① instr → 'if' exp 'then' instr 'fi'
② instr → print 'eo'
③ exp → 'true'
④ exp → 'false'

	if	then	fi	print	true	false
instr	①			②		
exp					③	④

First (instr) = {'if', 'print'}

③ exp → neg cond
④ neg → 'not' neg
⑤ neg → ε
⑥ cond → 'true'
⑦ cond → 'false'

First (3) = {not} ∪ First (cond) ⇒ {not, true, false}
First(4) = 'not'

Follow (neg) = {true, false}

First (6) = 'true'
First (7) = 'false'

⇒ if print not true false

rest	①	②			
exp	③	③	③		
neg	④	⑤	⑤		
cond		⑥	⑦		

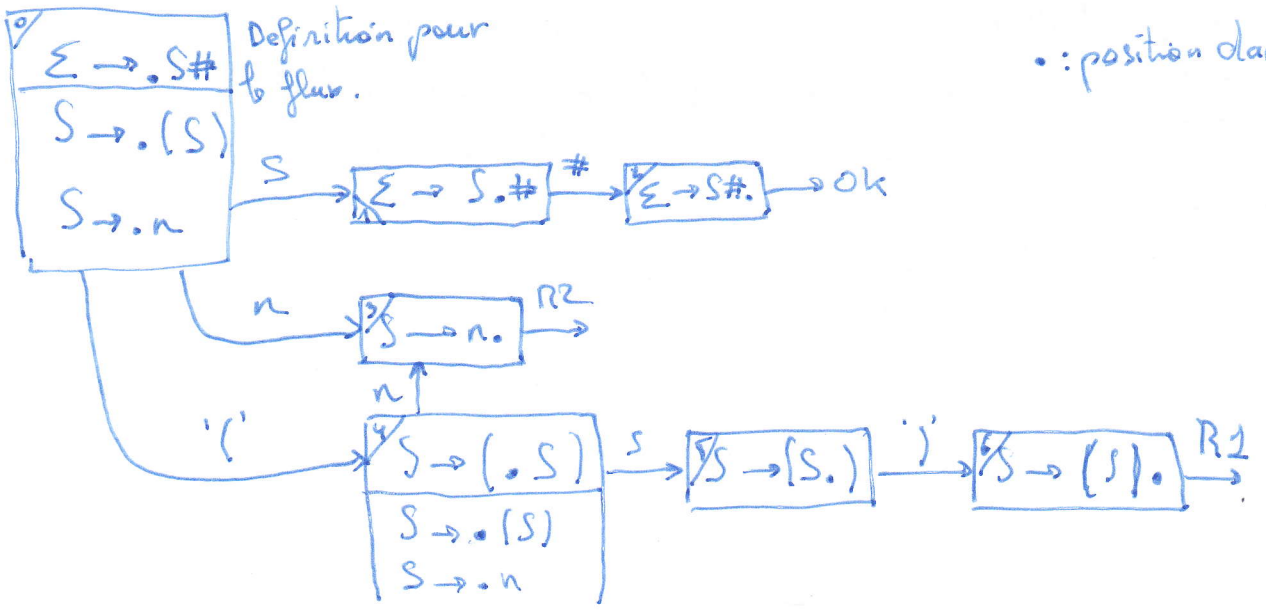
En cas d'erreur, le compilo continue la compilation

Parser LL : Manque d'expressivité
Parser LR : Plus compliqué à écrire à la main.

S → (S)
S → n

⇒ Rajouter une règle qui précise que l'on attend rien derrière

0 : Σ → S#
S → (S)
S → n



•: position dans le flux

	Action	(n)	# S
0	S	4	3 1
1	S		2
2	ACC		
3	R2		
4	S	4	3 5
5	S	6	
6	R1		

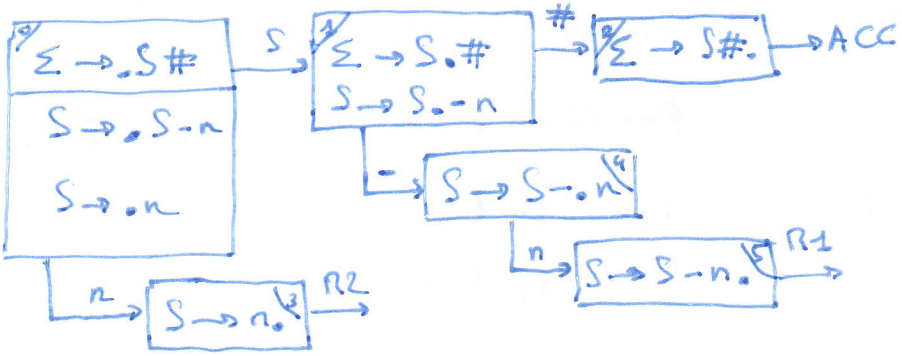
Aucun regard sur les éléments futurs
→ LR(0)

Les flèches entrantes portent toujours le même symbole
⇒ Connaître le n° d'état donne le dernier symbole trouvé

S : Shifter
ACC : Accept
Rn : Review Rule n° n

Exemple 2 : Avec Récurrence gauche

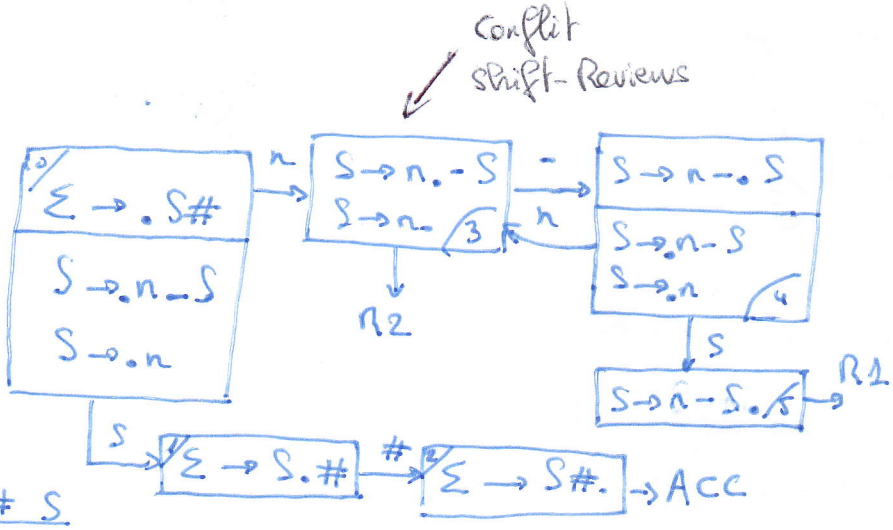
0 : Σ → S#
1 : S → S-n
2 : S → n



⇒ LR(0)

Exemple 3 : Avec récurrence droite

0 : Σ → S#
1 : S → n-S
2 : S → n



	Action	n	-	#	S
0	S	3			1
1	S				2
2	ACC				
3	S/R2	4			
4	S	3			5
5	R1				

Follow (S) : {#} LookHead (#) : R1

- Gère les récurrences droite mais avec un lookahead de 1
- La pile ne fait que croître (⚠ Stack overflow)

LH →	n	-	#	S
0	S3			S1
1				S2
2				ACC
3		S4		R1
4	S3			S5
5				R2

⇒ SLR(1)
↳ simple