LALR Analysis

Motivation

- As explained before, in LR(1) parsers there are many more states than in the previous procedures, LR(0) and SLR(1).
  - This is because there are states which contain the same configurations, but with different look-ahead symbols.
- A possible simplification of LR(1) parsers are LALR(1) parsers (Look-Ahead Left-to-Right parsers)
  - They have the same number of states as LR(0) and SLR(1) parsers.
  - For illustration, a language like Pascal will have a few hundreds of states if constructed as SLR(1), but it will have thousands if built as LR(1).
- **We are going to build LALR(1) parsers from LR(1) parsers**

LALR Analysis

LALR parsers: introductory example

- We have seen an LR(1) example grammar, which was not SLR(1):
  - **(1) S→A**
  - **(2) S→xb**
  - **(3) A→aAb**
  - **(4) A→B**
  - **(5) B→x**
- this grammar generated the following language:

$$\{xb\} \cup \{a^n x b^n \mid n \geq 0\}$$

LALR Analysis

LALR parsers: introductory example

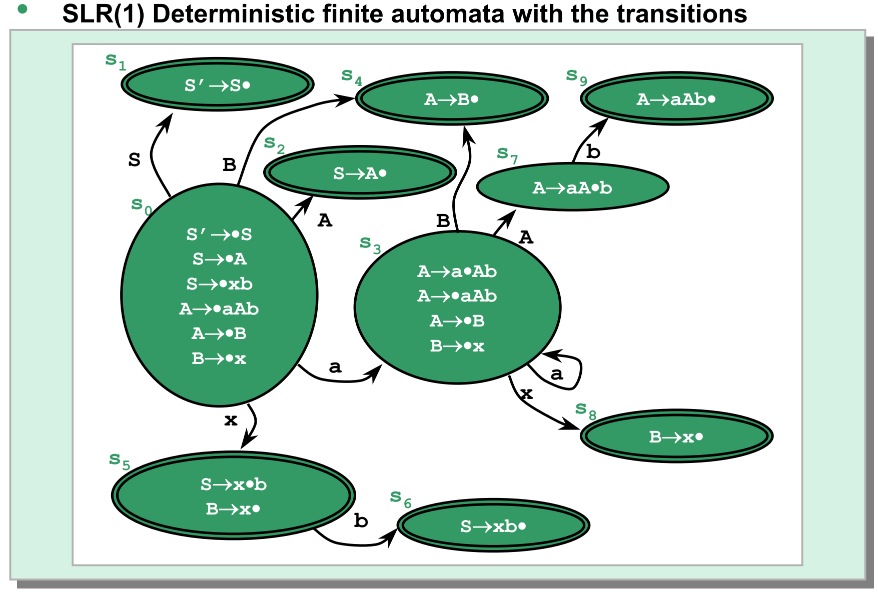
- Remember the augmented grammar:

(0) S'→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

LALR parsers: introductory example

- **SLR(1) Deterministic finite automata with the transitions**



LALR Analysis

LALR parsers: introductory example

- This was the analysis table for the SLR(1) grammar, with the conflict

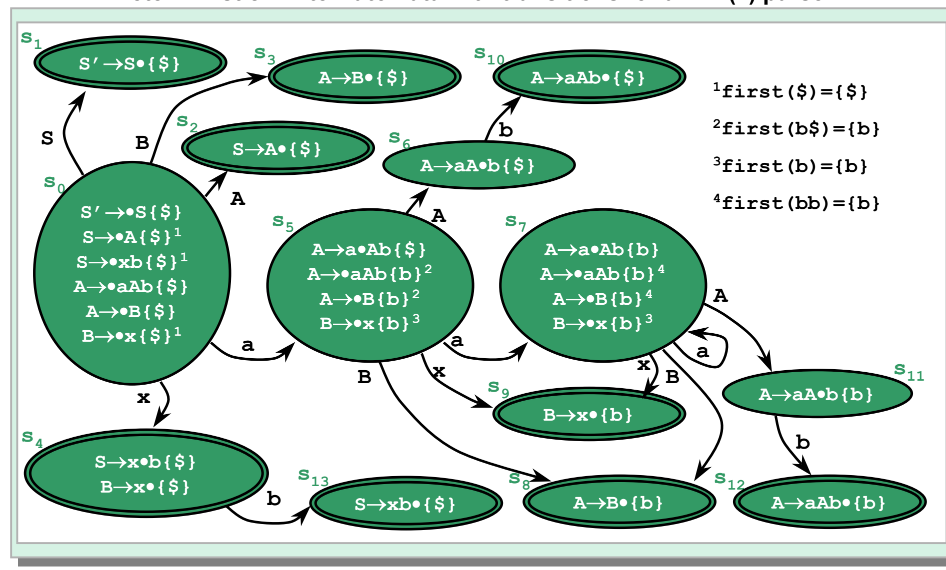
(0) S′→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

