for the given grammar

$E \to E + T$

$E \to T$

$T \to T * F$

$T \to F$

$F \to id$

$F \to (E)$

$\Sigma = \{id, +, *, (,)\}$

we say a LR parser accepts a program when the stack has the starting starting symbol and remaining input is empty.

Stack	input
$E	$

↳ this configuration says that the program is accepted grammatically accurate.

Step 1 ☺

But what if:

Stack	input
$ .... E	$

since we can only see the topmost element of stack

To avoid the scenario we add a new non-terminal and appoint it as the new start symbol & E' will not be in any other production rule.

$E' \to E$ → augmentation of the grammer

$E \to E + T$

$E \to T$

$T \to T * F$

$T \to F$

$F \to id$

$F \to (E)$

Step 2: Constructing automaton → LR(0)

↳ automata is plural of automaton

what is a state in the automaton?

→ In DFA state means how much string we have consumed. (since DFA is deterministic)

But for CFG:

we have many options through the grammer.

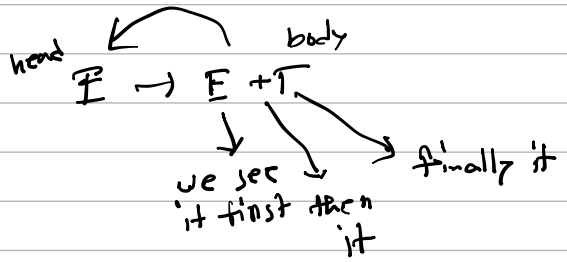
(what we have seen so far) + (what we expect to see next (or in the future (not only the immediate next))

↓ same as DFA

↓

meaning of state in automaton

* here the bottom up approach, we see the body then transition to head.



we can also write in this manner:

we can only shift ← $E \rightarrow \bullet E + T$

it works like a cursor. How much of this production we have seen so far.

this at first means we have not seen anything so far.

⇒ these are called LR(0) items.

only shift ⇐ $E \to E \bullet + T$

I have seen E in the body and hoping to see '+' in future.

we are ready to reduce ⇐ $E \to E + T \bullet$

I have seen the whole of a body and now I expect to reduce the body into the head.

∴ A state of the automaton is the set of production parts we think we have seen partially or whole.

* strings are consist of terminals (tokens).

$(id + id) * id$

where is E here?

$E \to E \bullet + T$

I have seen part of the input that I reduced to E earlier.

to ensure end we also include the $ in the sigma.

$\Sigma = \{id, +, *, (,), \$\}$

if we have $A \to \alpha \bullet B\beta$ in a state

and $B \to \gamma_1$, $B \to \gamma_2$, ⋮, $B \to \gamma_n$ are B's productions

we have to add all items: $B \to \bullet\gamma_1$, $B \to \bullet\gamma_2$, ⋮, $B \to \bullet\gamma_n$ in the state.

now F can be reduced to E

same for T

that's why we add all possible rules of E.

first state

$E' \to .E$ → E has 2 more production rules we have to add them as well

$E \to .E + T$ ← we have already considered E & set does not approve duplicate

$E \to .T$ → T has 2 more ∴ add

$T \to .T * F$

$T \to .F$ → same for F

$F \to .id$

$F \to .(E)$

I_0

$E' \to .E$

$E \to .E + T$

$E \to .T$

$T \to .T * F$

$T \to .F$

$F \to .id$

$F \to .(E)$

E → we don't know which E? so we have to increment both

I_1

$E' \to E.$

$E \to E. + T$

+

I_2

$E \to E + .T$

$T \to .T * F$

$T \to .F$

$F \to .id$

$F \to .(E)$

for

for

$ → accept

T

I_3

$E \to T.$

$T \to T. * F$

→ this type of state asks to expand/shift more

will expand more

F

id

I_4

$T \to F.$

I_5

$F \to id.$

*

I_7

$T \to T * .F$

$F \to .id$

$F \to .(E)$

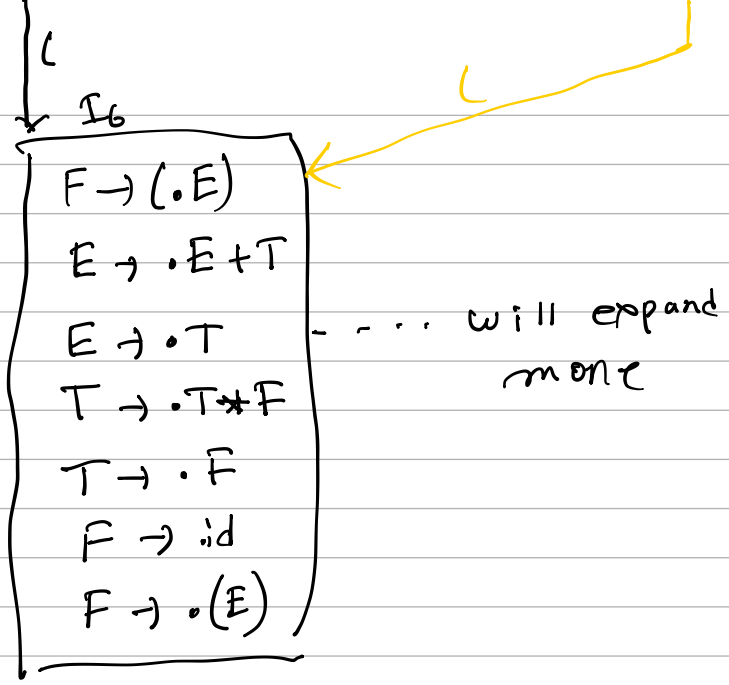
F

I_8

$T \to T * F.$

→ this type of states indicates we can reduce

id



automaton

* CLR can have exponentially more states than SLR for same grammers. in one automaton LALR concises the number of states of CLR.