Precedência de operadores

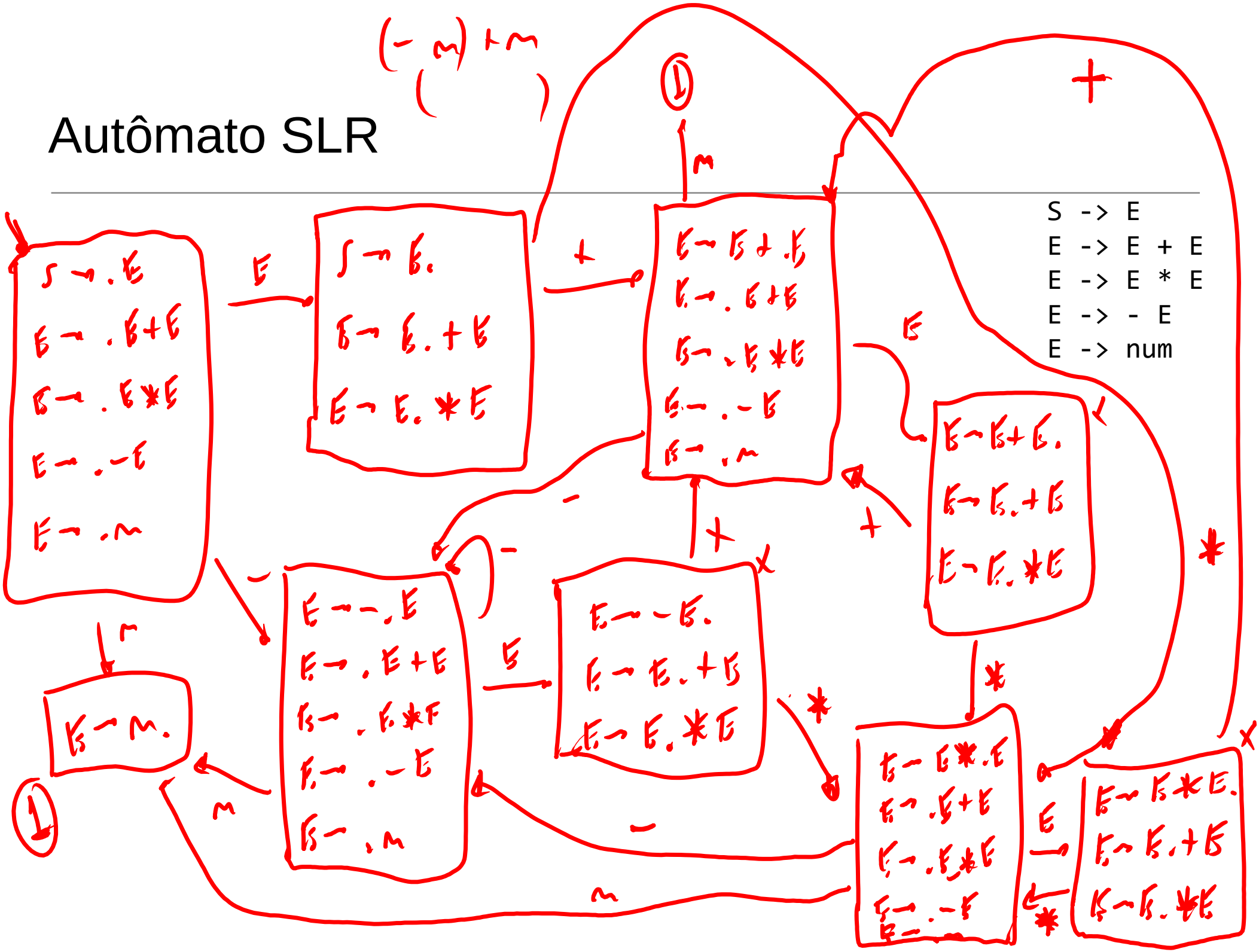
- Vamos agora ver uma gramática mais complexa:

```
S -> E
E -> E + E
E -> E * E
E -> - E
E -> num
```

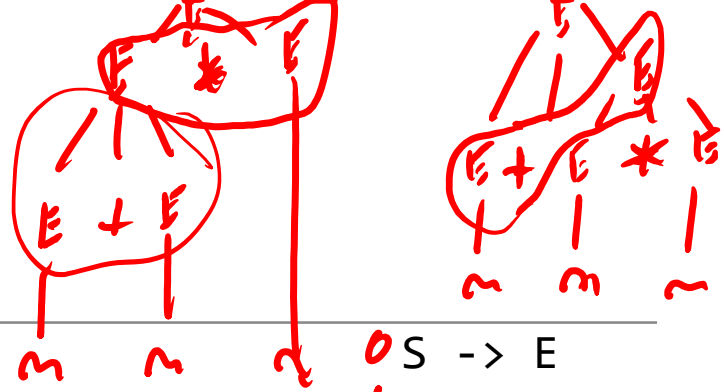
- Qual será o comportamento dessa gramática nas entradas:

```
num + num * num
num * num + num
- num + num
```

