```
EAdd. Exp  ::= Exp  "+" Exp1 ;
EMul. Exp1 ::= Exp1 "*" Integer ;
EInt. Exp1 ::= Integer ;

coercions Exp 1 ;
```

```
Exp -> Exp '+' Exp1          (3)
Exp -> Exp1                  (4)
Exp1 -> Exp1 '*' Integer     (5)
Exp1 -> Integer              (6)
Exp1 -> '(' Exp ')'          (7)
```

the interesting shift-reduce conflict distinguishing arises at the place marked (*)

Stack	Input	CFG Rules	BNFC Rules
empty	1 + 2 * 3		
1	+ 2 * 3		
Integer	+ 2 * 3		
Exp1	+ 2 * 3	6	EInt
Exp	+ 2 * 3	4	
Exp +	2 * 3		
Exp + 2	* 3		
Exp + Integer	* 3		
Exp + Exp1	* 3	6	EInt (*)
Exp + Exp1 *	3		
Exp + Exp1 * 3			
Exp + Exp1 * Integer	5		EInt
Exp + Exp1	3		EMul
Exp			EAdd

read input shift

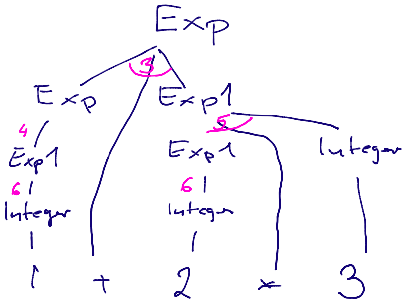
apply a rule reduce

	1	+ 2 * 3
shift	1 + .	2 * 3
reduce	Integer	+ 2 * 3

reductions are only allowed on the top of the stack

From this table extract the

concrete syntax tree



abstract syntax tree