



# Compilers

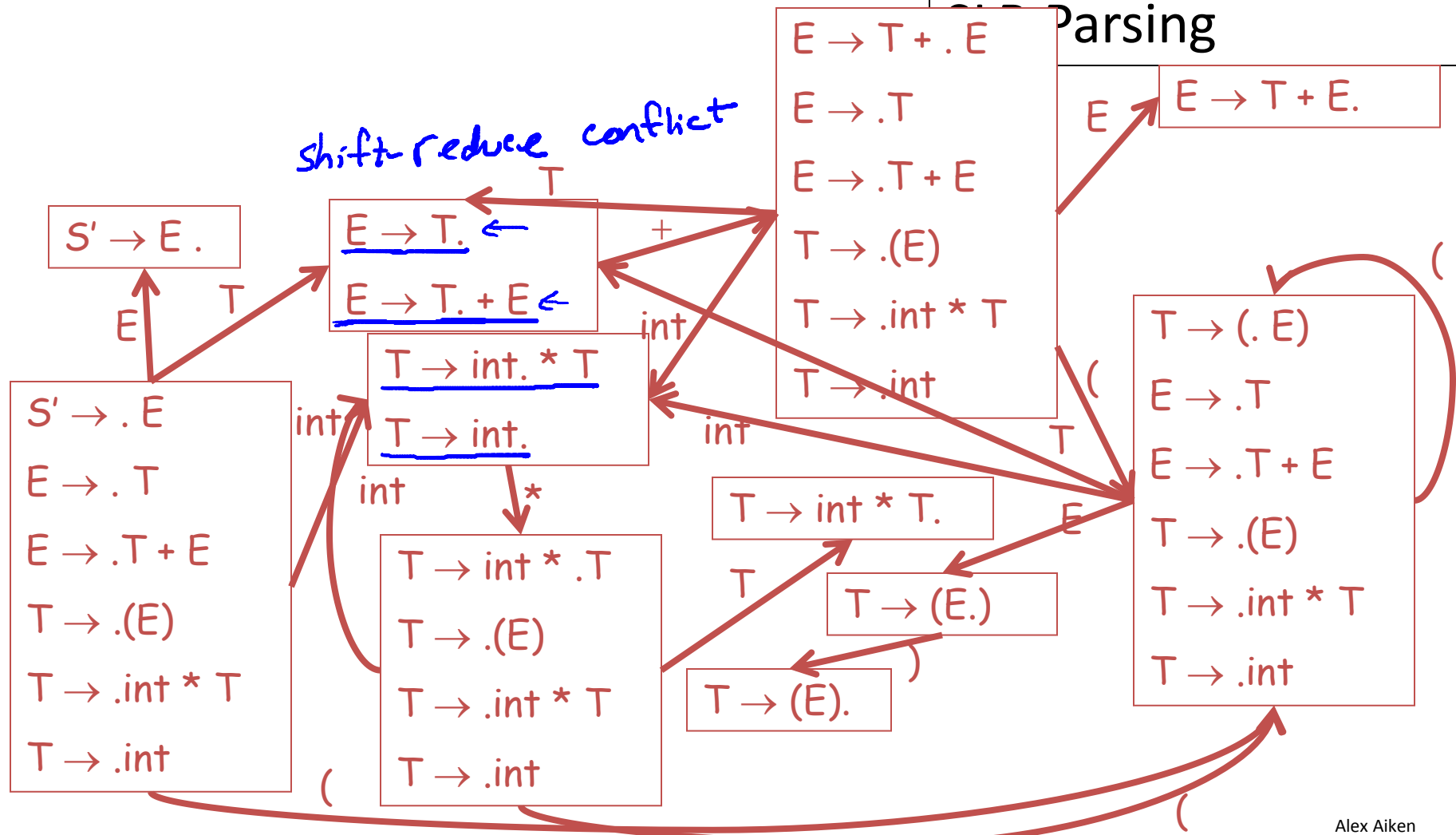
---

## SLR Parsing

- LR(0) Parsing: Assume
  - stack contains  $\alpha$
  - next input is  $t$
  - DFA on input  $\alpha$  terminates in state  $s$
- Reduce by  $X \rightarrow \beta$  if
  - $s$  contains item  $X \rightarrow \beta$ .
- Shift if
  - $s$  contains item  $X \rightarrow \beta.t$
  - equivalent to saying  $s$  has a transition labeled  $t$

- LR(0) has a reduce/reduce conflict if:
  - Any state has two reduce items:
    - $X \rightarrow \beta.$  and  $Y \rightarrow \omega.$
- LR(0) has a shift/reduce conflict if:
  - Any state has a reduce item and a shift item:
    - $X \rightarrow \beta.$  and  $Y \rightarrow \omega.t\delta$

shift-reduce conflict



- SLR = “Simple LR”
- SLR improves on LR(0) shift/reduce heuristics
  - Fewer states have conflicts

- Idea: Assume
  - stack contains  $\alpha$
  - next input is  $t$
  - DFA on input  $\alpha$  terminates in state  $s$

