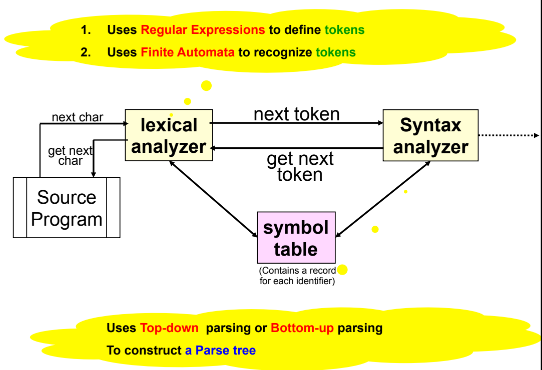
Language Processing Systems

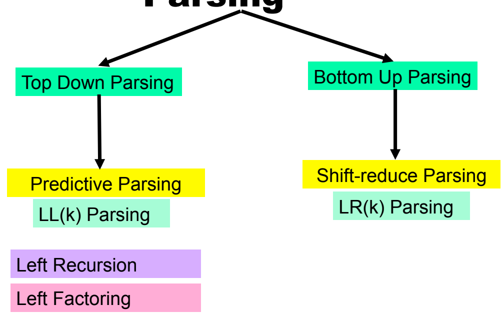
Prof. Mohamed Hamada

Software Engineering Lab.
The University of Aizu
Japan

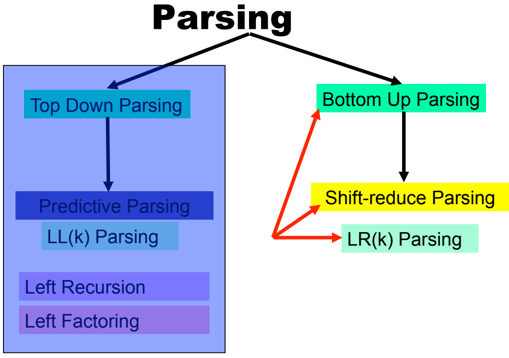
Syntax Analysis (Parsing)



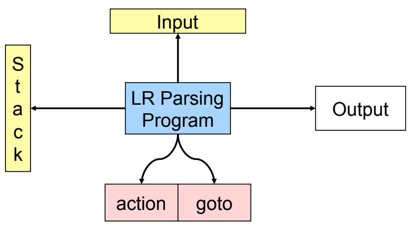
Parsing



Parsing



LR Parser Example



Shift reduce parser

1. Construct the *action-goto table* from the given grammar

2. Apply the shift-reduce parsing algorithm to construct the parse tree

Shift reduce parser

1. Construct the *action-goto table* from the given grammar

This is what make difference between different typs of shift reduce parsing such as SLR, CLR, LALR

In this course due to short of time we will not study how to construct the ***action-goto table***

Shift reduce parser

2. Apply the shift-reduce parsing algorithm to construct the parse tree

The following algorithm shows how we can construct the move parsing table for an input string w$ with respect to a given grammar G.

```
set ip to point to the first symbol of the input string w$
repeat forever
begin
      if action[top(stack), current-input(ip)] = shift(s)  then begin
            push current-input(ip) then s on top of the stack
            advance ip to the next input symbol
      end
      else if action[top(stack), current-input(ip)] = reduce A → β then
      begin
          pop 2*|β| symbols off the stack;
          push A then goto[top(stack), A] on top of the stack;
          output the production A → β
      end
      else if action[top(stack), current-input(ip)] = accept then
              return
      else error()
end
```

LR Parser Example

The following grammar:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Can be parsed with this action and goto table

State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

s represents shift
r represents reduce
acc represents accept
empty represents error

Parser Example

GRAMMAR:
(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

INPUT: | id | + | id | * | id | $ |

STACK: 0

LR Parsing Program

OUTPUT:

State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Parser Example

GRAMMAR:
(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

INPUT: | id | * | id | + | id | $ |

STACK: 5 / id / 0

LR Parsing Program

OUTPUT:

F
|
id

State	action						goto		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Parser Example

GRAMMAR:
(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

INPUT: | id | * | id | + | id | $ |

STACK: 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:

F
|
id

Parser Example

GRAMMAR:
(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

INPUT: | id | * | id | + | id | $ |

STACK:
3
F
0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:

T
|
F
|
id

Parser Example

GRAMMAR:
(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

INPUT: | id | * | id | + | id | $ |

STACK: 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:

T
|
F
|
id

Parser Example

GRAMMAR:
(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

INPUT: | id | * | id | + | id | $ |

STACK:
2
T
0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:

T
|
F
|
id

Parser Example

GRAMMAR:
(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

INPUT: | id | * | id | + | id | $ |

STACK:
7
*
2
T
0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:

T
|
F
|
id

Parser Example

GRAMMAR:
(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

INPUT: | id | * | id | + | id | $ |

STACK:
5
id
7
*
2
T
0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:

T　　F
|　　|
F　　**id**
|
id

GRAMMAR:
(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Parser Example

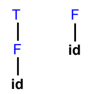
INPUT: **id** | ***** | **id** | **+** | **id** | **$**

STACK: 7 | * | 2 | T | 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:



GRAMMAR:
(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Parser Example

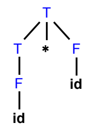
INPUT: **id** | ***** | **id** | **+** | **id** | **$**

STACK: 10 | F | 7 | * | 2 | T | 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:



GRAMMAR:
(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Parser Example

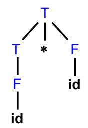
INPUT: **id** | ***** | **id** | **+** | **id** | **$**

STACK: 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:



