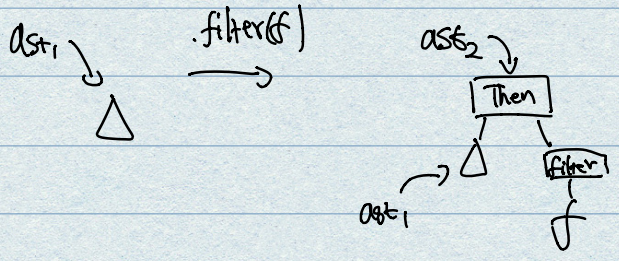
Lecture #5.

Implementation of fluent AST construction.

Q.apply(Math.sqrt)..... filter(f)

ast1



Adjust for call chaining

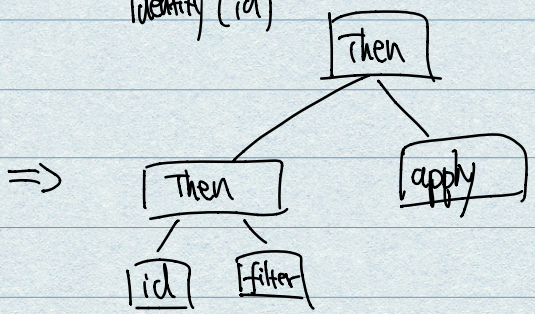
Q.filter(f).apply(g)

↑

query

↓

Identity (id)



Q has a type of ASTNode (subtype IdNode)

Optimization.

inefficiency of 'q.filter(f).count()'

① stores the data ② two iterations.

ThenNode (ThenNode (x , FilterNode(f)) , count)

→ ThenNode (x , countIf(f))

"tree rewrite rule".

Optimization traversal order.

- Bottom up / Top down ?
- post-order optimization
  - 1 → left child
  - 2 → right child
  - 3 → this node

(Lecture #5)

Global Optimization.

- control flow.
- multiple execution paths
- Control Flow Graph.

=> · Global Flow Analysis
· Constant propagation
· Liveness analysis (eleminate redundant stmts)

Local Optimization.

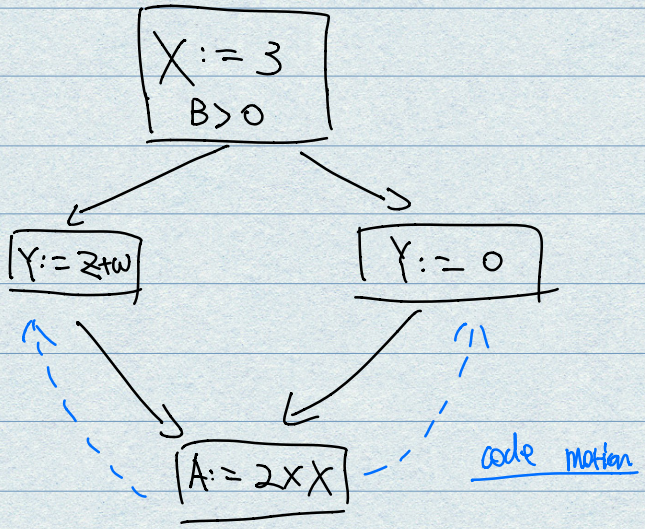
· Basic block: code sequence with no jumps
· Basic block optimization:
  - Constant propagation.
  - Dead code elimination.

- etc

X := 3

Y := Z × W

Q := X + Y



To Replace a use of x by a constant k.

→ every path to the use of x

the last assignment to x is x := k.

Global Analysis

- The optimization depends on the property Prop at a particular point in program execution.

  { Prop is definitely true
  { don't know whether Prop is true.