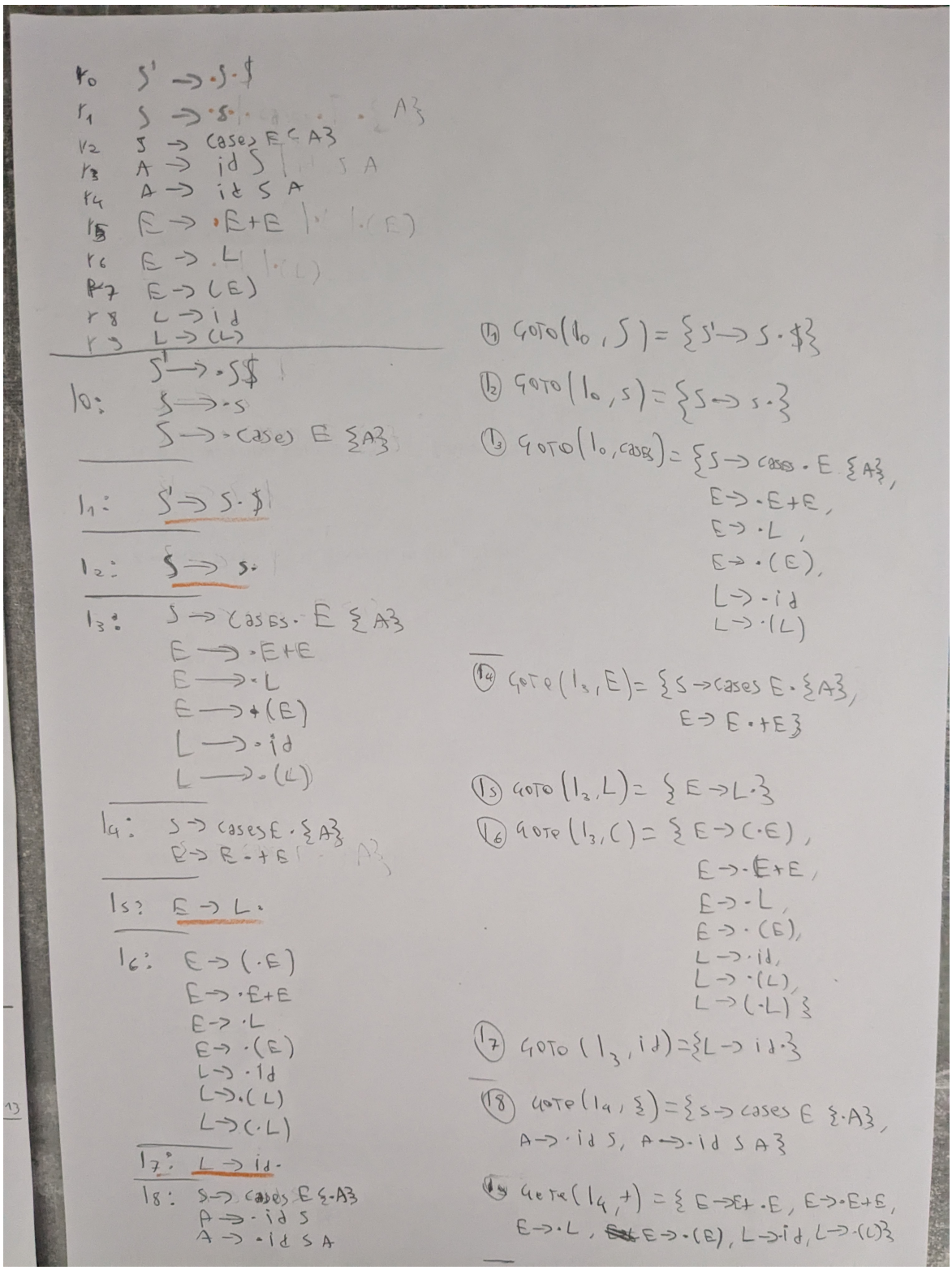
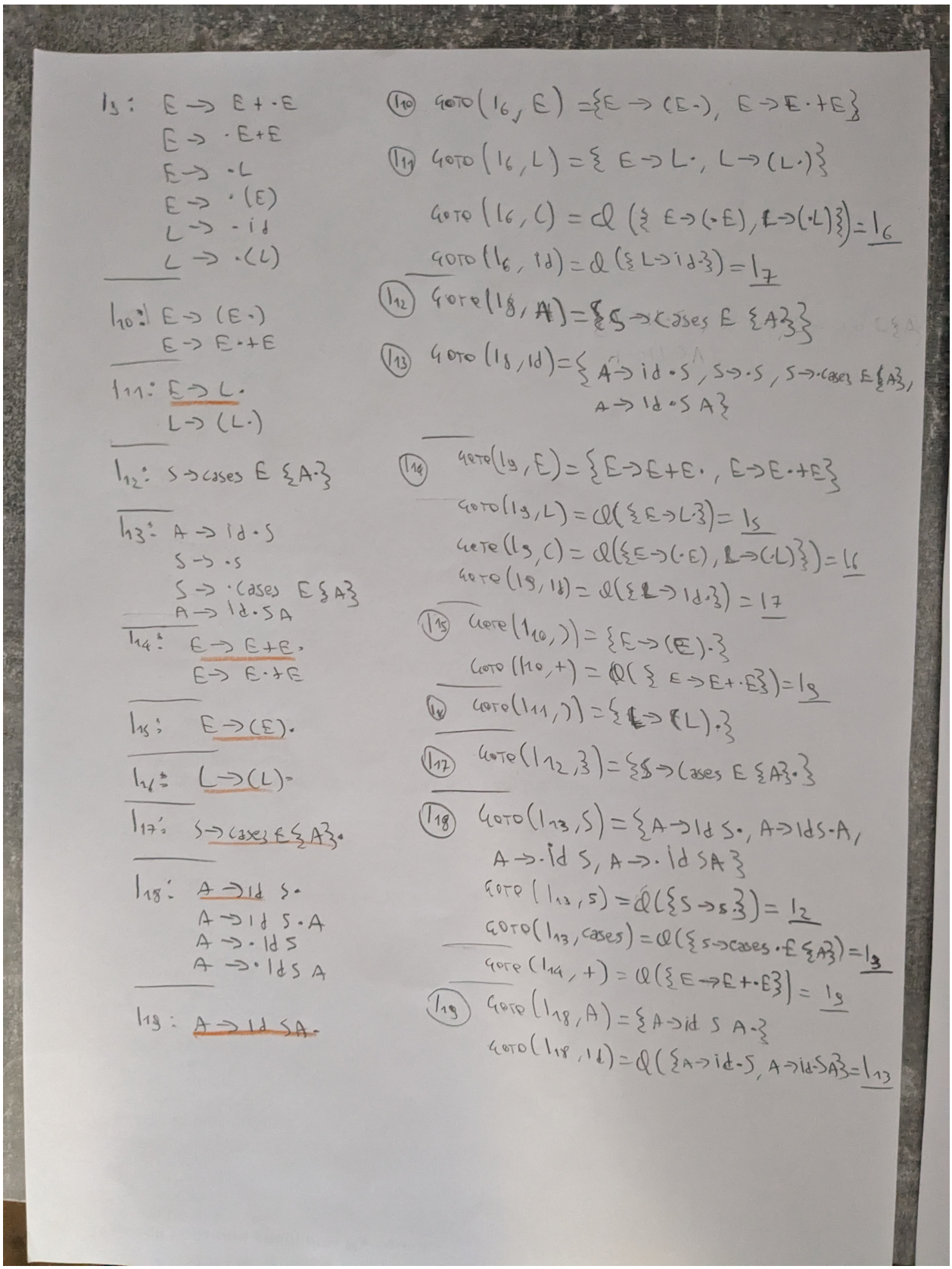
ProgettoLC parteB Alvise Bruniera Relazione

