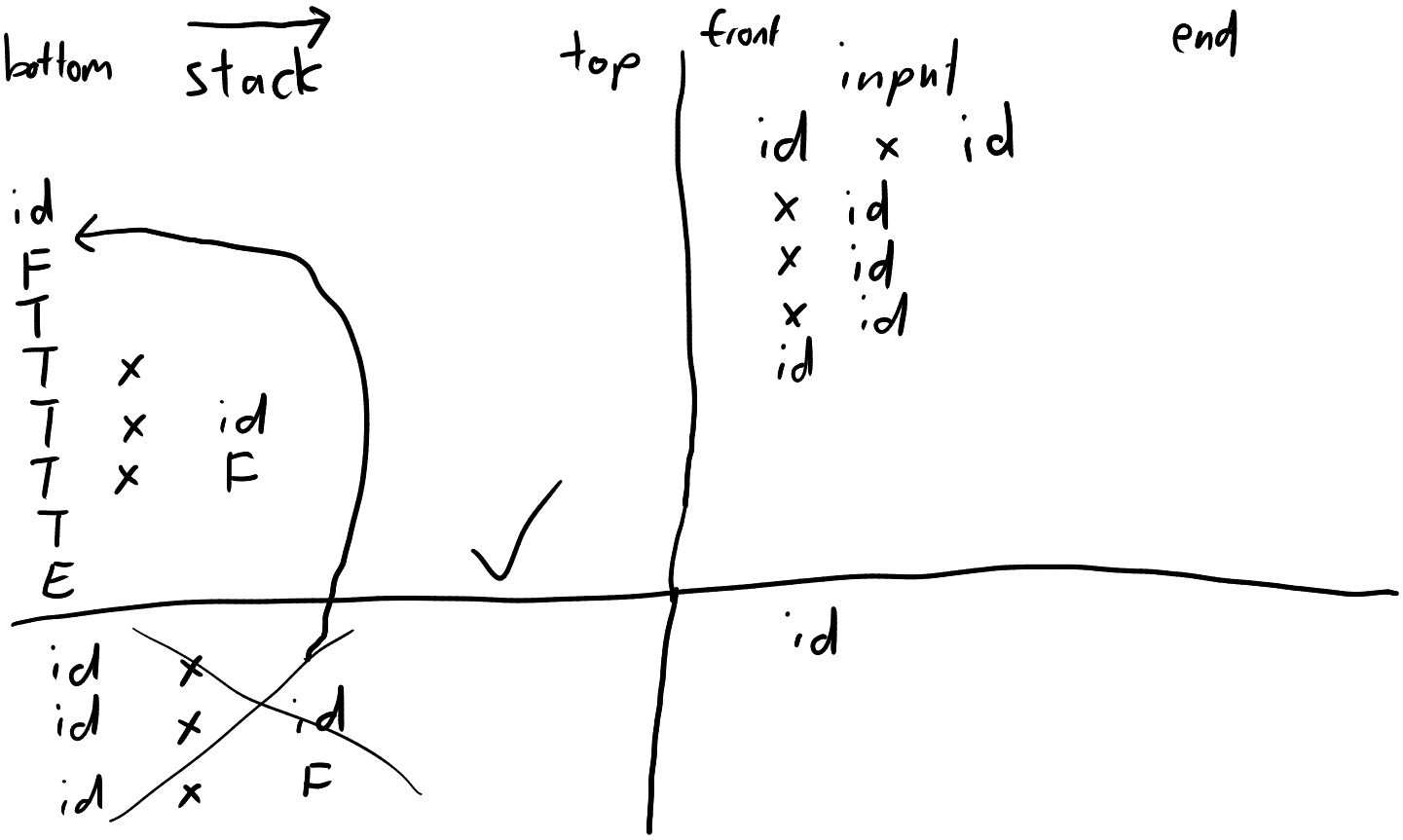
1. shift an input token (as a terminal) on to the stack
2. reduce the top n symbols on the stack by a production
   - replace them with the LHS nonterminal of a production with the same n symbols on its RHS

