Syntax Analysis

★ Grammar (EBNF)

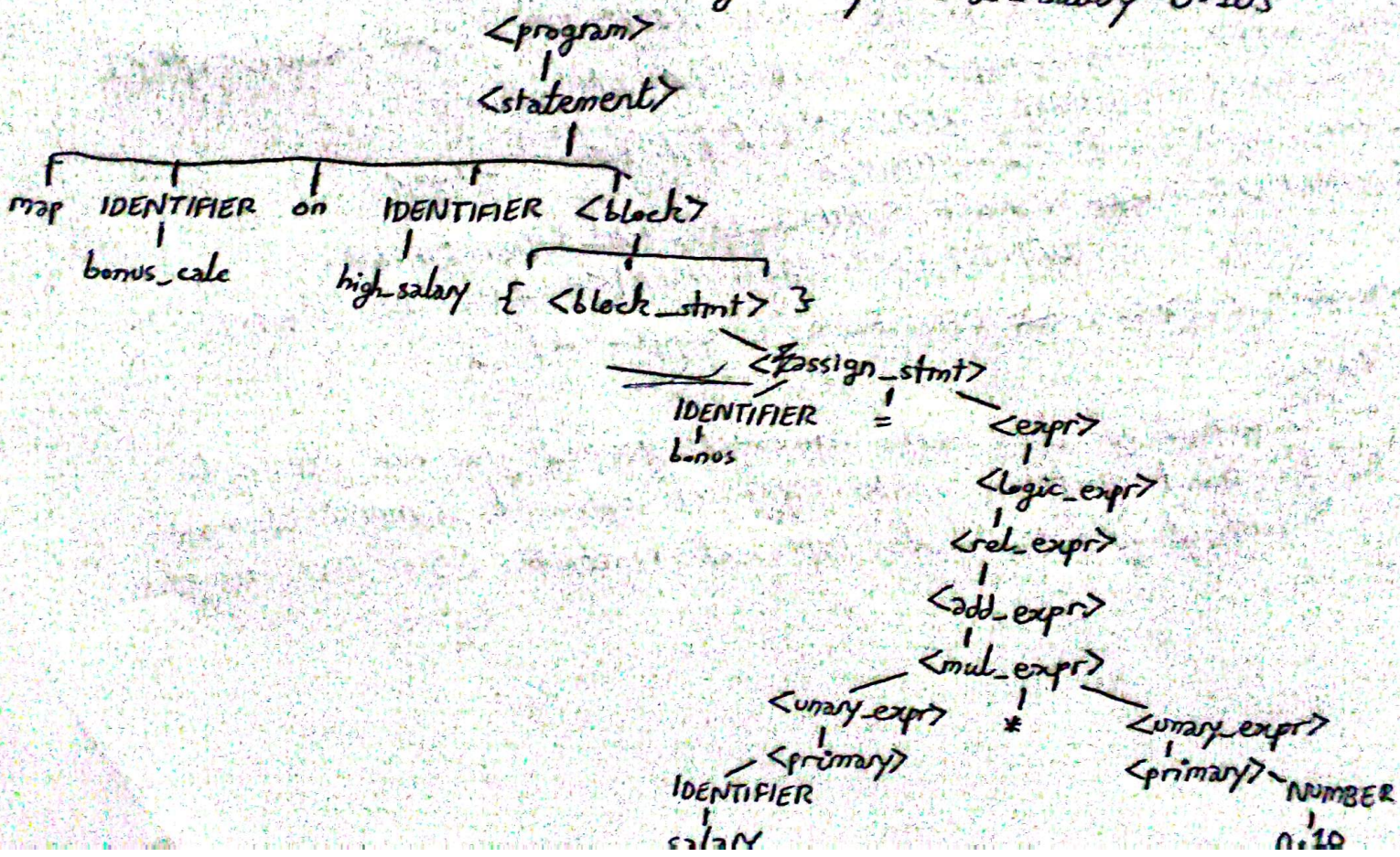
<program> ::= { <statement> }
<statement> ::= <load_stmt> | <filter_stmt> | <map_stmt> |
<aggregate_stmt> | <for_stmt> | <print_stmt> |
<assign_stmt> | ~~<expr_stmt>~~
<load_stmt> ::= "load" ~~id~~ IDENTIFIER "from" STRING_LITERAL
<filter_stmt> ::= "filter" IDENTIFIER <block>
<map_stmt> ::= "map" IDENTIFIER "on" IDENTIFIER <block>
<aggregate_stmt> ::= "aggregate" IDENTIFIER "on" IDENTIFIER <block>
<for_stmt> ::= "for" IDENTIFIER "in" IDENTIFIER <block>
<print_stmt> ::= "print" ~~<expr>~~ <expr_list>
<assign_stmt> ::= ~~<id>~~ IDENTIFIER "=" <expr>
<block> ::= "{" { <block_stmt> } "}"
<block_stmt> ::= <where_stmt> | <assign_stmt> | <print_stmt>
<where_stmt> ::= "where" <expr>
<expr_list> ::= <expr> {"," <expr>}
<expr> ::= <logic_expr>
<logic_expr> ::= <rel_expr> { ("and" | "or") <rel_expr> }
<rel_expr> ::= <add_expr> [("==" | "!=" | "<" | ">" | "<=" | ">=") <add_expr>]
<add_expr> ::= <mul_expr> { ("+" | "-") <mul_expr> }
<mul_expr> ::= <unary_expr> { ("*" | "/" | "%") <unary_expr> }
<unary_expr> ::= ["-"] <primary>
<primary> ::= ~~<id>~~ NUMBER | STRING_LITERAL | IDENTIFIER |
IDENTIFIER "." IDENTIFIER | <function_call> | "(" <expr> ")"
<function_call> ::= ~~<id>~~ IDENTIFIER "(" [expr_list] ")"

*Notes:
- ( ) without quotation marks represents grouped grammar elements
- [ ] without quotation marks represents optional grammar elements
- { } without quotation marks represents repetition (0 or more times)

* Example Program (For parse tree and derivation)

```
load employees from "employees.csv"
filter high_salary {
    where salary > 50000
}

map bonus_calc on high_salary {
    bonus = salary * 0.10
}

aggregate stats on bonus_calc {
    avg_salary = avg(salary)
    avg_bonus = avg(bonus)
}

print stats

for row in bonus_calc {
    print row.name, row.salary, row.bonus
}
```

* Parse Tree for "map bonus_calc on high_salary { bonus = salary * 0.10 }"



* Parse Tree for "for row in bonus_calc {print row.name, row.salary, row.bonus}