SLR

Ho iniziato numerando le regole della grammatica aumentata ed eseguendo la collezione canonica per trovare gli stati:

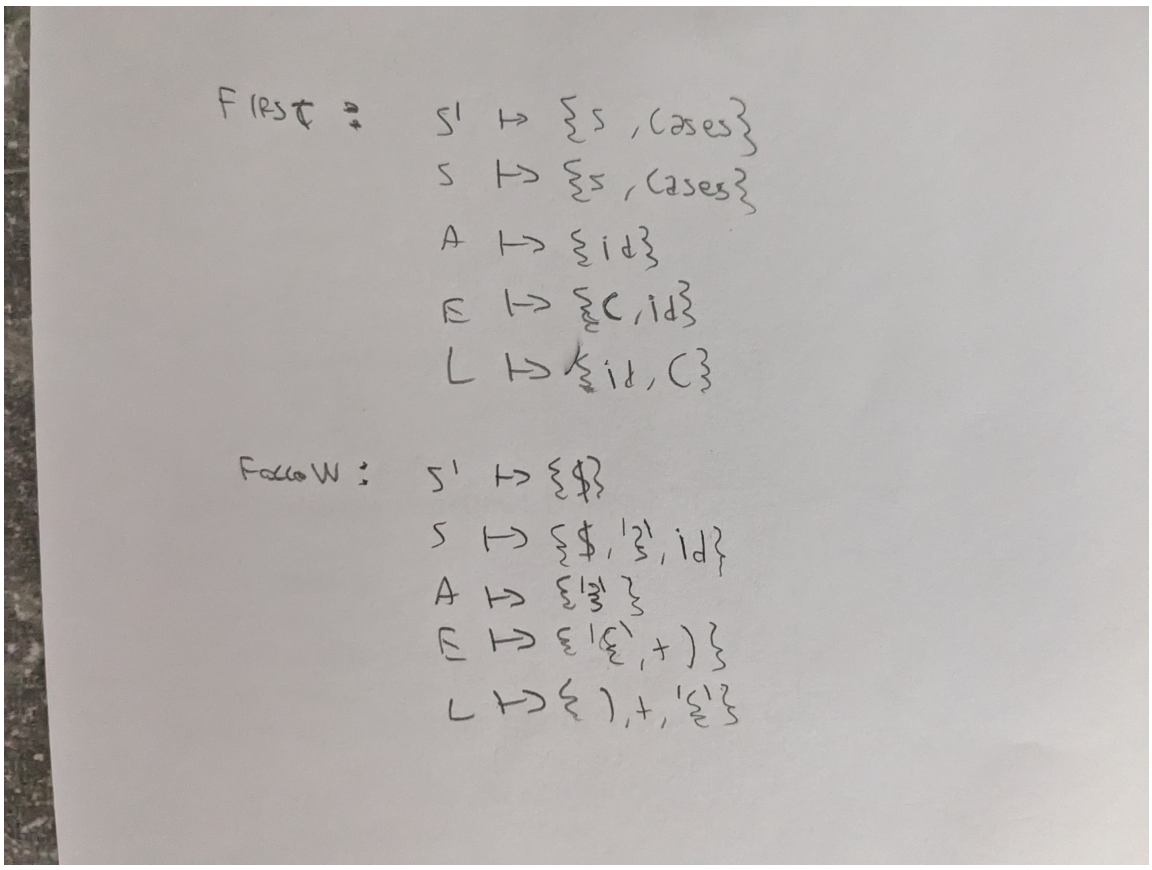




Sono segnati in arancione gli item del tipo "$A \to \alpha \cdot$".

Trovando la collezione canonica ho sia trovato (e numerato) gli stati che calcolato la funzione GOTO.

Poi ho calcolato la funzione FIRST e con quella la funzione FOLLOW per tutti i non terminali.



Adesso posso calcolare la tabella del parser SLR ed aggiungere la gestione degli errori come visto a lezione, utilizzando la funzione GOTO.

