Corso di linguaggi e compilatori

Relazione – Parte 3 – Gruppo 2

Giovanni D'Agostino, Beatrice Portelli

17 Aprile 2019

Esercizio 1

1) Grammatica LL(1), FIRST e FOLLOW

La grammatica LL(1) da noi definita equivalente alla grammatica L data è la seguente:

L → N M
M → ++L | ε
N → [O | E:N
O →] | E P
P → , LL | ..E]
LL →] | E,LL
E → id | num

FIRST(L) = FIRST(N) = { [, id, num }
FIRST(M) = { ++, ε }
FIRST(O) = FIRST(LL) = {], id , num }
FIRST(P) = { , , .. } (la virgola in rosso è un token della grammatica)
FIRST(E) = { id, num }

FOLLOW(L) = FOLLOW(M) = {$}
FOLLOW(N) = FOLLOW(O) = FOLLOW (P) = FOLLOW(LL) = {++, $}
FOLLOW(E) = { :, .., , ,] }

2) Tabella di parsing ed error recovery

	id	num	:	++	..	,	[	]	$
L	L → NM	L → NM	e1	e2	e2	e2	L → NM	e3	s1
M	e4	e4	e5	M → ++L	e6	e7	e8	e9	M → ε
N	N → E:N	N → E:N	e10	s2	e6	e7	N → [O	e3	s3
O	O → EP	O → EP	e5	s2	e1	e1	e11	O →]	s3
P	e13	e13	e5	s2	P → ..E]	P → ,LL	e12	e13	s3
LL	LL → E,LL	LL → E,LL	e5	s2	e6	e14	e12	LL →]	s3
E	E → id	E → num	e1	e15	s4	s5	e12	e16	s3

Errori:

- per gli errori e1 ed e16 abbiamo scelto arbitrariamente di inserire un id, sarebbe stato possibile inserire un num
- nell'esecuzione del parsing sull'input dato abbiamo usato la convenzione (spiegata nel libro di testo, capitolo 4.5.5) di effettuare un pop sullo stack nel caso che il token al top of stack non corrisponda al simbolo in input, dare il messaggio che il il terminale al top of stack è stato inserito e poi riprendere il parsing
- gli errori che cominciano con s sono stati ricavati tramite l'uso del synchronizing set

e1: echo "missing id or num before operator" insert id
e2: echo "input starting with unexpected token" skip
e3: echo "missing [here" insert [
e4: echo "unexpected id or num found" skip
e5: echo "misuse of : here" skip
e6: echo "misuse of .. here" skip
e7: echo "unexpected , found" skip
e8: echo "missing ++ operator before [" insert ++
e9: echo "duplicated] found" skip
e10: echo "duplicated : found" skip
e11: echo "duplicated [found" skip
e12: echo "unexpected [found" skip
e13: echo "missing , here" insert ,
e14: echo "duplicated , found" skip
e15: echo "unexpected ++ found" skip
e16: echo "missing id or num before]" insert id

s1: echo "unexpected eof" pop push M
s2: echo "unexpected ++ found" pop
s3: echo "unexpected eof" pop until M
s4: echo "misuse of .. inside a list" pop
s5: echo "misuse of , inside a list" pop

3) Esecuzione dell'input

STACK	INPUT	ACTION	MATCHED
L$	[[id..[num]++id,num]:[id id]$	L → NM	
NM$	[[id..[num]++id,num]:[id id]$	N→[O	
[OM$	[[id..[num]++id,num]:[id id]$	match [	[
OM$	[id..[num]++id,num]:[id id]$	e11 (skip)	[
OM$	id..[num]++id,num]:[id id]$	O→EP	[
EPM$	id..[num]++id,num]:[id id]$	E→id	[
id PM$	id..[num]++id,num]:[id id]$	match id	[id
PM$	..[num]++id,num]:[id id]$	P→..E]	[id
..E]M$	..[num]++id,num]:[id id]$	match ..	[id ..
E]M$	[num]++id,num]:[id id]$	e12 (skip)	[id ..
E]M$	num]++id,num]:[id id]$	E→num	[id ..
num]M$	num]++id,num]:[id id]$	match num	[id .. num
]M$	]++id,num]:[id id]$	match]	[id .. num]

