Lecture #4. Query Language.

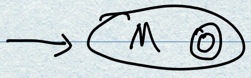
Call Chaining for AST Construction

Program Optimization via AST Rewrite.

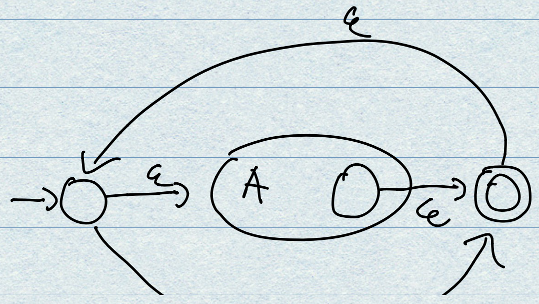
$$\begin{tikzcd}
\text{AST} \arrow[d] \\
\boxed{\text{Syntax-driven code generation.}} \arrow[d] \\
\text{NFA} \arrow[d] \\
\boxed{\text{Subset Construction}} \arrow[d] \\
\text{DFA}
\end{tikzcd}$$

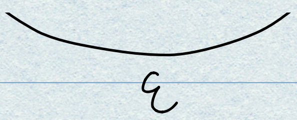
=> Regular Expressions to NFA

- Construct NFA for its constituents
- Notation: NFA for rexp M.

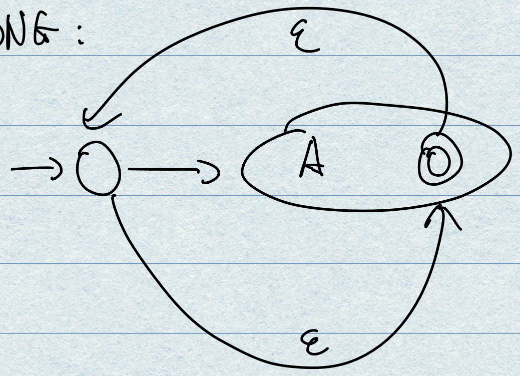


A^* :

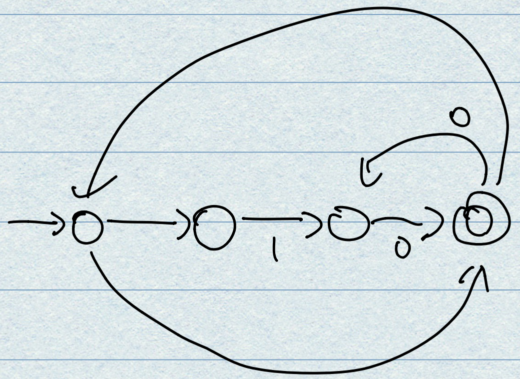




WRONG :



why : $\left(10(00)^*\right)^*$



$\boxed{\text{00 is accepted}}$ X .

Query Language.

```
var data = [ . . . ];
var query = Q.filter( x => . . . . );
var out = query.run(data);
```

=> Deep embedding -> optimization.

↳ program is represented as data (AST)

⇒ Shallow embedding

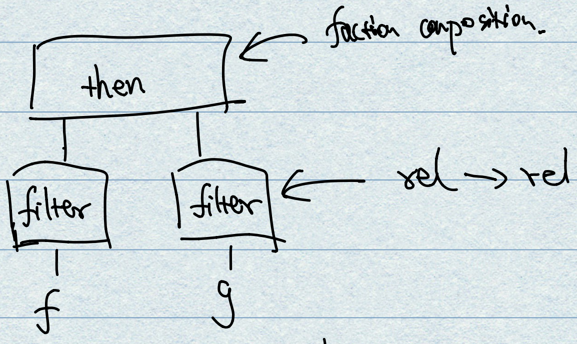
data.filter (λ);

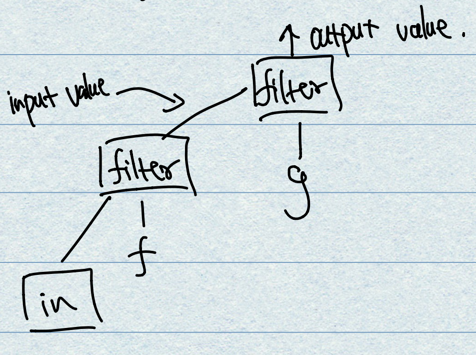
Differences :

1. we can optimize AST in deep version.
2. deferred execution. (run later)

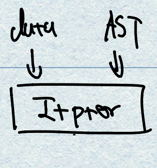
```
Q.filter ( x => x. stars >= 2)
 .filer( x => x.stars <= 8);
```

AST:



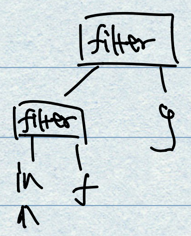


The Interpreter.

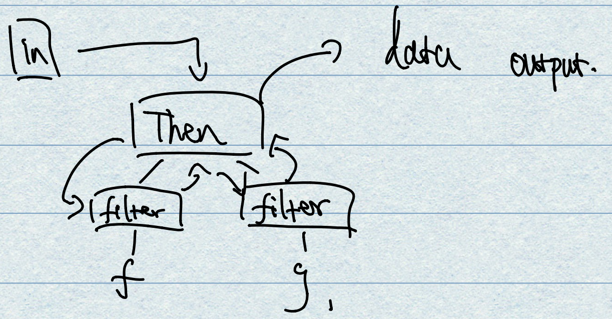


↓
output.

bottom up or topdown?



starts here, → bottom up.


in
data
output.
Then
filter
filter
f
g,