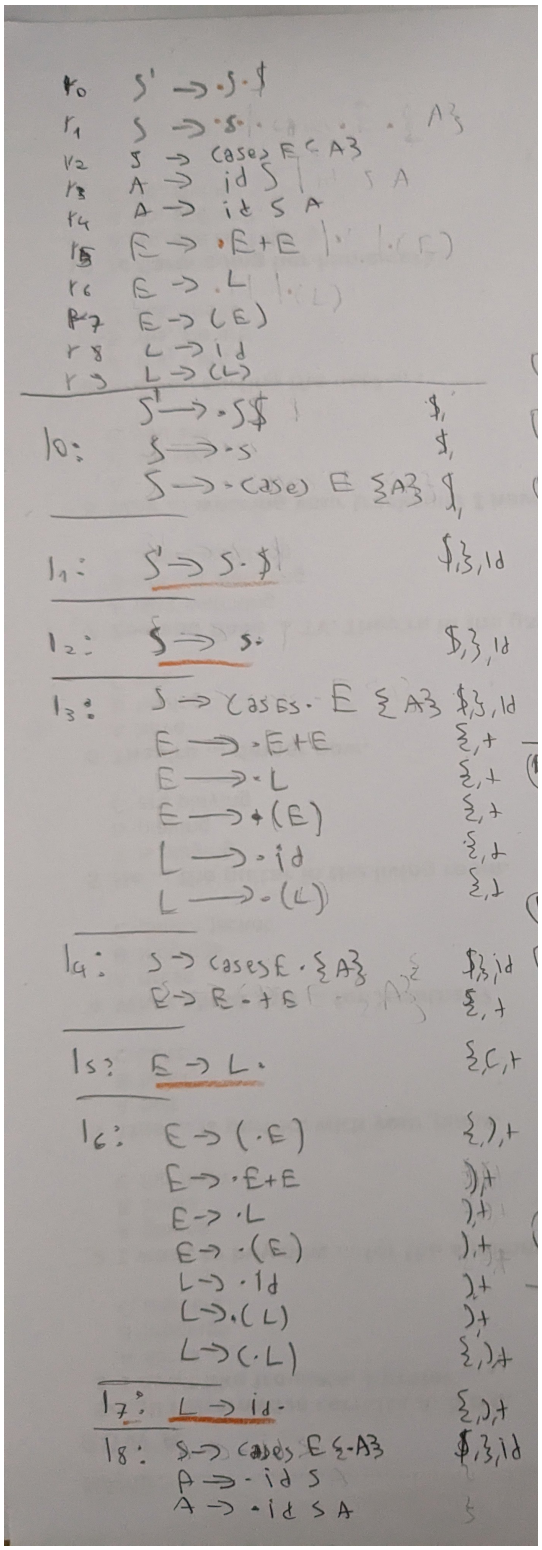
	ACTION									GOTO				
	(	**)**	**+**	**cases**	**id**	**s**	**{**	**}**	**$**	**S'**	**S**	**A**	**E**	**L**
I0	e6	e6	e6	s3	e6	s2	e6	e6	e0	e1	1	e2	e3	e3
I1	e6	e6	e6	e6	e6	e6	e6	e6	acc	e1	e1	e1	e1	e1
I2	e20	e20	e20	e20	r1	e20	e20	r1	r1	e1	e1	e1	e1	e1
I3	s6	e6	e7	e4	s7	e4	e7	e6	e8	e1	e15	e5	4	5
I4	e10	e6	s9	e11	e10	e11	s8	e6	e8	e1	e1	e1	e1	e1
I5	e6	r6	r6	e4	e11	e4	r6	e6	e8	e1	e1	e1	e1	e1
I6	s6	e12	e12	e6	s7	e6	e6	e6	e8	e1	e1	e1	10	11
I7	e6	r8	r8	e6	e6	e6	r8	e6	e8	e1	e1	e1	e1	e1
I8	e6	e6	e6	e9	s13	e9	e6	e6	e8	e1	e15	12	e14	e14
I9	s6	e13	e13	e6	s7	e6	e6	e6	e8	e1	e15	e14	14	5
I10	e6	s15	s9	e6	e6	e6	e6	e6	e8	e1	e15	e14	e14	e14
I11	e6	s16/r6	r6	e6	e6	e6	r6	e6	e8	e1	e15	e1	e1	e1
I12	e18	e18	e18	e18	e18	e18	e18	s17	e8	e1	e1	e1	e1	e1
I13	e17	e17	e17	s3	e17	s2	e17	e17	e8	e1	18	e1	e1	e1
I14	e6	r5	s9/r5	e6	e6	e6	r5	e6	e8	e1	e15	e1	e1	e1
I15	e6	r7	r7	e6	e6	e6	r7	e6	e8	e1	e1	e1	e1	e1
I16	e6	r9	r9	e6	e6	e6	r9	e6	e8	e1	e1	e1	e1	e1
I17	e22	e22	e22	e22	r2	e22	e22	r2	r2	e1	e1	e1	e1	e1
I18	e19	e19	e19	e21	s13	e21	e19	r3	e8	e1	e16	19	e14	e14
I19	e1	e1	e1	e1	e1	e1	e1	r4	e8	e1	e1	e1	e1	e1