M$	++id,num]:[id id]$	M→++L	[id .. num]
++L$	++id,num]:[id id]$	match ++	[id .. num] ++
L$	id,num]:[id id]$	L→NM	[id .. num] ++
NM$	id,num]:[id id]$	N→E:N	[id .. num] ++
E:NM$	id,num]:[id id]$	E→id	[id .. num] ++
id:NM$	id,num]:[id id]$	match id	[id .. num] ++ id
:NM$	,num]:[id id]$	pop ": inserted"	[id .. num] ++ id :
NM$	,num]:[id id]$	e7 (skip)	[id .. num] ++ id :
NM$	num]:[id id]$	N→E:N	[id .. num] ++ id :
E:NM$	num]:[id id]$	E→num	[id .. num] ++ id :
num:NM$	num]:[id id]$	match num	[id .. num] ++ id : num
:NM$	]:[id id]$	pop ": inserted"	[id .. num] ++ id : num :
NM$	]:[id id]$	e3 (insert [)	[id .. num] ++ id : num :
NM$	[]:[id id]$	N→[O	[id .. num] ++ id : num :
[OM$	[]:[id id]$	match [	[id .. num] ++ id : num : [
OM$	]:[id id]$	O→]	[id .. num] ++ id : num : [
]M$	]:[id id]$	match]	[id .. num] ++ id : num : []
M$	:[id id]$	e5 (skip)	[id .. num] ++ id : num : []
M$	[id id]$	e8 (insert ++)	[id .. num] ++ id : num : []
M$	++[id id]$	M→++L	[id .. num] ++ id : num : []
++L$	++[id id]$	match ++	[id .. num] ++ id : num : [] ++
L$	[id id]$	L → NM	[id .. num] ++ id : num : [] ++
NM$	[id id]$	N→[O	[id .. num] ++ id : num : [] ++
[OM$	[id id]$	match [	[id .. num] ++ id : num : [] ++ [
OM$	id id]$	O→EP	[id .. num] ++ id : num : [] ++ [
EPM$	id id]$	E→id	[id .. num] ++ id : num : [] ++ [
idPM$	id id]$	match id	[id .. num] ++ id : num : [] ++ [id
PM$	id]$	e13 (insert ,)	[id .. num] ++ id : num : [] ++ [id
PM$	,id]$	P→,LL	[id .. num] ++ id : num : [] ++ [id
,LLM$	,id]$	match ,	[id .. num] ++ id : num : [] ++ [id,
LLM$	id]$	LL→E,LL	[id .. num] ++ id : num : [] ++ [id,
E,LLM$	id]$	E→id	[id .. num] ++ id : num : [] ++ [id,
id,LLM$	id]$	match id	[id .. num] ++ id : num : [] ++ [id,id
,LLM$	]$	pop ", inserted"	[id .. num] ++ id : num : [] ++ [id,id,
LLM$	]$	LL→]	[id .. num] ++ id : num : [] ++ [id,id,
]M$	]$	match]	[id .. num] ++ id : num : [] ++ [id,id,]
M$	$	M→ε	[id .. num] ++ id : num : [] ++ [id,id,]
$	$	OK	[id .. num] ++ id : num : [] ++ [id,id,]

Esercizio 2

1) FIRST e FOLLOW

N.B. Nella grammatica abbiamo introdotto il nuovo simbolo iniziale C' e la produzione C' → C

FIRST(C') = FIRST(C) = {pred, num, id, (}
FIRST(E) = FIRST(Es) = {num, id, (}

FOLLOW(C') = {$}
FOLLOW(C) = {$, &}
FOLLOW(E) = {rel, +, *,), , , $, & }
FOLLOW(Es) = {)}

2) Item LR(0)

I 0
C' → • C
C → • E rel E
C → • C & C
C → • pred (Es)
E → • num
E → • id
E → • E + E
E → • E * E
E → • (E)