- Reduce by  $X \rightarrow \beta$  if

- $s$  contains item  $X \rightarrow \beta.$
- $t \in \text{Follow}(X)$

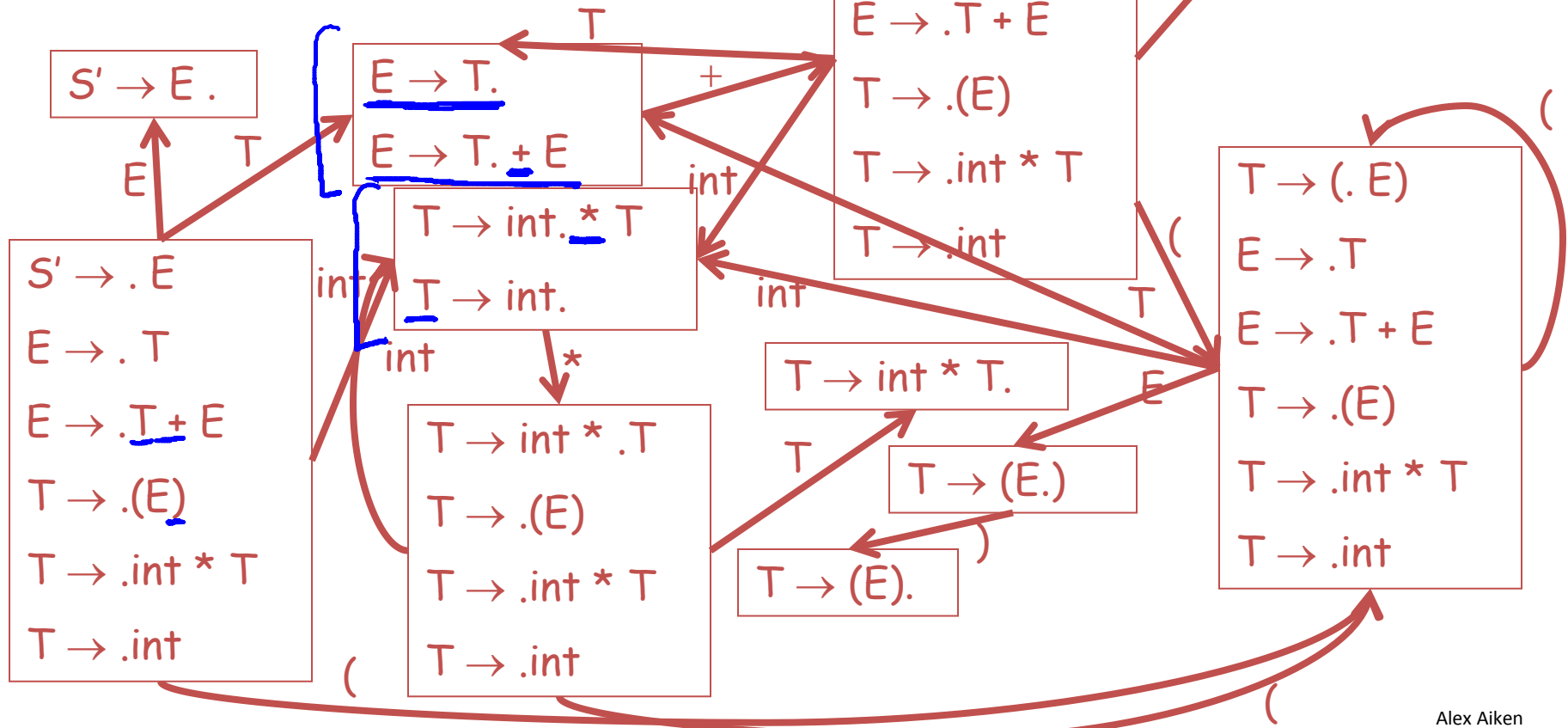
$\rightarrow$   $\begin{array}{l} \dots \beta | t \\ \dots \underline{X} | t \end{array}$

- Shift if

- $s$  contains item  $X \rightarrow \beta.t\omega$

- If there are conflicts under these rules, the grammar is not SLR
- The rules amount to a heuristic for detecting handles
  - The SLR grammars are those where the heuristics detect exactly the handles

$\text{Follow}(E) = \{\$, )\}$   
 $\text{Follow}(T) = \{\$, ), +\}$






- Lots of grammars aren't SLR
  - including all ambiguous grammars
- We can parse more grammars by using precedence declarations
  - Instructions for resolving conflicts

- Consider our favorite ambiguous grammar:
  - $E \rightarrow E + E \mid E * E \mid (E) \mid \text{int}$
- The DFA for this grammar contains a state with the following items:
  - $E \rightarrow E * E \cdot$       ~~$E \rightarrow E \cdot + E$~~
  - shift/reduce conflict!
- Declaring “ $*$  has higher precedence than  $+$ ” resolves this conflict in favor of reducing

- The term “precedence declaration” is misleading
- These declarations do not define precedence; they define conflict resolutions
  - Not quite the same thing!

1. Let  $\underline{M}$  be DFA for viable prefixes of  $G$
2. Let  $\underline{S} | x_1 \dots x_n \underline{\$}$  be initial configuration
3. Repeat until configuration is  $\underline{S} | \underline{\$}$ 
  - Let  $\underline{\alpha} | \underline{w}$  be current configuration
  - Run  $\underline{M}$  on current stack  $\underline{\alpha}$
  - ~~If  $\underline{M}$  rejects  $\underline{\alpha}$ , report parsing error~~
    - Stack  $\underline{\alpha}$  is not a viable prefix
  - If  $\underline{M}$  accepts  $\underline{\alpha}$  with items  $\underline{I}$ , let  $\underline{a}$  be next input
    - Shift if  $\underline{X \rightarrow \beta. \underline{a} \gamma} \in \underline{I}$
    - Reduce if  $\underline{X \rightarrow \beta.} \in \underline{I}$  and  $\underline{a} \in \text{Follow}(X)$
    - Report parsing error if neither applies 

- If there is a conflict in the last step, grammar is not  $\text{SLR}(k)$
- $k$  is the amount of lookahead
  - In practice  $k = \underline{1}$