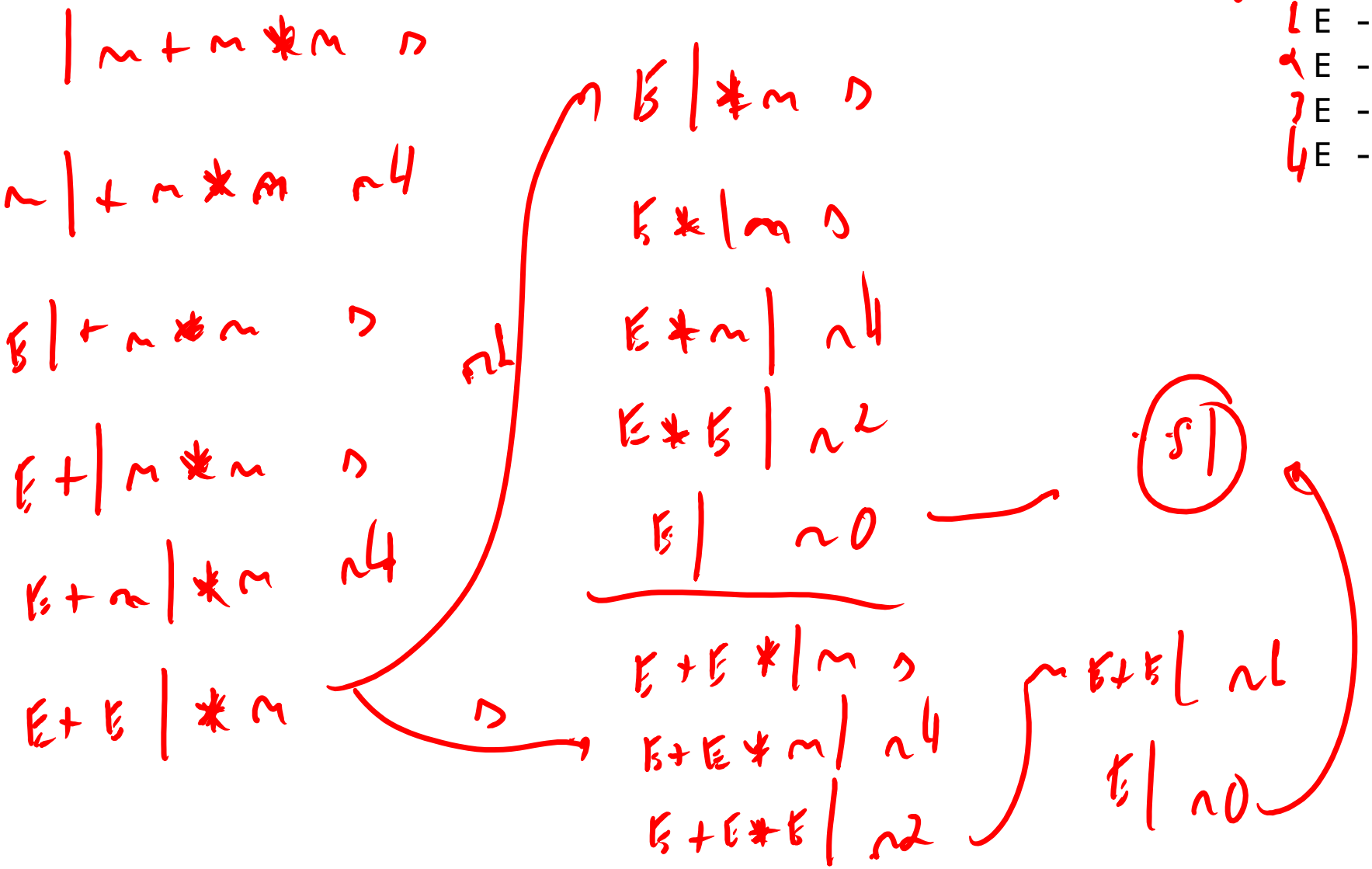
Autômato SLR



Analisado num + num * num

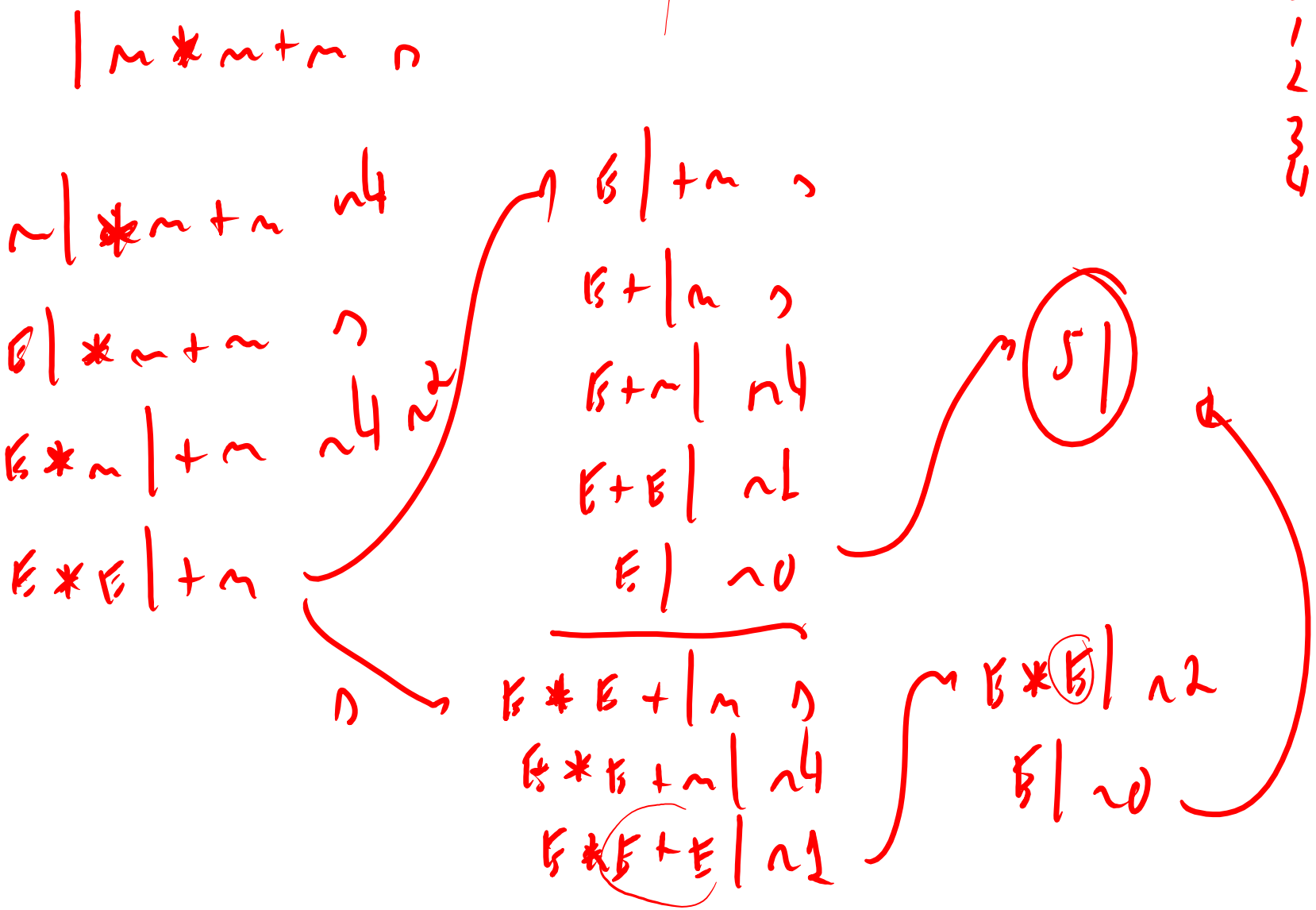


0 S -> E
1 E -> E + E
2 E -> E * E
3 E -> - E
4 E -> num



Analisando num * num + num

0 S -> E
1 E -> E + E
2 E -> E * E
3 E -> - E
4 E -> num



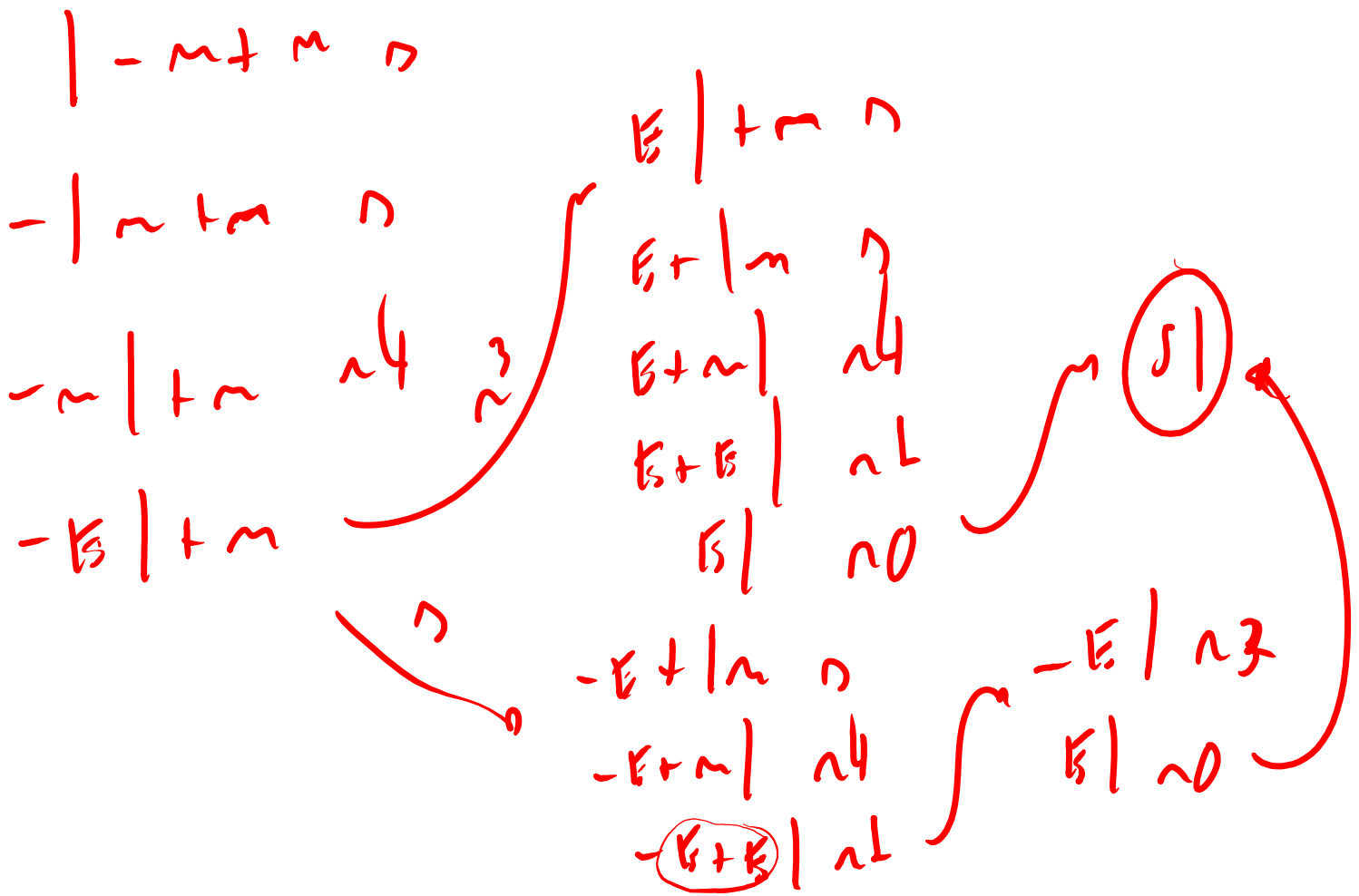
Controle de precedência

- Podemos levar em conta a precedência dos operadores na solução de conflitos shift-reduce
- Se o operador do shift tem precedência maior que a do operador do reduce, fazer shift, senão fazer o reduce
- Isso nos dá a árvore correta nos nossos exemplos, assumindo que a precedência de * é maior que a de +