I 1
C' → C •
C → C • & C

I 2
C → E • rel E
E → E • + E
E → E • * E

I 3
C → pred • (Es)

I 4
E → num •

I 5
E → id •

I 6
E → (• E)
E → • num
E → • id
E → • E + E
E → • E * E
E → • (E)

I 7
C → C & • C
C → • E rel E
C → • C & C
C → • pred (Es)
E → • num
E → • id
E → • E + E
E → • E * E
E → • (E)

I 8
C → E rel • E
E → • num
E → • id
E → • E + E
E → • E * E
E → • (E)

I 9
E → E + • E
E → • num
E → • id
E → • E + E
E → • E * E
E → • (E)

I 10
E → E * • E
E → • num
E → • id
E → • E + E
E → • E * E
E → • (E)

I 11
C → pred (• Es)
Es → • E
Es → • E , Es
E → • num
E → • id
E → • E + E
E → • E * E
E → • (E)

I 12
E → (E •)
E → E • + E
E → E • * E

I 13
C → C & C •
C → C • & C

I 14
C → E rel E •
E → E • + E
E → E • * E

I 15
E → E + E •
E → E • + E
E → E • * E

I 16
E → E * E •
E → E • + E
E → E • * E

I 17
C → pred (Es •)

I 18
Es → E •
Es → E • , Es
E → E • + E
E → E • * E

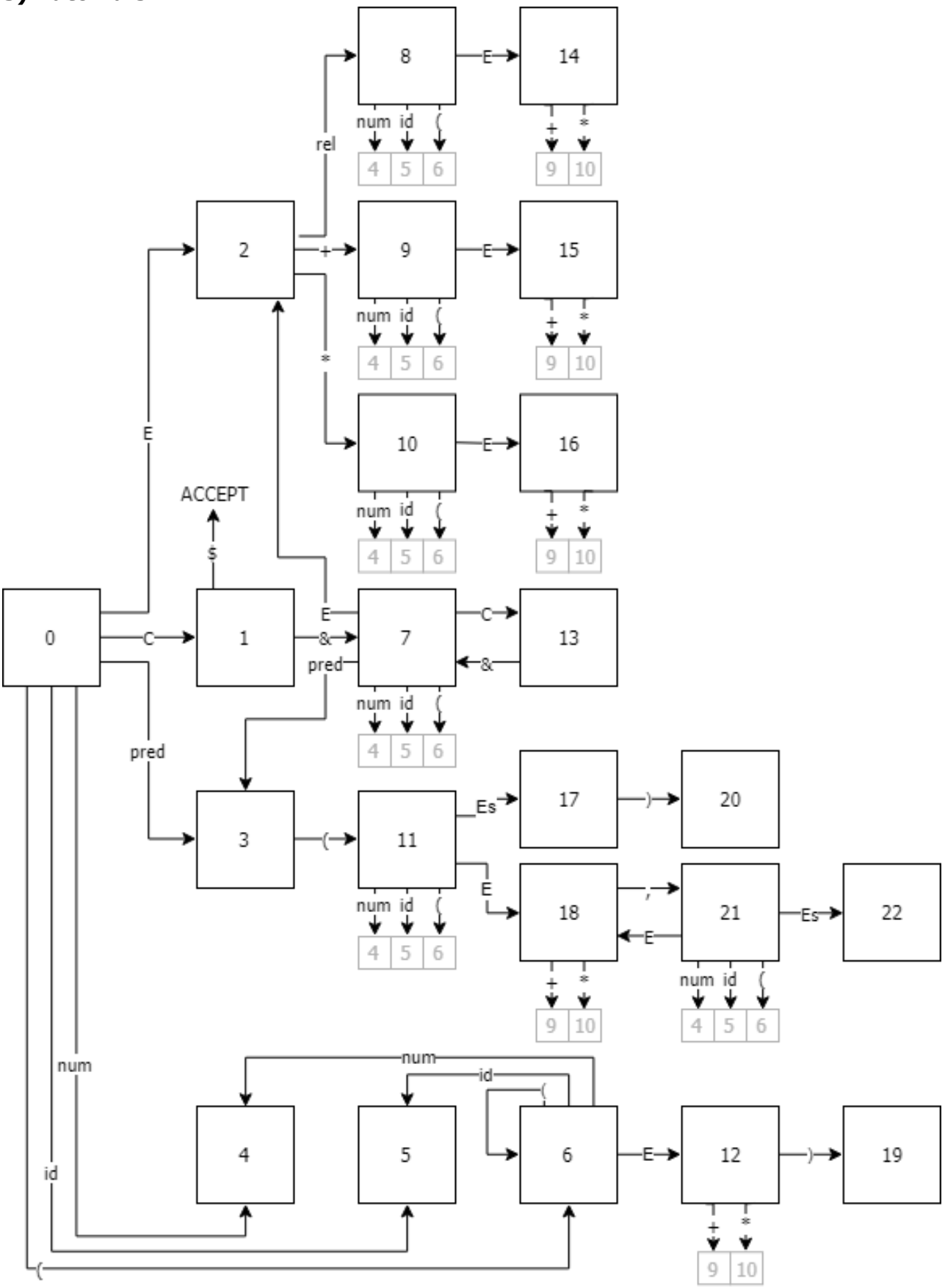
I 19
E → (E) •

I 20
C → pred (Es) •

I 21
Es → E , • Es
Es → • E
Es → • E , Es
E → • num
E → • id
E → • E + E
E → • E * E
E → • (E)

I 22
Es → E , Es •

3) Automa SLR


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
ACCEPT
$
E