	Σ_T				Σ_N		
E	a	b	x	$	S	A	B
0	s3		s5		1	2	4
1				acc			
2				r1			
3	s3		s8			7	4
4		r4		r4			
5		r5/s6		r5			
6				r2			
7							
8		r5		r5			
9		r3		r3			
	Action				Go-to		

LALR Analysis

LALR parsers: introductory example

- **Deterministic Finite Automata with transitions for an LR(1) parser**



LALR Analysis

LALR(1) parsers: introductory example

- As can be seen,
  - The states in an LR(1) parser are the same as in an LR(0) parser, but some of them appear several times with different look-ahead symbols in the configurations.
  - The idea is to simplify the analyser, by merging all the states with the same configurations but different look-ahead symbols.
    - The resulting analyser will contain the same states than the SLR(1) parser.
    - Having less states, the resulting analyser will be less powerful than the original LR(1) parser, and there is a higher probability of having collisions (as happened in SLR(1) parsers)
    - On the other hand, there will be less collisions in LALR(1) parsers than in SLR(1) parsers, so there is a gain in this process.

LALR Analysis

LALR(1)

- In the example,
  - s_5 and s_7 contain the same configurations, but with different look-ahead symbols. We can merge them in the following state:

    s_{57} = {
    A→a•Ab{$,b}
    A→•aAb{b}
    A→•B{b}
    B→•x{b}}
  - s_3 and s_8 can also be merged into the following state:

    s_{38} = {
    A→B• {$,b}}
  - s_6 and s_{11} can also be merged into the following state:

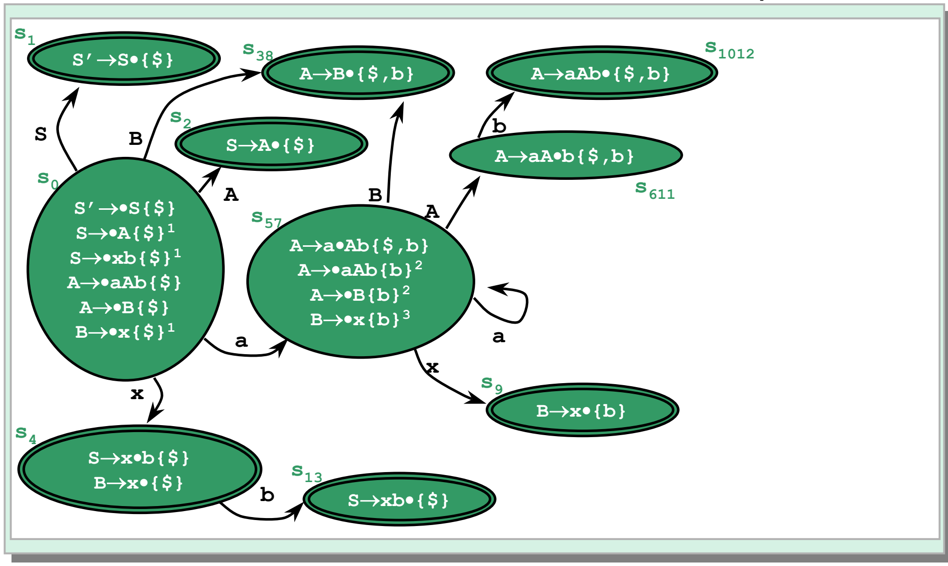
    s_{611} = {
    A→aA•b{$,b}}
  - s_{10} and s_{12} can also be merged into the following state:

    s_{1012} = {
    A→aAb• {$,b}}

LALR Analysis

LALR(1) parsers: introductory example

- **Deterministic Finite Automata with transitions for an LALR parser**



LALR Analysis

LALR(1)

