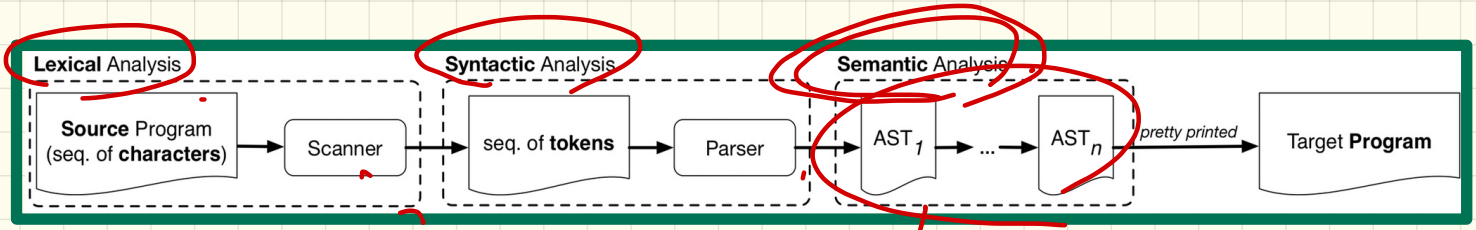
Lecture 2

Wednesday January 8

Example Compiler One: Scanner, Parser, Optimizer

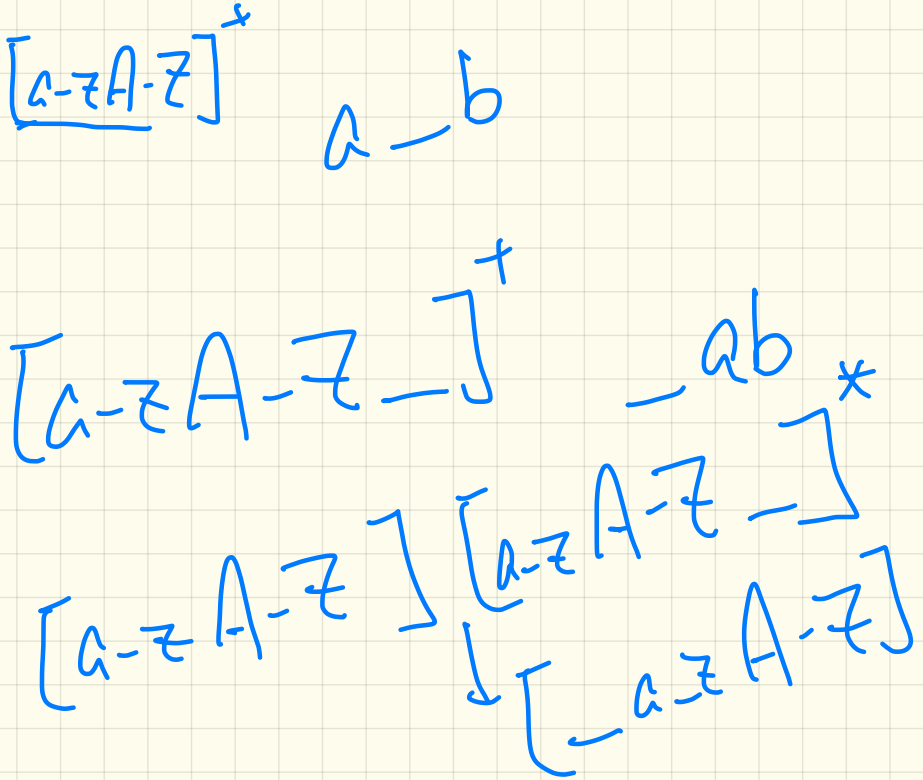


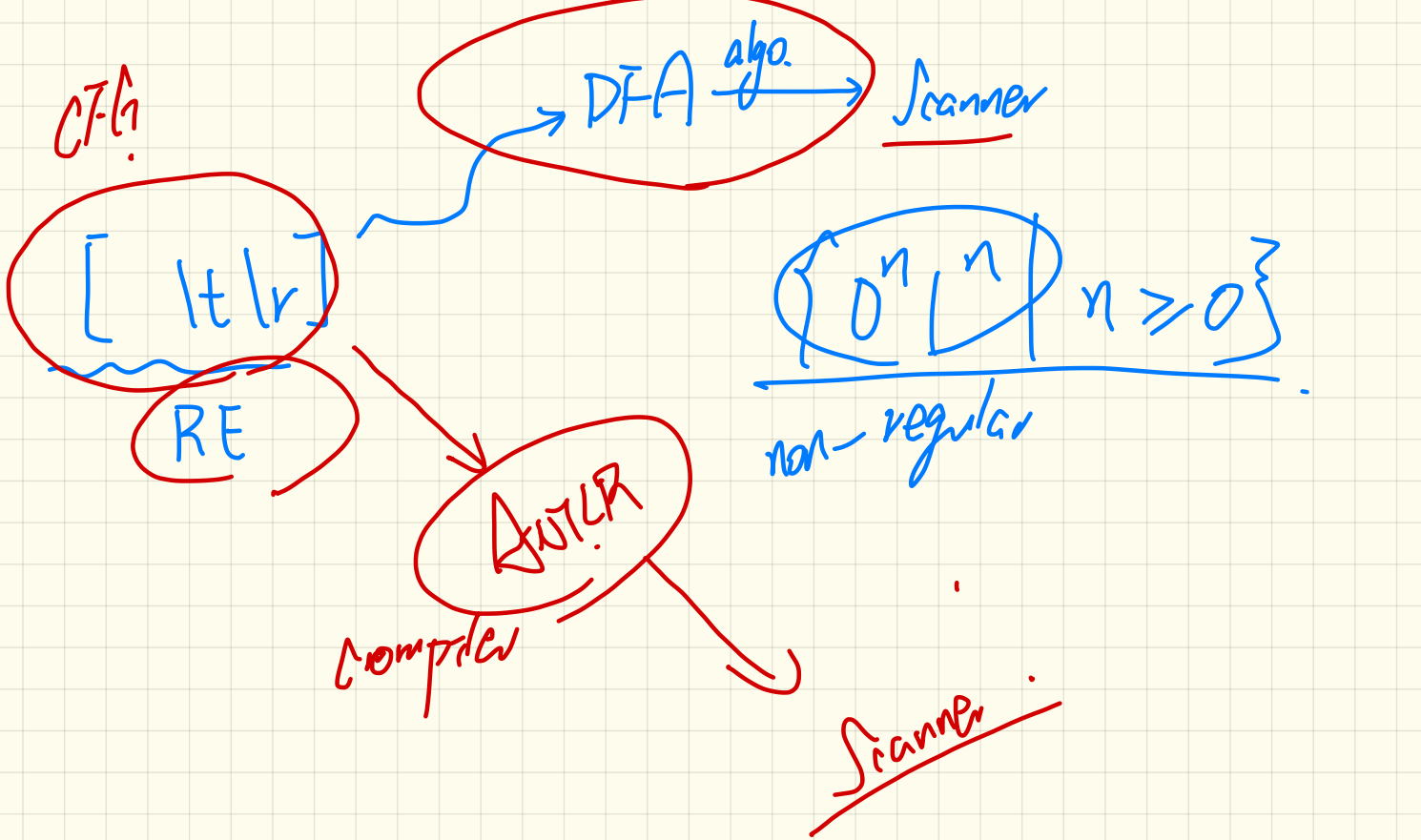
class A { B b; C c; }

Python
Haskell

IR → IR
↳ semantics-preserving

class A { B b; C c; } → class A { B b; C c; }




CFG
DFA
algo.
Scanner
[|t|r]
RE
{ 0^n 1^n | n ≥ 0 }
non-regular
LEX
compiler
Scanner

CFG for while-loop.

::=

