CS480
Translators

LALR(1) Parsing/Ambiguous Grammars

Finish Chap. 4

Odds and Ends

~~while~~ ~~strings~~ ~~ops~~

- Test Friday, 3/7
- Milestone 4??? – Sunday night
  -if, stdout
- Dr. Stevenson speaks 3/3, Monday

int -> float for float arith,

anything but ints & floats

Extra credit +5 typechecking, e.g.

[+ string float]

Try this – Quiz 8…

- Determine if the Grammar is SLR(1) or LR(1)

1 S-> T **else** F;

2 T-> E

3 T-> **i;**

4 F->E

5 E-> E**+i**

6 E-> **i**

I_0 S' → •S , \$ I_1

S → •T else F; , \$ I_2

T → •E , else I_3

T → •i; , else I_4

E → •E+i , else I_3

E → •i , else I_4

E → •E+i , + I_3

E → •i , + I_4

- Construct a parse table for the grammar
- Add 2 questions for Dr. Stevenson on languages, compilers, or AI

LR(1) Parse...

S' –> S

1. S-> T **else** F**;**
2. T-> E
3. T-> **i;**
4. F->E
5. E-> E **+ i**
6. E-> **i**

Configurating set	***Successor***	***Configurating set***	***Successor***
I0: S' –> •S, $	I1	I6: E –> E+•i, else/+	I11
S –> •T else F;, $	I2	I7: T –> i;•, else	Reduce 3
T –> •E, else	I3	I8: S –> T else F•;, $	I12
T –> •i;, else	I4	I9: F –> E•, ;	Reduce 4
E –> •E+i, else/+	I3	E–> E•+i , ;/+	I13
E –> •i, else/+	I4	I10: E –> i•, ;/+	Reduce 6
I1: S' –> S•, $	Accept	I11: E –> E+i•, else/+	Reduce 5
I2: S –> T• else F;, $	I5	I12: S –> T else F;•, $	Reduce 1
I3: T –> E•, else	Reduce 2	I13: E –> E+•i, ;/+	I14
E –> E•+i, else/+	I6	I14: E –> E+i•, ;/+	Reduce 5
I4: T –> i•;, else	I7		
E –> i•, else/+	Reduce 6		
I5: S –> T else •F;, $	I8		
F –> •E , ;	I9		
E–> •E+i , ;/+	I9		
E –> •id , ;/+	I10		

LR(1) Parse Table…

Stack	i	;	else	+	$	S	T	F	E
0	s4					1	2		3
1					A				
2			s5						
3			R2	s6					
4		s7	r6	r6					
5	s10							8	9
6	s11								
7			r3						
8		s12							
9		r4		s6					
10		r6		r6					
11		r5	r5	r5					
12					r1				
13	s14								
14		r5		r5					