- Shifts in the table:
  - It is the same as in LR(0)
  - They can be obtained by following the transitions in the table.
  - If the automata can go from s_i to s_j by means of symbol X, then we shall add the following action:
    - $$\text{Syntactic\_table[i,X]=}\begin{cases} sj & Si\ X\in\Sigma_T \\ j & Si\ X\in\Sigma_N \end{cases}$$
- Reductions in the table:
  - In the cells for the states which contain reduction configurations, of the form $A\rightarrow\gamma\bullet\{\sigma_1,...,\sigma_n\}$ we have to add the reduction of the rule $A\rightarrow\gamma$ only in the columns for their look-ahead non-terminal symbols, i.e., $\{\sigma_1,...,\sigma_n\}$.
  - Therefore, this step is the same as in LR(1), once the diagram for LALR has been built using the previous procedure.

LALR Analysis

LALR(1)

- **Acceptation:**
  - **It is the same as in LR(0) analysers**
  - If a state s_i has a transition with the terminal symbol **$** to the final state with the configuration `axiom'→axiom $•`, we have to add the accept action to `Syntactic_table[i,$]`.
  - Is it possible to find alternative techniques for acceptation in LR(1) parsers.
- **Error:**
  - **It is the same as in LR(0)**
  - All the empty cells have associated the `error` action.

LALR Analysis

LALR(1): parsing examples

- **Analysis table**

  The following is an example of analysis with two strings:

  aaxbb

  ax

	Σ_T				Σ_N			
E	**a**	**b**	**x**	**$**	**S'**	**S**	**A**	**B**
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

LALR Analysis

{xb, a^nxb^n | n≥0}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	a	x	b	b	$

0

(0) S'→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, a^nxb^n | n≥0}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	a	x	b	b	$

57	a	0

(0) S'→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, a^nxb^n | n≥0}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	a	x	b	b	$

57	a	57	a	0

(0) S'→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, a^nxb^n | n≥0}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	a	x	b	b	$

9	x	57	a	57	a	0

(0) S'→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, $a^n x b^n$ | n≥0}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	a	x	b	b	$

38	B	57	a	57	a	0

(0) S′→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, $a^n x b^n$ | n≥0}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	a	x	b	b	$

611	A	57	a	57	a	0

(0) S′→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, $a^n x b^n$ | n≥0}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	a	x	b	b	$

1012	b	911	A	57	a	57	a	0

(0) S′→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, $a^n x b^n$ | n≥0}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	a	x	b	b	$

611	A	57	a	0

(0) S′→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, $a^n x b^n$ | n≥0}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	a	x	b	b	$

1012	b	611	A	57	a	0

(0) S′→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, $a^n x b^n$ | n≥0}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	a	x	b	b	$

2	A	0

(0) S′→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, $a^n x b^n$ | n≥0}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	a	x	b	b	$

1	S	0

(0) S′→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, $a^n x b^n$ | n≥0}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	a	x	b	b	$

0

(0) S′→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, $a^n x b^n$ | $n \geq 0$}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	x	$									

0											

(0) S'→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, $a^n x b^n$ | $n \geq 0$}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	x	$									

57	a	0									

(0) S'→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, $a^n x b^n$ | $n \geq 0$}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	x	$									

9	x	57	a	0							

(0) S'→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

{xb, $a^n x b^n$ | $n \geq 0$}

	Σ_T				Σ_N			
E	a	b	x	$	S'	S	A	B
0	s57		s4			1	2	38
1				acc				
2				r1				
38		r4		r4				
4		s13		r5				
57	s57		s9				611	38
611		s1012						
9		r5						
1012		r3		r3				
13				r2				
	Action				Go-to			

a	x	$									

0											

(0) S'→S$
(1) S→A
(2) S→xb
(3) A→aAb
(4) A→B
(5) B→x

LALR Analysis

Evaluation

- Power:
  - LALR(1) is less powerful than LR(1), but more so than SLR(1).
  - However, most structures found in programming languages are LALR(1), so they can be parsed with this procedure.

- Efficiency:
  - There are less states in an LALR(1) parser than in an LR(1) parser.