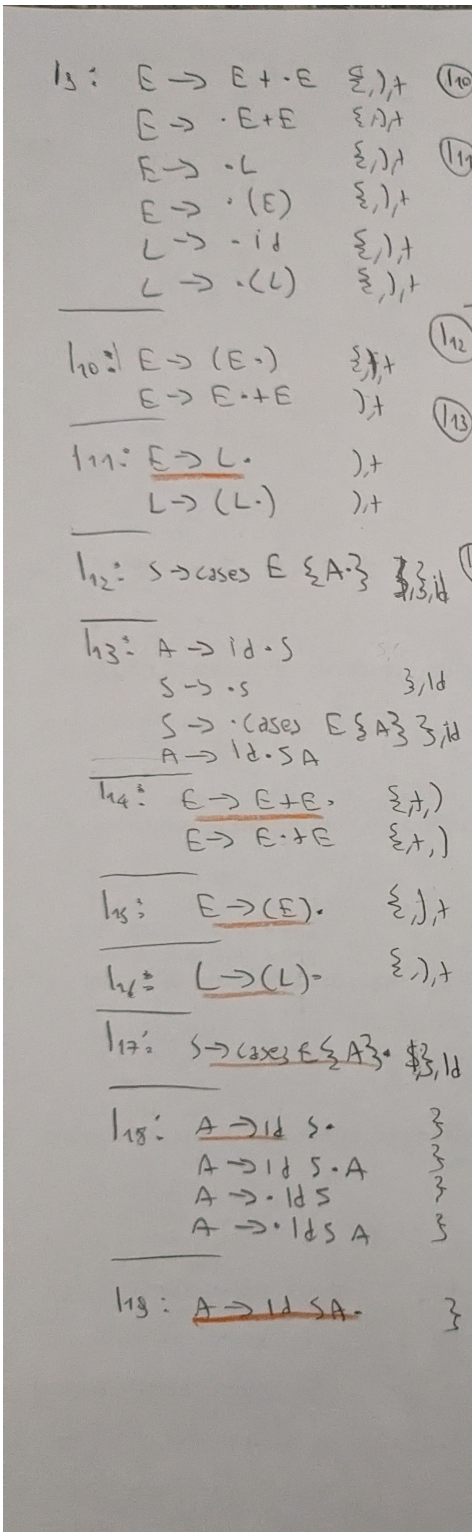
Notiamo due conflitti shift/reduce (segnati in giallo nella tabella action).

e0	print "empty program"; panic
e1	print "internal error"; panic
e2	print "action outside of case"; drop
e3	print "expression outside of case"; drop
e4	print "double statement"; skip
e5	print "cannot use outside of case"; goto I4
e6	print "unexpected {lookahead}"; skip
e7	print "missing expression"; push E; goto I4
e8	print "incomplete program"; panic
e9	print "missing id"; goto I13
e10	print "missing +"; add +
e11	print "missing }"; add }
e12	print "missing expression"; push E; goto I10
e13	print "missing expression"; push E; goto I14
e14	print "unexpected {TOS}"; drop
e15	print "double statement"; drop
e16	print "missing id"; pop X; push id; push I13; push X: goto I18
e17	print "unexpected {lookahead}"; skip until s or cases
e18	print "unexpected {lookahead}"; skip until }
e19	print "unexpected {lookahead}"; skip until id or }
e20	print "unexpected {lookahead}, accepting anyway"; reduce r1
e21	print "missing id"; push id; push I13
e22	print "unexpected {lookahead}, accepting anyway"; reduce r2

LALR

Adesso per trovare il LALR parto dagli stati della collezione canonica ed aggiungo i lookahead. Per farlo ripercorro gli stati e la funzione GOTO calcolati precedentemente, propagando i lookahead. (Li scrivo man mano a destra degli item già trovati, per non dover ricopiare tutto in una tabella).





Da questi posso calcolare la tabella LALR:

	ACTION									GOTO				
	(	**)**	**+**	**cases**	**id**	**s**	**{**	**}**	**$**	**S'**	**S**	**A**	**E**	**L**
I0				s3		s2					1			
I1									acc					
I2					r1			r1	r1					
I3	s6				s7								4	5
I4			s9				s8							
I5		r6	r6				r6							
I6	s6				s7								10	11
I7		r8	r8				r8							
I8					s13							12		
I9	s6				s7								14	5
I10		s15	s9											
I11		s16/r6	r6				e6							
I12								s17						
I13				s3		s2					18			
I14		r5	s9/r5				r5							
I15		r7	r7				r7							
I16		r9	r9				r9							
I17					r2			r2	r2					
I18					s13			r3				19		
I19								r4						

Notiamo che è identica in tutto (inclusi i conflitti shift/reduce) tranne che nella casella I11-{ (in rosso). È normale perché SLR e LALR hanno sempre gli stessi shift, però il LALR *può* non avere alcuni reduce (quindi potenzialmente avendo meno conflitti e riconoscendo più linguaggi).

