LR(k) items and construct NFA from them (Sudkamp's book)

- Finding LR(1) items

  LR(0) items are of the form [$A \rightarrow \alpha \cdot \beta$]

  shows 'local' parsing configuration in the RHS

LR(1) items are $[A \rightarrow \alpha \cdot \beta, a]$

a is the next symbol after RHS of A (non-local LA)

$$LR1 - context(A \rightarrow w) = LR0 - context(A \rightarrow w) FOLLOW_1(A)$$

if $z \in FOLLOW_1(A)$ and $S \Rightarrow_R^* uAv$, then uwz is a handle (viable prefix)

if $\beta \neq \epsilon$ in an LR(1) item, LA (lookahead) is not used

if $\beta = \epsilon$, then the item is like $[A \rightarrow \alpha \cdot, a]$, which means reduce by that rule if next symbol (which is now the next symbol after RHS) is a

- *valid* LR(1) items: things to appear on stack top

  $S \Rightarrow uAv \Rightarrow uw_1w_2av \ \ (v \in \Sigma^*)$

  $[A \rightarrow w_1 \cdot w_2, x]$ is valid if uw_1 is a viable prefix and $x \in \Sigma \cup \{\$\}$

- Why try SLR if LR is more general? LR tables are two or three orders of magnitude larger than SLR tables. But not every LR grammar is SLR. Lookahead LR (LALR) combines LR(1) states with common 'core' (LR(0) states) hence reducing the number of states. LALR seems to be practical for PL grammars.

- if the merged states do not introduce conflicts, the grammar is LALR(1)

- SLR $\subset$ LALR $\subset$ LR