C
&
pred
rel
+
*
(
)
Es
,
num
id
num id (
+ *
4 5 6
9 10

4) Tabella di Parsing SLR

STATI	ACTION											GOTO		
	*	+	rel	&	pred	(	)	num	id	,	$	C	E	Es
0	e1	e1	e1	e2	S3	S6	e2	S4	S5	e2	e3	1	2	
1	i1	i1	i1	S7	i1	i1	i1	i1	i1	i1	ACC			
2	S10	S9	S8	e8	i1	i1	e6	i1	i1	e7	e12			
3	e13	e13	e9	e13	e14	S11	e6	e15	e15	e7	e12			
4	R4	R4	R4	R4	e5	e10	R4	e11	e11	R4	R4			
5	R5	R5	R5	R5	e5	e10	R5	e11	e11	R5	R5			
6	e1	e1	e1	e16	e17	S6	e18	S4	S5	e7	e12		12	
7	e1	e1	e1	e14	S3	S6	e6	S4	S5	e7	e12	13	2	
8	e1	e1	e14	e1	e17	S6	e6	S4	S5	e7	e19		14	
9	e1	e14	e1	e1	e17	S6	e6	S4	S5	e7	e19		15	
10	e14	e1	e1	e1	e17	S6	e6	S4	S5	e7	e19		16	
11	e1	e1	e1	e16	e17	S6	e18	S4	S5	e18	e12		18	17
12	S10	S9	e20	e8	i1	i1	S19	i1	i1	e7	e20			
13	i1	i1	i1	**R2**/S7	i1	i1	i1	i1	i1	i1	R2			
14	S10	S9	e9	R1	i1	i1	e6	i1	i1	e7	R1			
15	R6/**S10**	**R6**/S9	R6	R6	i1	i1	R6	i1	i1	R6	R6			
16	**R7**/S10	**R7**/S9	R7	R7	i1	i1	R7	i1	i1	R7	R7			
17	i1	i1	i1	i1	i1	i1	S20	i1	i1	i1	i1			
18	S10	S9	e20	e20	i1	i1	R9	i1	i1	S21	e20			
19	R8	R8	R8	R8	e5	e10	R8	e11	e11	R8	R8			
20	e4	e4	e4	R3	e22	e22	e21	e22	e22	e7	R3			
21	e1	e1	e9	e16	e17	S6	e18	S4	S5	e21	e19		18	22
22	i1	i1	i1	i1	i1	i1	R10	i1	i1	i1	i1			

