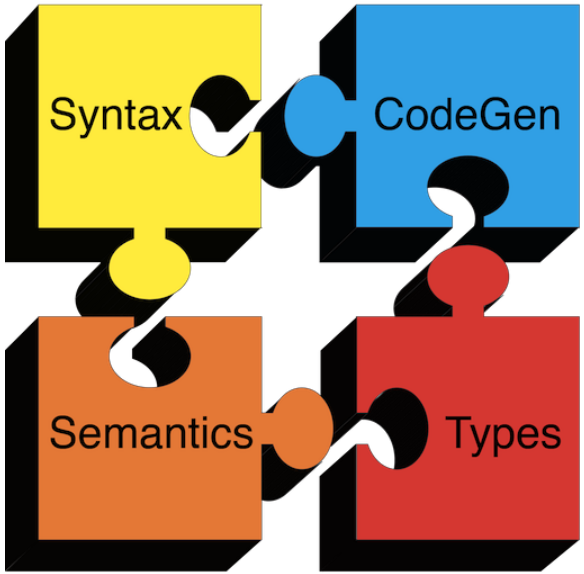
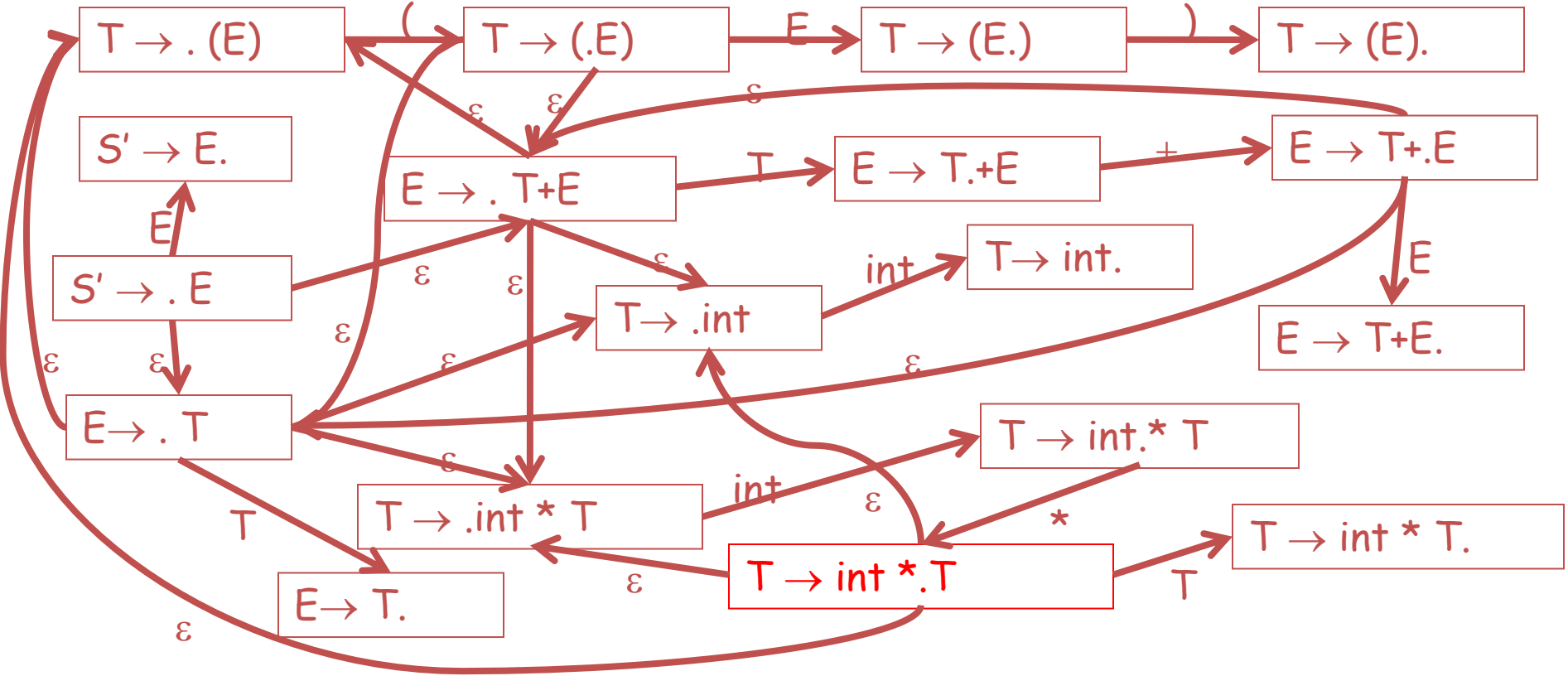


