← • HOME • UP • →

Top/Bottom-up parsing/LR(0) machine

9.4. The LR(0) Finite State Machine

An LR parser finds the handle (or decides to do a shift) by reading the stack from bottom to top using a finite-state machine. The state in which the machine stops tells the parser what to do.

For an LR(0) parser, each state of the finite-state machine is labeled by a set of LR(0) items.

The start state

Suppose that S' is the start state of an augmented grammar, and the sole production for S' is $S' \rightarrow S$.

The start state is labeled by the closure of set $\{S' \rightarrow \cdot\ S\}$. Here is the start state for our augmented expression grammar.

```
 ___________________
| E' → · E          |
| E → · T           |
| E → · E + T       |
| T → · F           |
| T → · T * F       |
| F → · n           |
| F → · ( E )       |
|___________________|
```

Remark. This start state happens to include all of the productions. That does not always happen. You only get the LR(0) items that come from the closure process.

Transitions out of the start state

Now find all of the symbols that immediately follow a dot in the start state. There is a transition coming out of the start state for each of those symbols.

In the state shown above, there are 5 symbols that follow a dot: E, T, F, **n** and **(**.

For each of those symbols σ, find all of the LR(0) items in the current state where σ immediately follows the dot.

Make a new state q_σ, and put into q_σ all of those items with the dot moved to just after σ.

Now take a closure in that state. The transition labeled σ goes to state q_σ.

An example makes that much more clear. In the start state shown above, there are two LR(0) items with E immediately after the dot.

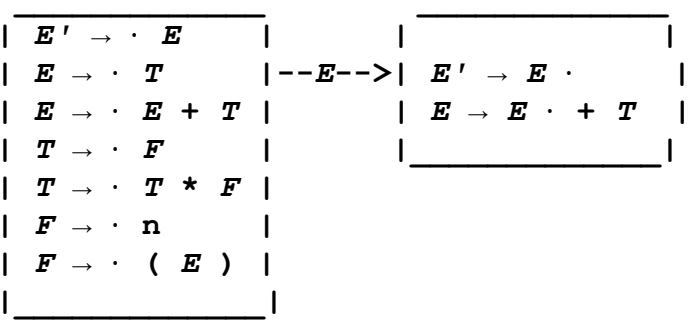
$$E' \rightarrow \cdot\ E$$
$$E \rightarrow \cdot\ E + T$$

Moving the dot across E in those items yields

$$E' \rightarrow E\ \cdot$$
$$E \rightarrow E\ \cdot + T$$

Forming the closure of that set of LR(0) items does not change it because neither of the two items has a dot followed by a nonterminal.

Here is a slightly larger part of the finite state machine.



The transition labeled '**(**' coming out of the start state goes to a state initially labeled by set

$$\{F \rightarrow \mathbf{(}\ \cdot\ E\ \mathbf{)}\}$$

Taking the closure yields

$$F \rightarrow \mathbf{(}\ \cdot\ E\ \mathbf{)}$$
$$E \rightarrow \cdot\ T$$
$$E \rightarrow \cdot\ E + T$$
$$T \rightarrow \cdot\ F$$
$$T \rightarrow \cdot\ T\ \mathbf{*}\ F$$
$$F \rightarrow \cdot\ \mathbf{n}$$
$$F \rightarrow \cdot\ \mathbf{(}\ E\ \mathbf{)}$$

Building the full LR(0) finite state machine

You must do the same thing for every state q, providing it with transitions labeled by each symbol that occurs immediately after a dot in q.