WhileLoop T_WHILE T_LPAREN BoolExp T_RPAREN T_LCB

"while" "(" ")" "{"

Non-Terminals (recursive case)

Instruction

T_RCB

"}"

BoolExp ::= . . -

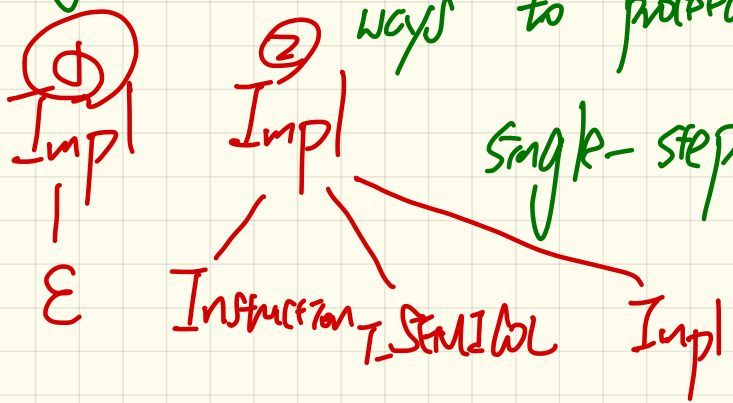
Instruction ::= . . -

while-Loop: Context-Free Grammar (CFG)

WhileLoop	::=	WHILE LPAREN *BoolExpr* RPAREN LCBRAC *Impl* RCBRAC
Impl	::=	
	|	*Instruction* SEMICOL *Impl*

alternative (or)

starting from Impl; how many possible ways to proceed with a single-step of derivation?