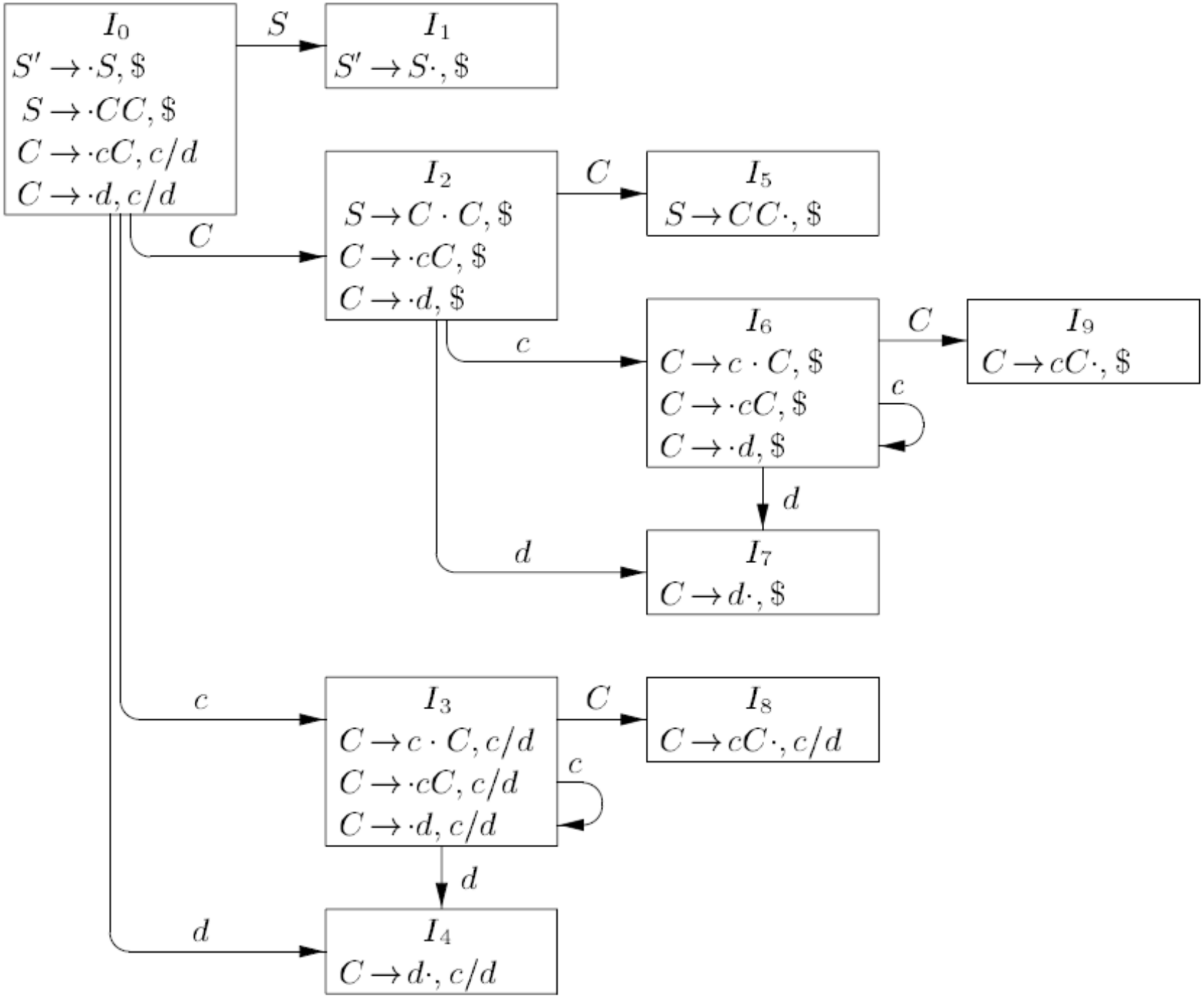
LR(1) Conditions

1. For any item in the set [A –> $\underline{u}$•x$\underline{v}$, a] with x as a terminal, there is no item in the set of the form [B –> $\underline{v}$•, x]. In the action table, this translates no shift-reduce conflict for any state. The successor function for x either shifts to a new state or reduces, but not both.

2. The lookaheads for all complete items within the set must be disjoint, e.g. set cannot have both [A –> $\underline{u}$•, a] and [B –> $\underline{v}$•, a] This translates to no reduce-reduce conflict on any state. If more than one non-terminal could be reduced from this set, it must be possible to uniquely determine which is appropriate from the next input token.

LR(1) vs. LALR(1)

- LR(1) more powerful
- LALR(1) has less states



I_0
$S' \to \cdot S, \$$
$S \to \cdot CC, \$$
$C \to \cdot cC, c/d$
$C \to \cdot d, c/d$