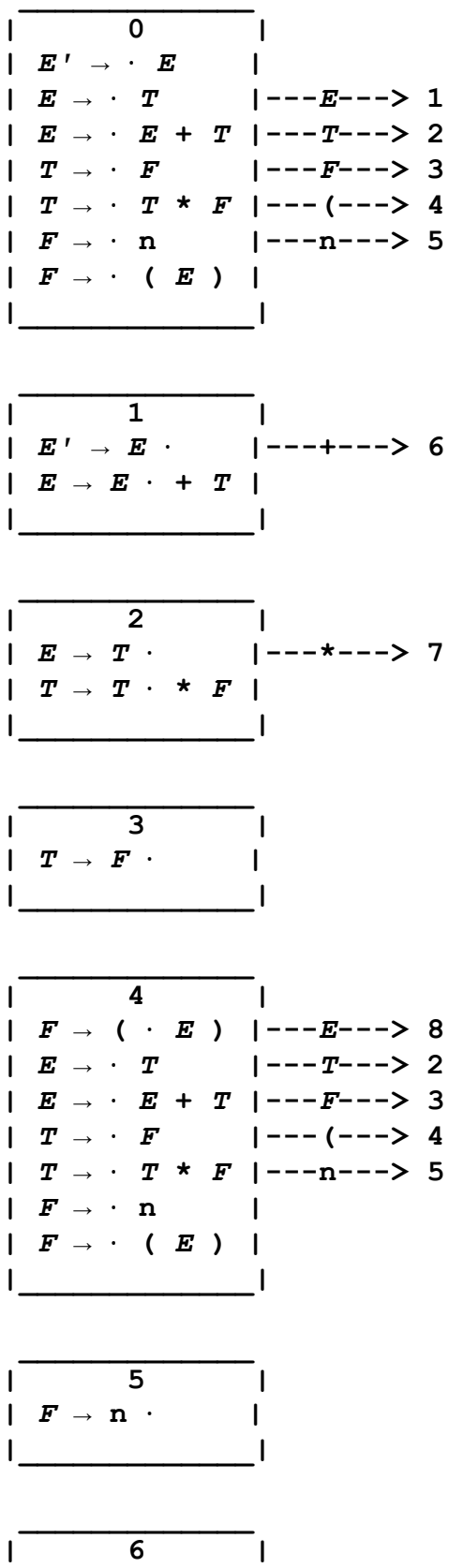
It is important to have only one copy of each state. No two states should contain exactly the same set of LR(0) items.

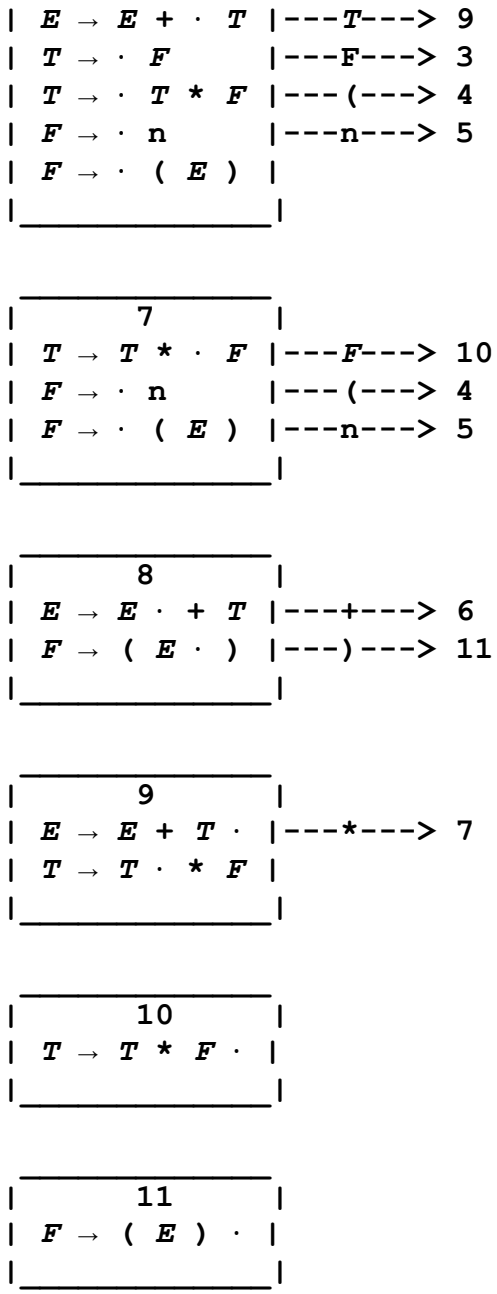
If you find that you have built a new state that has exactly the same items in it as an existing state, make the transition go to that existing state instead.

Here is the complete machine for our augmented expression grammar.

It is too awkward to show the whole thing as a diagram, so states are numbered, and transitions show the state by number.

This finite state machine is shown on page 244 of the Dragon Book.





← • HOME • UP • →

Top/Bottom-up parsing/LR(0) machine