i1: echo "the impossible happened" internal error halt

e1: echo "missing expression before operator" insert id
e2: echo "file starting with unexpected symbol" skip
e3: echo "empty input" pop until 0 push 1
e4: echo "misuse of operator: left operand must be an expression" skip
e5: echo "unexpected pred here" skip
e6: echo "unexpected) here" skip
e7: echo "unexpected , here" skip
e8: echo "misuse of & here: incorrect left operand" skip
e9: echo "unexpected rel here" skip
e10: echo "unexpected (here" skip
e11: echo "missing algebraic operator here" insert *
e12: echo "unexpected eof" pop until 0 push 1
e13: echo "unexpected operator here" skip
e14: echo "repeated operator here" skip
e15: echo "missing (here" insert (
e16: echo "unexpected & here" skip
e17: echo "expected expression found pred" pop until 0
e18: echo "missing expression before separator" insert id
e19: echo "missing expression before eof" insert id
e20: echo "missing) here" insert)
e21: echo "repeated separator here" skip
e22: echo "missing & here" insert &

5) Conflitti shift/reduce

N.B. Nei conflitti shift/reduce della matrice di parsing sono state riportate in grassetto le action scelte, mentre in caratteri più piccoli le action scartate.

- ACTION[13, &]: abbiamo scelto di effettuare un reduce per implementare l'associatività a sinistra dell'operatore & nella regola C → C & C
- ACTION[15, *]: abbiamo scelto di effettuare uno shift per implementare la precedenza dell'operatore * sull'operatore +
- ACTION[15, +]: abbiamo scelto di effettuare un reduce per implementare l'associatività a sinistra dell'operatore + nella regola E → E + E
- ACTION[16, *]: abbiamo scelto di effettuare un reduce per implementare l'associatività a sinistra dell'operatore * nella regola E → E * E
- ACTION[16, +]: abbiamo scelto di effettuare un reduce per implementare la precedenza dell'operatore * sull'operatore +

6) Esecuzione dell'input (SLR)

STACK	SIMBOLI	INPUT	ACTION
0	$	num * id rel id & pred (num , num id) & id id rel num $	s4
0 4	$ num	* id rel id & pred (num , num id) & id id rel num $	r4 E -> num
0 2	$ E	* id rel id & pred (num , num id) & id id rel num $	s10
0 2 10	$ E *	id rel id & pred (num , num id) & id id rel num $	s5
0 2 10 5	$ E * id	rel id & pred (num , num id) & id id rel num $	r5 E -> id
0 2 10 16	$ E * E	rel id & pred (num , num id) & id id rel num $	r7 E -> E * E
0 2	$ E	rel id & pred (num , num id) & id id rel num $	s8
0 2 8	$ E rel	id & pred (num , num id) & id id rel num $	s5
0 2 8 5	$ E rel id	& pred (num , num id) & id id rel num $	r5 E -> id
0 2 8 14	$ E rel E	& pred (num , num id) & id id rel num $	r1 C -> E rel E
0 1	$ C	& pred (num , num id) & id id rel num $	s7
0 1 7	$ C &	pred (num , num id) & id id rel num $	s3
0 1 7 3	$ C & pred	(num , num id) & id id rel num $	s11
0 1 7 3 11	$ C & pred (	num , num id) & id id rel num $	s4
0 1 7 3 11 4	$ C & pred (num	, num id) & id id rel num $	r4 E -> num
0 1 7 3 11 18	$ C & pred (E	, num id) & id id rel num $	s21
0 1 7 3 11 18 21	$ C & pred (E ,	num id) & id id rel num $	s4
0 1 7 3 11 18 21 4	$ C & pred (E , num	id) & id id rel num $	e11 (insert *)
0 1 7 3 11 18 21 4	$ C & pred (E , num	* id) & id id rel num $	r4 E -> num
0 1 7 3 11 18 21 18	$ C & pred (E , E	* id) & id id rel num $	s10
0 1 7 3 11 18 21 18 10	$ C & pred (E , E *	id) & id id rel num $	s5
0 1 7 3 11 18 21 18 10 5	$ C & pred (E , E * id	) & id id rel num $	r5 E -> id
0 1 7 3 11 18 21 18 10 16	$ C & pred (E , E * E	) & id id rel num $	r7 E -> E * E
0 1 7 3 11 18 21 18	$ C & pred (E , E	) & id id rel num $	r9 Es -> E
0 1 7 3 11 18 21 22	$ C & pred (E , Es	) & id id rel num $	r10 Es -> E , Es
0 1 7 3 11 17	$ C & pred (Es	) & id id rel num $	s20
0 1 7 3 11 17 20	$ C & pred (Es)	& id id rel num $	r3 C -> pred (Es)