$E \to E + T \mid T$
$T \to T \times F \mid F$
$F \to (E) \mid id$

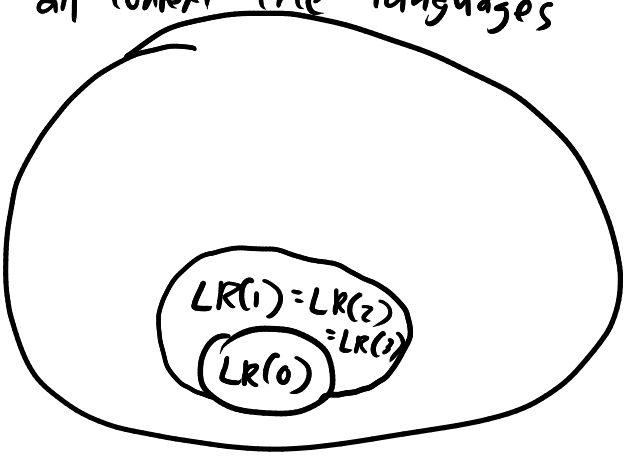
id × id



as languages

$LR(1) = LR(2) = LR(3)$

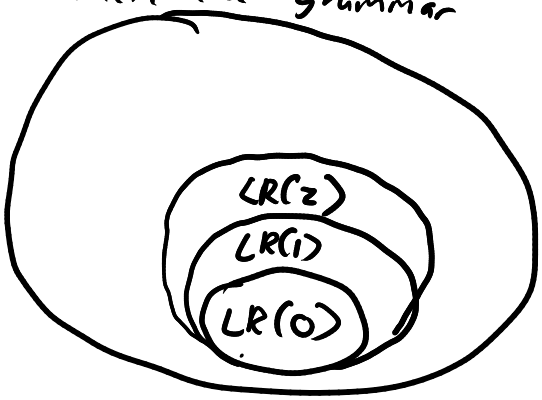
all context free languages



as grammars

$LR(0) \subsetneq LR(1) \subsetneq LR(2) \subsetneq LR(3) \ldots$

context-free grammar



$A \to X\ Y\ Z$ dot is an int $\in [0, n]$ n = #symbols on RHS

$A \to \cdot X\ Y\ Z$

$A \to X \cdot Y\ Z$

$A \to X\ Y \cdot Z$

$A \to X\ Y\ Z \cdot$

(P, d)

$A \to \varepsilon$ $n = 0$ dot $\in [0, 0]$

$A \to \cdot$

```
// I is a set of items
fn closure(I) {
	while changed {
		for (P, d) in I {
			if d == len(P) {
				continue;
			} else if P[d] is a terminal {
				continue;
			} else { // P[d] is a nonterminal
				for each production Q targetting P[d] {
					insert(I, (Q, 0));
				}
			}
		}
	}
}
```

```
// I is a set of items
// X is a symbol
fn goto(I, X) {
	result = {}
	for (P, d) in I {
		if d == len(P) {
			continue;
		} else if P[d] == X {
			insert(result, (P, d + 1));
		}
	}
	return result;
}
```

start = x1y1z1...

```
// G is a grammar
fn compute_states(G) {
	start_state = closure({ (start' ->.start, 0) });
	all_states = { start_state };
	while changed {
		for state in all_states {
			for symbol in G {
				next_state = closure(goto(state, symbol));
				insert(all_states, next_state);
			}
		}
	}
	return all_states;
}
```

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

start state
E' → · E
E → · E + T
E → · T
T → · T × F
T → · F
F → · (E)
F → · id

E → state 1
E' → E ·
E → E · + T
+ → 6

T → state 2
E → T ·
T → T · × F
× → 7

F → state 3
T → F ·

(→ state 4
F → (· E)
E → · E + T
E → · T
T → · T × F
T → · F
F → · (E)
F → · id
id → state 5
F → id ·
(→ 4
F → 3
id → 5
E → state 8
F → (E ·)
E → E · + T
) → state 12
F → (E) ·
+ → 6
T → state 7
T → T × · F
F → · (E)
F → · id
F → 3
T → state 10
E → E + T ·
T → T · × F
× → 7
id → 5
E → state 6
E → E + · T
T → · T × F
T → · F
F → · (E)
F → · id
F → 3
(→ 4
id → 5
T → state 9
E → T ·
T → T · × F
× → 7
F → state 11
T → T × F ·

LR Parsing Table

	+	×	(	)	id	$	E	T	F
0			s4		s5		1	2	3
1	s6	r7	r7	r7	r7				
2		s7							
3									
4			s4		s5		8	2	3
5	r6	r6	r6	r6	r6				
6			s4		s5			10	3
7			s4		s5				11
8	s6			s12					
10		s7							
11									
12									