Quindi decido di utilizzare la stessa gestione degli errori del caso SLR, modificando solo quella cella.

Esecuzione SLR

Parto eseguendo il parser SLR:

0	cases id + id { id s cases id id) { id s } s } $	s3
0 cases 3	id + id { id s cases id id) { id s } s } $	s7
0 cases 3 id 7	+ id { id s cases id id) { id s } s } $	L->id
0 cases 3 L 5	+ id { id s cases id id) { id s } s } $	E->L
0 cases 3 E 4	+ id { id s cases id id) { id s } s } $	s9
0 cases 3 E 4 + 9	id { id s cases id id) { id s } s } $	s7
0 cases 3 E 4 + 9 id 7	{ id s cases id id) { id s } s } $	L->id
0 cases 3 E 4 + 9 L 5	{ id s cases id id) { id s } s } $	E->L
0 cases 3 E 4 + 9 E 14	{ id s cases id id) { id s } s } $	E->E+E
0 cases 3 E 4	{ id s cases id id) { id s } s } $	s8
0 cases 3 E 4 { 8	id s cases id id) { id s } s } $	s13
0 cases 3 E 4 { 8 id 13	s cases id id) { id s } s } $	s2
0 cases 3 E 4 { 8 id 13 s 2	cases id id) { id s } s } $	unexpected cases, accepting anyway, S->s
0 cases 3 E 4 { 8 id 13 S 18	cases id id) { id s } s } $	S->s
0 cases 3 E 4 { 8 id 13 S 18	cases id id) { id s } s } $	missing id
0 cases 3 E 4 { 8 id 13 S 18 id 13	cases id id) { id s } s } $	s3
0 cases 3 E 4 { 8 id 13 S 18 id 13 cases 3	id id) { id s } s } $	s7
0 cases 3 E 4 { 8 id 13 S 18 id 13 cases 3 id 7	id) { id s } s } $	unexpected id
0 cases 3 E 4 { 8 id 13 S 18 id 13 cases 3 id 7	) { id s } s } $	L->id
0 cases 3 E 4 { 8 id 13 S 18 id 13 cases 3 L 5	) { id s } s } $	E->L
0 cases 3 E 4 { 8 id 13 S 18 id 13 cases 3 E 4	) { id s } s } $	unexpected)
0 cases 3 E 4 { 8 id 13 S 18 id 13 cases 3 E 4	{ id s } s } $	s8
0 cases 3 E 4 { 8 id 13 S 18 id 13 cases 3 E 4 { 8	id s } s } $	s13
0 cases 3 E 4 { 8 id 13 S 18 id 13 cases 3 E 4 { 8 id 13	s } s } $	s2
0 cases 3 E 4 { 8 id 13 S 18 id 13 cases 3 E 4 { 8 id 13 s 2	} s } $	S->s
0 cases 3 E 4 { 8 id 13 S 18 id 13 cases 3 E 4 { 8 id 13 S 18	} s } $	A->id S
0 cases 3 E 4 { 8 id 13 S 18 id 13 cases 3 E 4 { 8 A 12	} s } $	s17
0 cases 3 E 4 { 8 id 13 S 18 id 13 cases 3 E 4 { 8 A 12 } 17	s } $	unexpected cases, accepting

		anyway, S->cases E { A }
0 cases 3 E 4 { 8 id 13 S 18 id 13 S 18	s } $	missing id
0 cases 3 E 4 { 8 id 13 S 18 id 13 S 18 id 13	s } $	s2
0 cases 3 E 4 { 8 id 13 S 18 id 13 S 18 id 13 s 2	} $	S->s
0 cases 3 E 4 { 8 id 13 S 18 id 13 S 18 id 13 S 18	} $	A->id S
0 cases 3 E 4 { 8 id 13 S 18 id 13 S 18 A 19	} $	A->id S A
0 cases 3 E 4 { 8 id 13 S 18 A 19	} $	A->id S A
0 cases 3 E 4 { 8 A 12	} $	s17
0 cases 3 E 4 { 8 A 12 } 17	$	S->cases E { A }
0 S 1	$	Acc

Esecuzione LALR

Visto che le due tabelle differiscono solo nello stato 11, e che durante il parsing non ho incontrato quello stato. Il parser LALR produce esattamente la stessa esecuzione.