0 1 7 13	$ C & C	& id id rel num $	r2 C -> C & C
0 1	$ C	& id id rel num $	s7
0 1 7	$ C &	id id rel num $	s5
0 1 7 5	$ C & id	id rel num $	e11 (insert *)
0 1 7 5	$ C & id	* id rel num $	r5 E -> id
0 1 7 2	$ C & E	* id rel num $	s10
0 1 7 2 10	$ C & E *	id rel num $	s5
0 1 7 2 10 5	$ C & E * id	rel num $	r5 E -> id
0 1 7 2 10 16	$ C & E * E	rel num $	r7 E -> E * E
0 1 7 2	$ C & E	rel num $	s8
0 1 7 2 8	$ C & E rel	num $	s4
0 1 7 2 8 4	$ C & E rel num	$	r4 E -> num
0 1 7 2 8 14	$ C & E rel E	$	r1 C -> E rel E
0 1 7 13	$ C & C	$	r2 C -> C & C
0 1	$ C	$	ACC

7) Item LR(1)

I 0	
C' → • C	$
C → • E rel E	$ / &
C → • C & C	$ / &
C → • pred (Es)	$ / &
E → • num	rel / + / *
E → • id	rel / + / *
E → • E + E	rel / + / *
E → • E * E	rel / + / *
E → • (E)	rel / + / *

I 1	
C' → C •	$
C → C • & C	& / $

I 2	
C → E • rel E	& / $
E → E • + E	rel / + / *
E → E • * E	rel / + / *

I 3	
C → pred • (Es)	& / $

I 4	
E → num •	rel / + / * /) / , / & / $

I 5	
E → id •	rel / + / * /) / , / & / $

I 6	
E → (• E)	rel / + / * /) / , / & / $
E → • num	rel / + / * /) / , / & / $
E → • id	rel / + / * /) / , / & / $
E → • E + E	rel / + / * /) / , / & / $
E → • E * E	rel / + / * /) / , / & / $
E → • (E)	rel / + / * /) / , / & / $

I 7	
C → C & • C	& / $
C → • E rel E	& / $
C → • C & C	& / $
C → • pred (Es)	& / $
E → • num	rel / + / *
E → • id	rel / + / *
E → • E + E	rel / + / *
E → • E * E	rel / + / *
E → • (E)	rel / + / *

I 8	
C → E rel • E	& / $
E → • num	+ / * / & / $
E → • id	+ / * / & / $
E → • E + E	+ / * / & / $
E → • E * E	+ / * / & / $
E → • (E)	+ / * / & / $

I 9	
E → E + • E	rel / + / * /) / , / & / $
E → • num	rel / + / * /) / , / & / $
E → • id	rel / + / * /) / , / & / $
E → • E + E	rel / + / * /) / , / & / $
E → • E * E	rel / + / * /) / , / & / $
E → • (E)	rel / + / * /) / , / & / $

I 10	
E → E * • E	rel / + / * /) / , / & / $
E → • num	rel / + / * /) / , / & / $
E → • id	rel / + / * /) / , / & / $
E → • E + E	rel / + / * /) / , / & / $
E → • E * E	rel / + / * /) / , / & / $
E → • (E)	rel / + / * /) / , / & / $

I 11	
C → pred (• Es)	& / $
Es → • E	)
Es → • E , Es	)
E → • num	) / , / + / *
E → • id	) / , / + / *
E → • E + E	) / , / + / *
E → • E * E	) / , / + / *
E → • (E)	) / , / + / *