GRAMMAR:
(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Parser Example

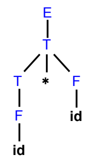
INPUT: **id** | ***** | **id** | **+** | **id** | **$**

STACK: 2 | T | 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:



GRAMMAR:
(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Parser Example

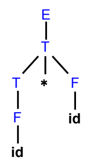
INPUT: **id** | ***** | **id** | **+** | **id** | **$**

STACK: 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:



GRAMMAR:
(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

Parser Example

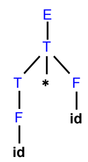
INPUT: **id** | ***** | **id** | **+** | **id** | **$**

STACK: 1 | E | 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:



Parser Example

GRAMMAR:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

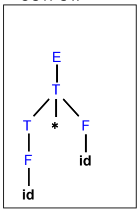
INPUT: | **id** | ***** | **id** | **+** | **id** | **$** |

STACK: 6, +, 1, E, 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:



Parser Example

GRAMMAR:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

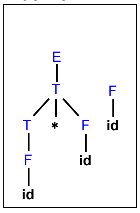
INPUT: | **id** | ***** | **id** | **+** | **id** | **$** |

STACK: 5, id, 6, +, 1, E, 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:



Parser Example

GRAMMAR:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

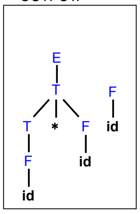
INPUT: | **id** | ***** | **id** | **+** | **id** | **$** |

STACK: 6, +, 1, E, 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:



Parser Example

GRAMMAR:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

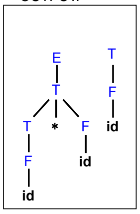
INPUT: | **id** | ***** | **id** | **+** | **id** | **$** |

STACK: 3, F, 6, +, 1, E, 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:



Parser Example

GRAMMAR:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

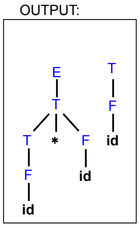
INPUT: | **id** | ***** | **id** | **+** | **id** | **$** |

STACK: 6, +, 1, E, 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:



Parser Example

GRAMMAR:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

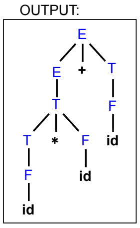
INPUT: | **id** | ***** | **id** | **+** | **id** | **$** |

STACK: 9, T, 6, +, 1, E, 0

LR Parsing Program

State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

OUTPUT:



GRAMMAR:

(1) E → E + T
(2) E → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

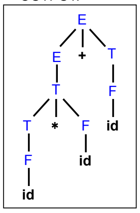
Parser Example

INPUT: | id | * | id | + | id | $ |

STACK: 0

LR Parsing Program

OUTPUT:



State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

GRAMMAR:

(1) E → E + T
(2) E' → T
(3) T → T * F
(4) T → F
(5) F → (E)
(6) F → **id**

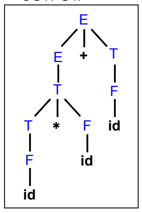
Parser Example

INPUT: | id | * | id | + | id | $ |

STACK: 1 / E / 0

LR Parsing Program

OUTPUT:



State	action						goto		
	id	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Constructing Parsing Tables

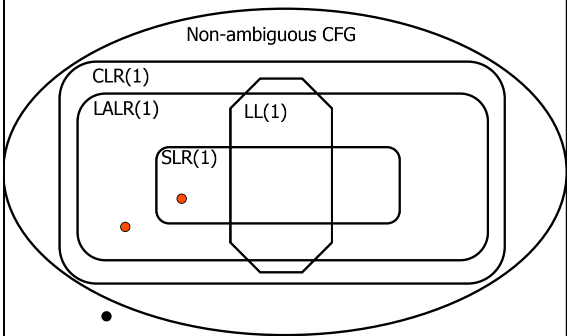
All LR parsers use the same parsing program that we demonstrated in the previous slides.
What differentiates the LR parsers are the action and the goto tables:

Simple LR (SLR): succeeds for the fewest grammars, but is the easiest to implement.

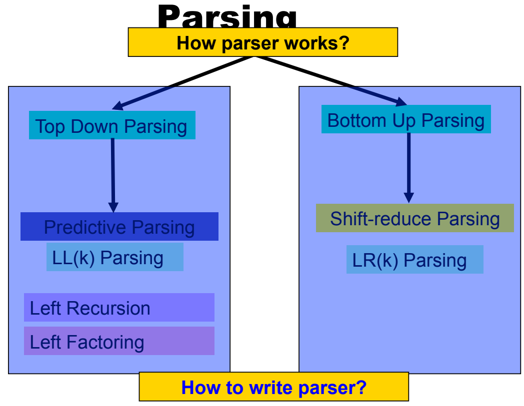
Canonical LR: succeeds for the most grammars, but is the hardest to implement. It splits states when necessary to prevent reductions that would get the parser stuck.

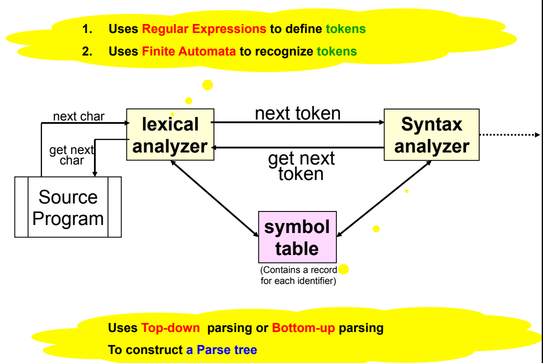
Lookahead LR (LALR): succeeds for most common syntactic constructions used in programming languages, but produces LR tables much smaller than canonical LR.

Grammar Hierarchy



Parsing





How to write a parser?

Yacc