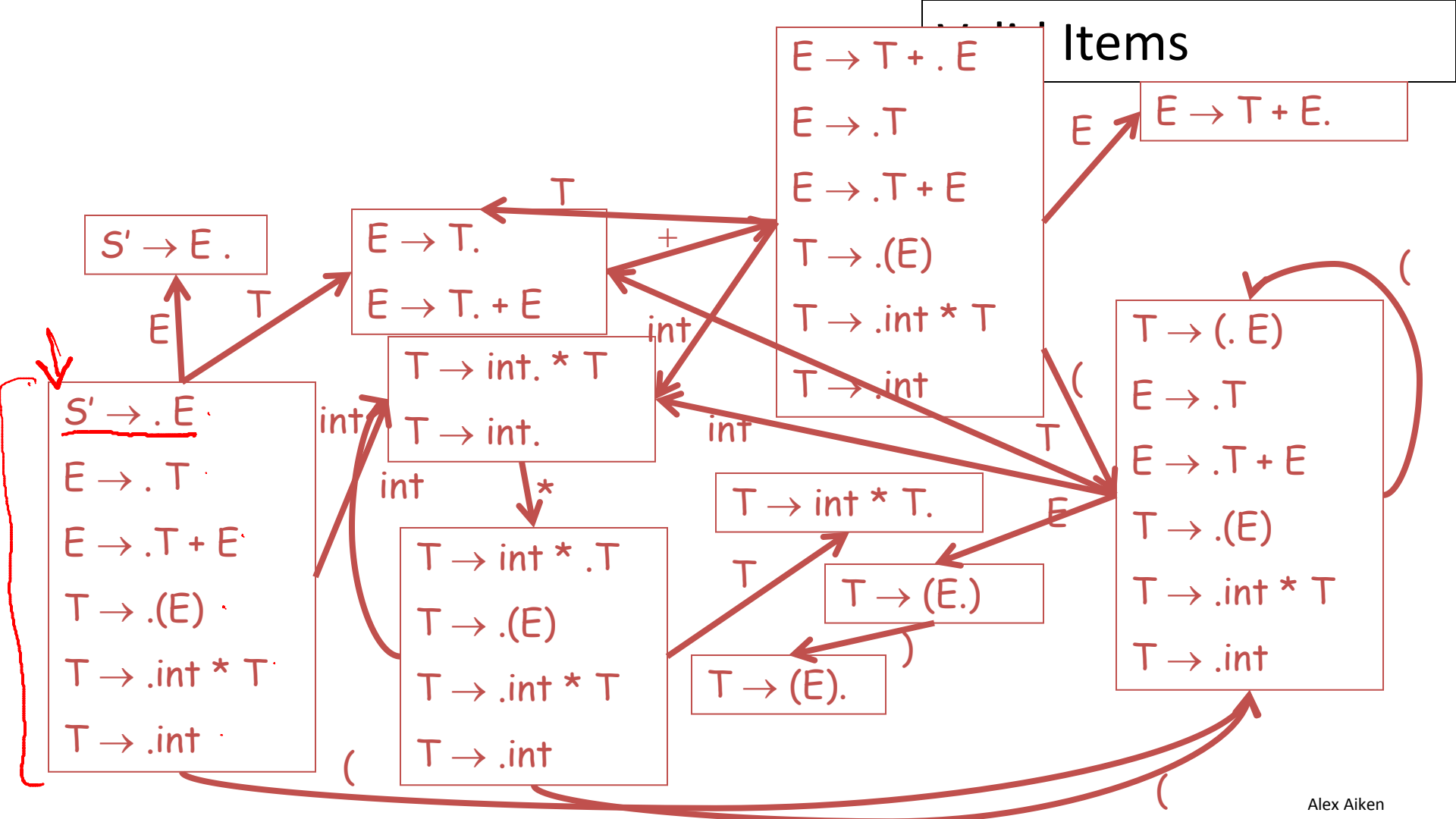
Compilers

Valid Items

Valid Items


T → . (E)
T → (.E)
T → (E.)
T → (E).
S' → E.
E → . T+E
E → T.+E
E → T+.E
S' → . E
T → int.
T → .int
E → T+E.
E→ . T
T → int.* T
T → .int * T
T → int *.T
T → int * T.
E→ T.
(
E
)
+
int
*
T
ε

Items


S' → E .
S' → . E
E → . T
E → .T + E
T → .(E)
T → .int * T
T → .int
E → T.
E → T. + E
T → int. * T
T → int.
T → int * .T
T → .(E)
T → .int * T
T → .int
E → T + . E
E → .T
E → .T + E
T → .(E)
T → .int * T
T → .int
E → T + E.
T → int * T.
T → (E.)
T → (E).
T → (. E)
E → .T
E → .T + E
T → .(E)
T → .int * T
T → .int
E
T
+
int
*
(
)

Valid Items

The states of the DFA are

"canonical collections of items"

or

"canonical collections of LR(0) items"

The Dragon book gives another way of constructing LR(0) items

Valid Items

Item $X \to \beta.\gamma$ is *valid* for a viable prefix $\alpha\beta$ if

$$S' \to^* \alpha X \omega \to \alpha\beta\gamma\omega$$

by a right-most derivation

After parsing $\alpha\beta$, the valid items are the possible tops of the stack of items