I 12	
E → (E •)	rel / + / * /) / , / & / $
E → E • + E	) / + / *
E → E • * E	) / + / *

I 13	
C → C & C •	& / $
C → C • & C	& / $

I 14	
C → E rel E •	& / $
E → E • + E	* / + / & / $
E → E • * E	* / + / & / $

I 15	
E → E + E •	rel / + / * /) / , / & / $
E → E • + E	rel / + / * /) / , / & / $
E → E • * E	rel / + / * /) / , / & / $

I 16	
E → E * E •	rel / + / * /) / , / & / $
E → E • + E	rel / + / * /) / , / & / $
E → E • * E	rel / + / * /) / , / & / $

I 17	
C → pred (Es •)	& / $

I 18	
Es → E •	)
Es → E • , Es	)
E → E • + E	) / , / + / *
E → E • * E	) / , / + / *

I 19	
E → (E) •	rel / + / * /) / , / & / $

I 20	
C → pred (Es) •	& / $

I 21	
Es → E , • Es	)
Es → • E	)
Es → • E , Es	)
E → • num	) / , / + / *
E → • id	) / , / + / *
E → • E + E	) / , / + / *
E → • E * E	) / , / + / *
E → • (E)	) / , / + / *

I 22	
Es → E , Es •	)

8) Tabella Parsing LALR

STATI	ACTION											GOTO		
	*	+	rel	&	pred	(	)	num	id	,	$	C	E	Es
0	e1	e1	e1	e2	S3	S6	e2	S4	S5	e2	e3	1	2	
1	i1	i1	i1	S7	i1	i1	i1	i1	i1	i1	ACC			
2	S10	S9	S8	e8	i1	i1	e6	i1	i1	e7	e12			
3	e13	e13	e9	e13	e14	S11	e6	e15	e15	e7	e12			
4	R4	R4	R4	R4	e5	e10	R4	e11	e11	R4	R4			
5	R5	R5	R5	R5	e5	e10	R5	e11	e11	R5	R5			
6	e1	e1	**e9**	e16	**e5**	S6	e18	S4	S5	e7	e12		12	
7	e1	e1	e1	e14	S3	S6	e6	S4	S5	e7	e12	13	2	
8	e1	e1	e14	e1	**e5**	S6	e6	S4	S5	e7	e19		14	
9	e1	e14	e1	e1	**e5**	S6	e6	S4	S5	e7	e19		15	
10	e14	e1	e1	e1	**e5**	S6	e6	S4	S5	e7	e19		16	
11	e1	e1	**e9**	e16	**e5**	S6	e18	S4	S5	e18	e12		18	17
12	S10	S9	**e9**	e8	i1	i1	S19	i1	i1	e7	e20			
13	i1	i1	i1	**R2**/S7	i1	i1	i1	i1	i1	i1	R2			
14	S10	S9	e9	R1	i1	i1	e6	i1	i1	e7	R1			
15	R6/**S10**	**R6**/S9	R6	R6	i1	i1	R6	i1	i1	R6	R6			
16	**R7**/S10	**R7**/S9	R7	R7	i1	i1	R7	i1	i1	R7	R7			
17	i1	i1	i1	i1	i1	i1	S20	i1	i1	i1	i1			
18	S10	S9	**e9**	e20	i1	i1	R9	i1	i1	S21	e20			
19	R8	R8	R8	R8	e5	e10	R8	e11	e11	R8	R8			
20	e4	e4	e4	R3	e22	e22	e21	e22	e22	e7	R3			
21	e1	e1	e9	e16	**e5**	S6	e18	S4	S5	e21	e19		18	22
22	i1	i1	i1	i1	i1	i1	R10	i1	i1	i1	i1			

Il testo degli errori e le azioni da loro effettuate sono gli stessi della tabella di parsing SLR.

