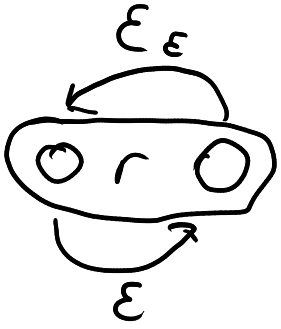
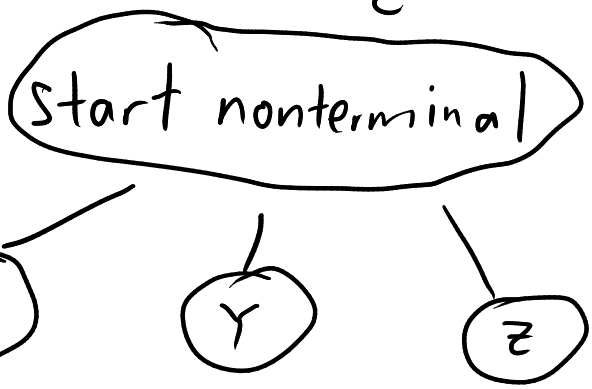
r*





start → X Y Z

X → id

$E \rightarrow E+E \;(1) \mid E \times E \;(2) \mid id \;(3)$

id + id × id

rightmost

E

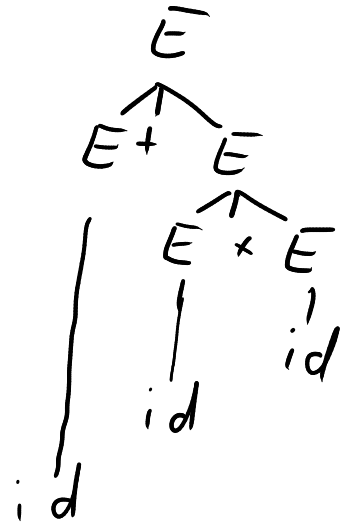
① E + E

② E + E × E

③ E + E × id

③ E + id × id

③ id + id × id



---

leftmost E

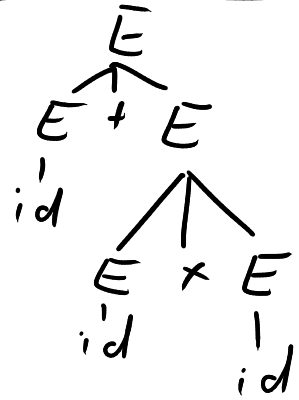
① E + E

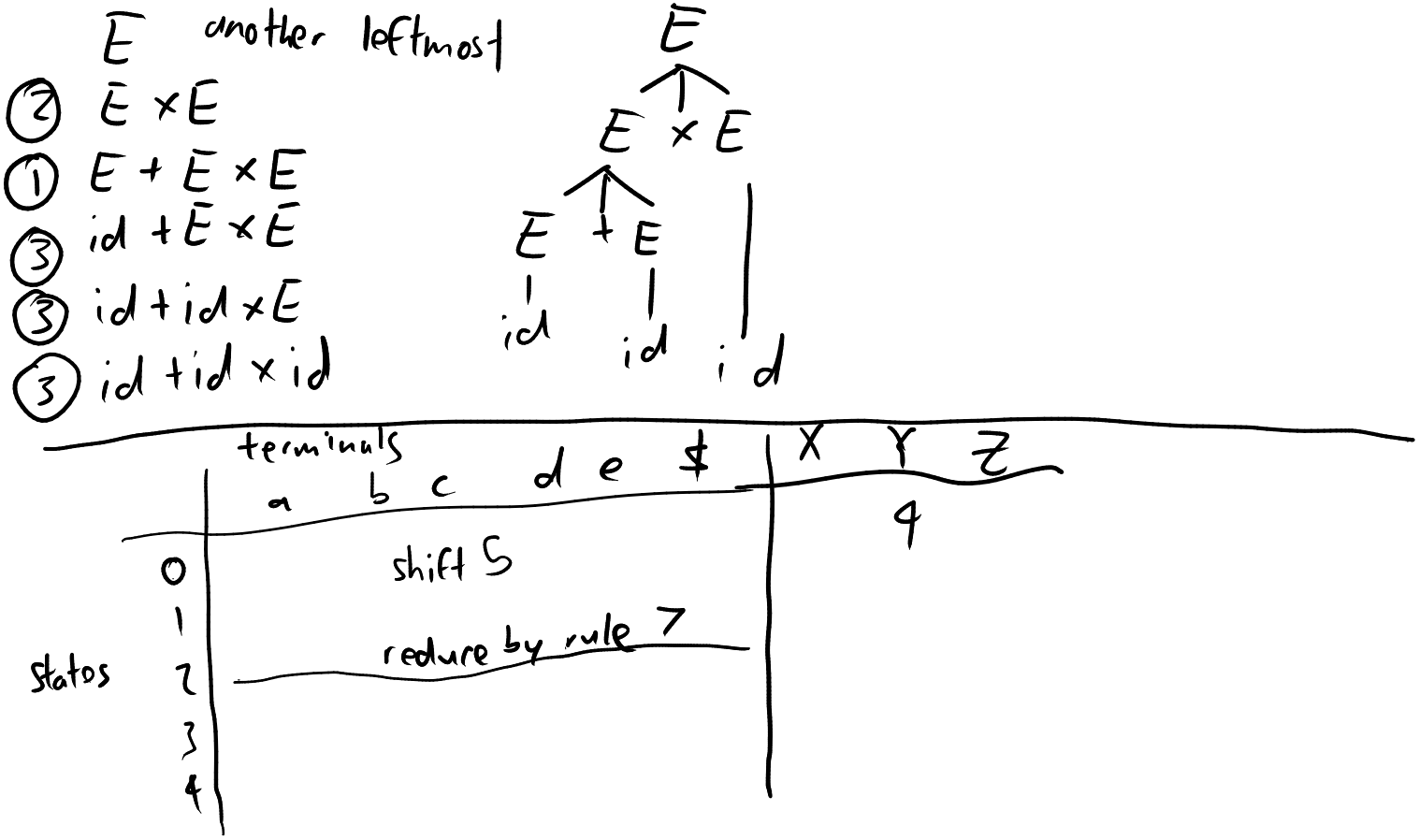
③ id + E

② id + E × E

③ id + id × E

③ id + id × id






```
// LR(1) parsing algorithm
// M is a table like above
// X is the top of the state stack
// a is the next input token
loop {
	if M[X, a] == "shift state j" {
		token_value = advance();
		push_state(j);
		push_value(token_value);
	} else if M[X, a] == "reduce production P" {
		n = new node; // corresponding to the LHS of P
		i = len(rhs(P)) - 1;
		for symbol in reverse(rhs(P)) {
			pop_state();
			n[i] = pop_value();
			i--;
		}
		output(P);
		// X is different from the beginning of the loop; we modified the stack
		push_state(M[X, lhs(P)]);
		push_value(n);
		if P is the augmented start rule {
			break;
		}
	} else {
		// error
	}
}
```

shift

reduce

```
// LR(0) parsing algorithm
loop {
	if X contains a finished item {
		reduce();
	} else {
		a = advance();
		if M[x, a] == "shift state j" {
			shift(j);
		} else {
			// error
		}
	}
}
```