- E quanto ao operador unário?

Analisando - num + num

- Vai ser a mesma coisa, a precedência dele tem que ser maior que a dos operadores binários

```
0 S -> E
1 E -> E + E
2 E -> E * E
3 E -> - E
4 E -> num
```



Precedência e associatividade

- O controle da precedência e o da associatividade usam o mesmo mecanismo
- Podemos ter ambos no analisador: se um operador é associativo à direita é como se a precedência dele fosse maior do que a dele mesmo, e aí escolhemos shift
- Um resumo da resolução de conflitos shift-reduce:
  - Para o mesmo operador, shift dá associatividade à direita, reduce à esquerda
  - Para operadores diferentes, shift dá precedência ao próximo operador, reduce ao atual

Gramática SLR para TINY

- Podemos dar uma gramática mais simples para TINY se usarmos um analisador SLR com controle de precedência:

```
S -> CMDS
CMDS -> CMDS ; CMD
CMDS -> CMD
CMD  -> if EXP then CMDS end
CMD  -> if EXP then CMDS else CMDS end
CMD  -> repeat CMDS until EXP
CMD  -> id := EXP
CMD  -> read id
CMD  -> write EXP
```

```
EXP  -> EXP < EXP
EXP  -> EXP = EXP
EXP  -> EXP + EXP
EXP  -> EXP - EXP
EXP  -> EXP * EXP
EXP  -> EXP / EXP
EXP  -> ( EXP )
EXP  -> num
EXP  -> id
```

<, =, {+, -}, {*, /}