I_1
$S' \to S\cdot, \$$

I_2
$S \to C \cdot C, \$$
$C \to \cdot cC, \$$
$C \to \cdot d, \$$

I_5
$S \to CC\cdot, \$$

I_6
$C \to c \cdot C, \$$
$C \to \cdot cC, \$$
$C \to \cdot d, \$$

I_9
$C \to cC\cdot, \$$

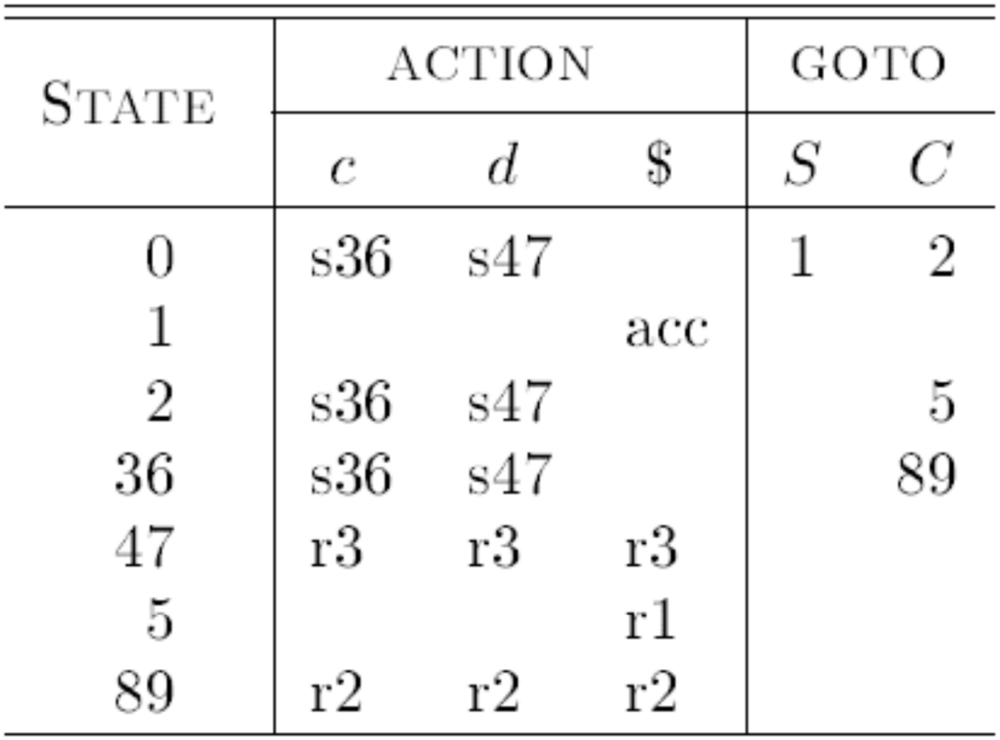
I_7
$C \to d\cdot, \$$

I_3
$C \to c \cdot C, c/d$
$C \to \cdot cC, c/d$
$C \to \cdot d, c/d$

I_8
$C \to cC\cdot, c/d$

I_4
$C \to d\cdot, c/d$

LALR(1) Parse Table/Brute Force



STATE	ACTION			GOTO	
	c	d	$	S	C
0	s36	s47		1	2
1			acc		
2	s36	s47			5
36	s36	s47			89
47	r3	r3	r3		
5			r1		
89	r2	r2	r2		

Figure 4.43: LALR parsing table for the grammar of Example 4.54

LALR Parse…

S' –> S

1. S-> T **else** F**;**
2. T-> E
3. T-> **i;**
4. F->E
5. E-> E**+i**
6. E-> **i**

Configurating set	*Successor*
I0: S' –> •S, $	I1
S –> •T else F;, $	I2
T –> •E, else	I3
T –> •i;, else	I4
E –> •E+i, else/+	I3
E –> •i, else/+	I4
I1: S' –> S•, $	Accept
I2: S –> T• else F;, $	I5
I3: T –> E•, else	Reduce 2
E –> E•+i, else/+	I6
I4: T –> i•;, else	I7
E –> i•, else/+	Reduce 6
I5: S –> T else •F;, $	I8
F –> •E , ;	I9
E–> •E+i , ;/+	I9
E –> •id , ;/+	I10

Configurating set	*Successor*
I6: E –> E+•i, else/+	**I11**
I7: T –> i;•, else	Reduce 3
I8: S –> T else F•;, $	I12
I9: F –> E•, ;	Reduce 4
E–> E•+i , ;/+	I13
I10: E –> i•, ;/+	Reduce 6
I11: E –> E+i•, else/+	**Reduce 5**
I12: S –> T else F;•, $	Reduce 1
I13: E –> E+•i, ;/+	**I14**
I14: E –> E+i•, ;/+	**Reduce 5**

LALR Parse Table…

Stack	i	;	else	+	$	S	T	F	E
0	s4					1	2		3
1					A				
2			s5						
3			r2	s613					
4		s7	r6	r6					
5	s10							8	9
613	s1114								
7			r3						
8		s12							
9		r4		s613					
10		r6		r6					
1114		r5	r5	r5					
12					r1				

When LALR(1) fails…

S' –> S

S –> aBc | bCc | aCd | bBd

B –> e

C –> e

I0: S' –> •S, $
    S –> •aBc, $
    S –> •bCc, $
    S –> •aCd, $
    S –> •bBd, $

I1: S' –> S•, $

I2: S –> a•Bc, $
    S –> a•Cd, $
    B –> •e, c
    C –> •e, d

I3: S –> b•Cc, $
    S –> b•Bd, $
    C –> •e, c
    B –> •e, d

I4: S –> aB•c, $

I5: S –> aC•d, $

I6: B –> e•, c
    C –> e•, d

I7: S –> bC•c, $

I8: S –> bB•d, $

I9: B –> e•, d
    C –> e•, c

I10: S –> aBc•, $

I11: S –> aCd•, $

I12: S –> bCc•, $

I13: S –> bBd•, $

LALR Table Construction

- Merge at the end vs. as you go

S' –> S

S –> V = E

E –> F | E + F

F –> V | int | (E)

V –> id

I0: S' –> •S, $

  S –> •V = E, $

  V –> •id, =

I1: S' –> S•, $

I2: S' –> V• = E, $

I3: V –> id•, =

I4: S –> V =•E, $

  E –> •F, $/+

  E –> •E + F, $/+

  F –>•V, $/+

  F –>•int, $/+

  F –>•(E), $/+

  V –>•id, $/+

I5: S –> V = E•, $

  E –> E• + F, $/+

I6: E –> F•, $/+

I7: F–> V•, $/+

I8: F–> int•, $/+

I9: F–> (•E), $/+

  E –> •F,)/+

  E –> •E + F,)/+

  F –> •V,)/+

  F –> •int,)/+

  F –> •(E),)/+

  V –> •id)/+

I10: F–> (E•), $/+

  E –> E• + F,)/+

???? I11: E –>F•,)/+

???? I12: F –>V•,)/+

Oregon State University