Attribute grammar and dependency graph

Attributes :- {num, id}

AS(exp) = AS(exp2) = AS(exp3) = AS(term) = {num↑ : double}

AS(assignment) = {id↑ : char}

AS(line) = {num↑ : double , id↑ : char}

Attribute Grammar :-

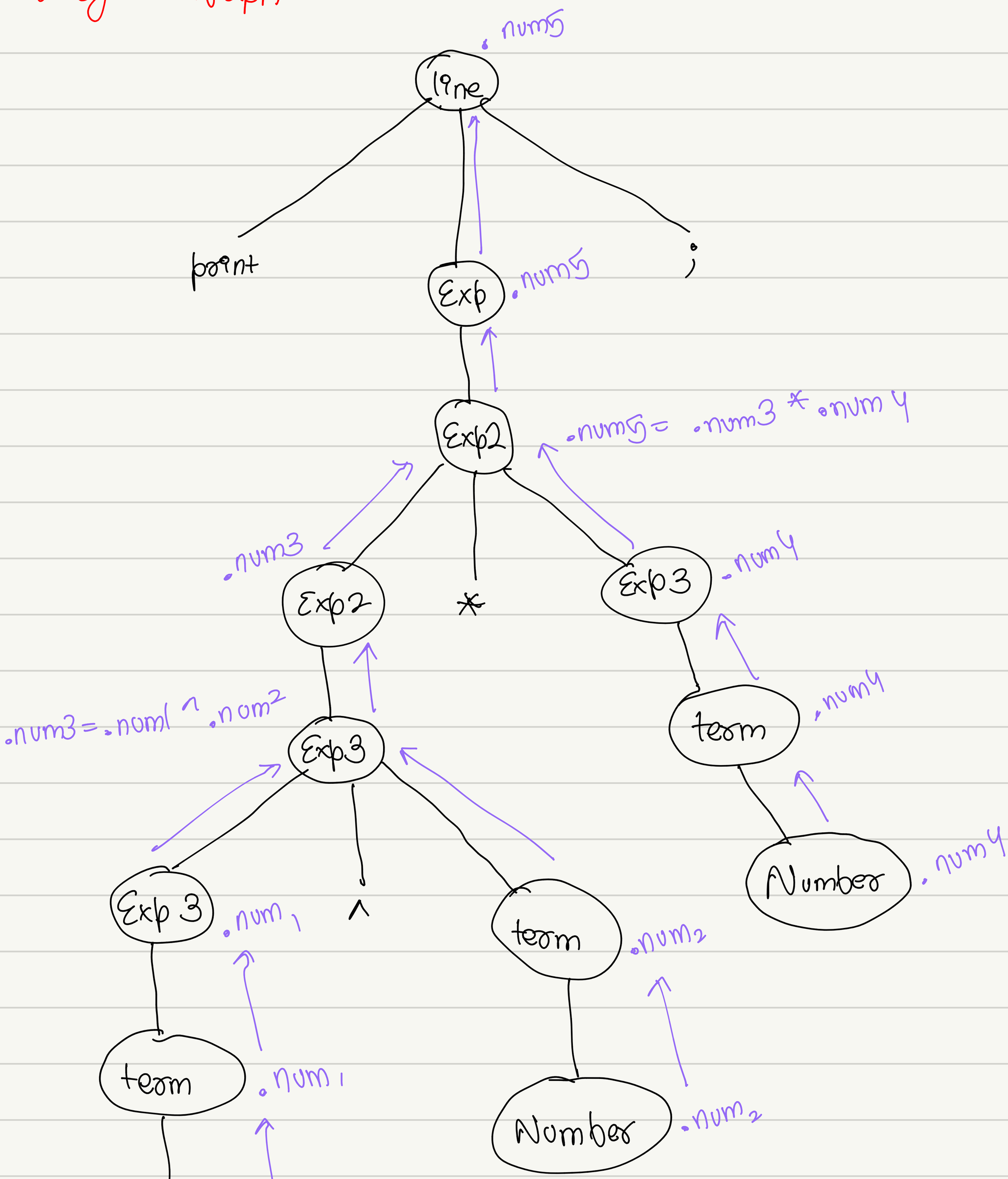
1. line -> assignment ;	{ line.id↑ := assignment.id↑ }
2. line -> exit;	{ }
3. line -> print exp ;	{ line.num↑ := exp.num↑ }
4. line -> line assignment ;	{ line.num↑ := line.num↑ , line.id↑ = assignment.id↑}
5. line -> line print exp;	{ line.num↑ := line.num↑ , line.num↑ = exp.num↑}
6. line -> line exit ;	{ line.num↑ := line.num↑}
7. assignment -> identifier = exp	{ assignment.id↑ := identifier.id↑ }
8. exp -> exp2	{ exp.num↑ := exp2.num↑ }
9. exp -> exp + exp2	{ exp.num↑ := exp.num↑ + exp2.num↑ }
10. exp -> exp - exp2	{ exp.num↑ := exp.num↑ - exp2.num↑ }
11. exp2 -> exp3	{ exp2.num↑ := exp3.num↑ }
12. exp2 -> exp2 * exp3	{ exp2.num↑ := exp2.num↑ * exp3.num↑ }
13. exp2 -> exp2 % exp3	{ exp2.num↑ := fmod(exp2.num↑, exp3.num)↑ }
14. exp2 -> exp2 / exp3	{ exp2.num↑ := exp2.num↑ / exp3.num↑ }
15. exp3 -> term	{ exp3.num↑ := term.num↑ }
16. exp3 -> exp3 ^ term	{ exp3.num↑ := pow(exp3.num↑, term.num↑) }
17. term -> number	{ term.num↑ := number }
18. term -> (exp)	{ term.num↑ := exp.num↑ }
19. term -> identifier	{ term.num↑ := symbol(identifier.id↑); }

Examples →

print 5^3*4 ; ⟶ print Number ^ Number * Number ;

Parse Tree

Dependency Graph



- num1 = 5
- num2 = 3
- num4 = 4

- num3 = 125
- num5 = 500