La tabella è identica alla tabella di parsing SLR, fatta eccezione per gli errori segnati in grassetto: nel parser SLR alcuni dei potenziali errori generati dalla presenza di rel e pred erano stati rimandati, in questo caso si è scelto di gestirli subito, evitando di delegarli.

9) Esecuzione dell'input (LALR)

N.B Per questo input l'esecuzione del parsing LALR ha dato gli stessi risultati dell'esecuzione del parsing SLR.

STACK	SIMBOLI	INPUT	ACTION
0	$	num * id rel id & pred (num , num id) & id id rel num $	s4
0 4	$ num	* id rel id & pred (num , num id) & id id rel num $	r4 E -> num
0 2	$ E	* id rel id & pred (num , num id) & id id rel num $	s10
0 2 10	$ E *	id rel id & pred (num , num id) & id id rel num $	s5
0 2 10 5	$ E * id	rel id & pred (num , num id) & id id rel num $	r5 E -> id
0 2 10 16	$ E * E	rel id & pred (num , num id) & id id rel num $	r7 E -> E * E
0 2	$ E	rel id & pred (num , num id) & id id rel num $	s8
0 2 8	$ E rel	id & pred (num , num id) & id id rel num $	s5
0 2 8 5	$ E rel id	& pred (num , num id) & id id rel num $	r5 E -> id
0 2 8 14	$ E rel E	& pred (num , num id) & id id rel num $	r1 C -> E rel E
0 1	$ C	& pred (num , num id) & id id rel num $	s7
0 1 7	$ C &	pred (num , num id) & id id rel num $	s3
0 1 7 3	$ C & pred	(num , num id) & id id rel num $	s11
0 1 7 3 11	$ C & pred (	num , num id) & id id rel num $	s4
0 1 7 3 11 4	$ C & pred (num	, num id) & id id rel num $	r4 E -> num
0 1 7 3 11 18	$ C & pred (E	, num id) & id id rel num $	s21
0 1 7 3 11 18 21	$ C & pred (E ,	num id) & id id rel num $	s4
0 1 7 3 11 18 21 4	$ C & pred (E , num	id) & id id rel num $	e11 (insert *)
0 1 7 3 11 18 21 4	$ C & pred (E , num	* id) & id id rel num $	r4 E -> num
0 1 7 3 11 18 21 18	$ C & pred (E , E	* id) & id id rel num $	s10
0 1 7 3 11 18 21 18 10	$ C & pred (E , E *	id) & id id rel num $	s5
0 1 7 3 11 18 21 18 10 5	$ C & pred (E , E * id	) & id id rel num $	r5 E -> id
0 1 7 3 11 18 21 18 10 16	$ C & pred (E , E * E	) & id id rel num $	r7 E -> E * E
0 1 7 3 11 18 21 18	$ C & pred (E , E	) & id id rel num $	r9 Es -> E
0 1 7 3 11 18 21 22	$ C & pred (E , Es	) & id id rel num $	r10 Es -> E , Es
0 1 7 3 11 17	$ C & pred (Es	) & id id rel num $	s20

0 1 7 3 11 17 20	$ C & pred (Es)	& id id rel num $	r3 C -> pred (Es)
0 1 7 13	$ C & C	& id id rel num $	r2 C -> C & C
0 1	$ C	& id id rel num $	s7
0 1 7	$ C &	id id rel num $	s5
0 1 7 5	$ C & id	id rel num $	e11 (insert *)
0 1 7 5	$ C & id	* id rel num $	r5 E -> id
0 1 7 2	$ C & E	* id rel num $	s10
0 1 7 2 10	$ C & E *	id rel num $	s5
0 1 7 2 10 5	$ C & E * id	rel num $	r5 E -> id
0 1 7 2 10 16	$ C & E * E	rel num $	r7 E -> E * E
0 1 7 2	$ C & E	rel num $	s8
0 1 7 2 8	$ C & E rel	num $	s4
0 1 7 2 8 4	$ C & E rel num	$	r4 E -> num
0 1 7 2 8 14	$ C & E rel E	$	r1 C -> E rel E
0 1 7 13	$ C & C	$	r2 C -> C & C
0 1	$ C	$	ACC