Valid Items

An item I is valid for a viable prefix α if the DFA recognizing viable prefixes terminates on input α in a state s containing I

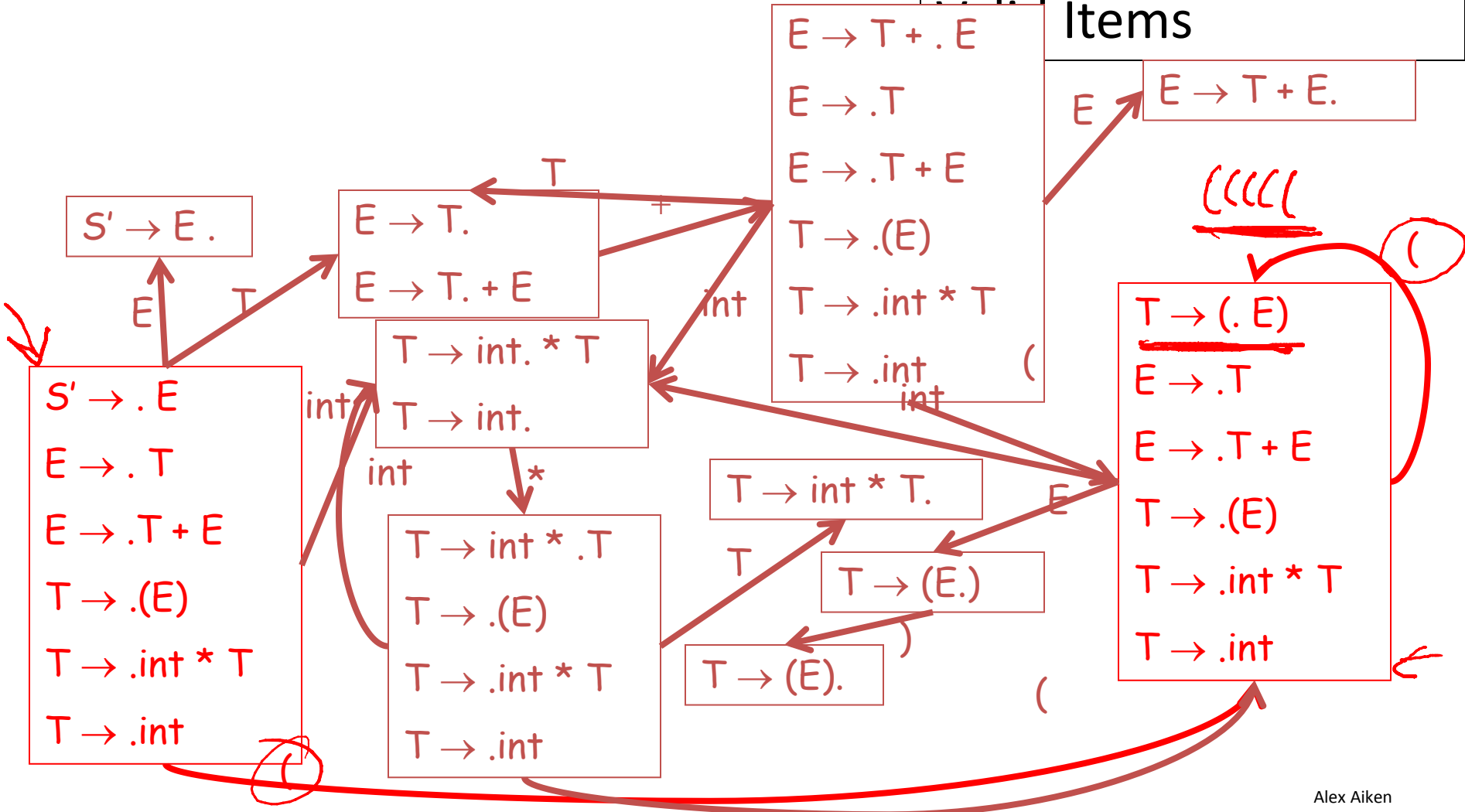
The items in s describe what the top of the item stack might be after reading input α

Valid Items

- An item is often valid for many prefixes

- Example: The item $T \rightarrow (.E)$ is valid for prefixes

Items


S' → . E
E → . T
E → .T + E
T → .(E)
T → .int * T
T → .int
S' → E .
E → T.
E → T. + E
T → int. * T
T → int.
T → int * .T
T → .(E)
T → .int * T
T → .int
T → int * T.
T → (E.)
T → (E).
E → T + . E
E → .T
E → .T + E
T → .(E)
T → .int * T
T → .int
E → T + E.
T → (. E)
E → .T
E → .T + E
T → .(E)
T → .int * T
T → .int
E
T
int
*
+
(
)

Valid Items

Using the automaton on the previous slide, choose the valid items for the prefix: (int *

- ☐ E → (.E)
- ☐ T → int *.T
- ☐ E → .T + E
- ☐ T → .int

To show the automaton, click "Hide Question"

↓