① while (...) { x = y; }

② while (...) { x = y; y = z; }

a: ARRAY[INT] a[0].

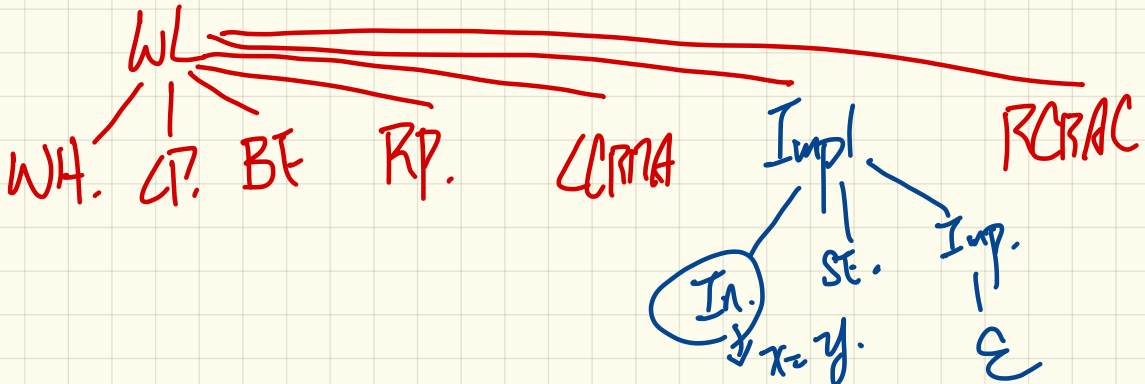
WhileLoop ::= WHILE LPAREN *BoolExpr* RPAREN LCBRAC *Impl* RCBRAC

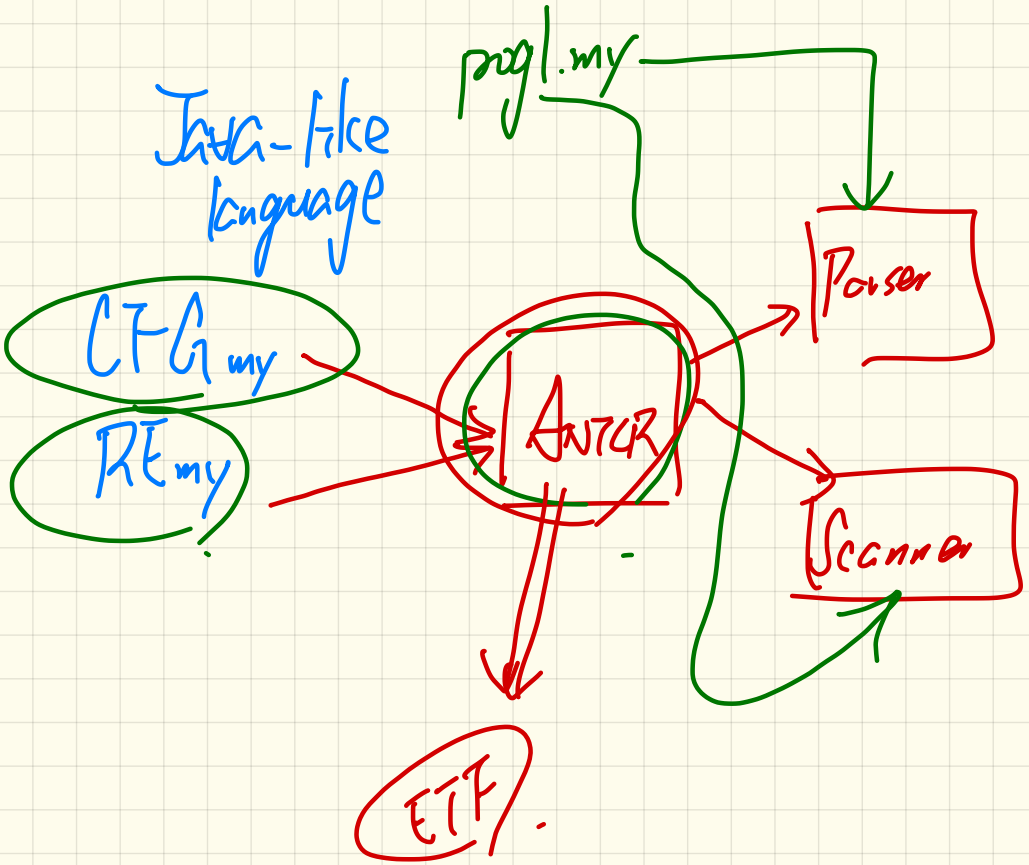
Impl ::= ① ε

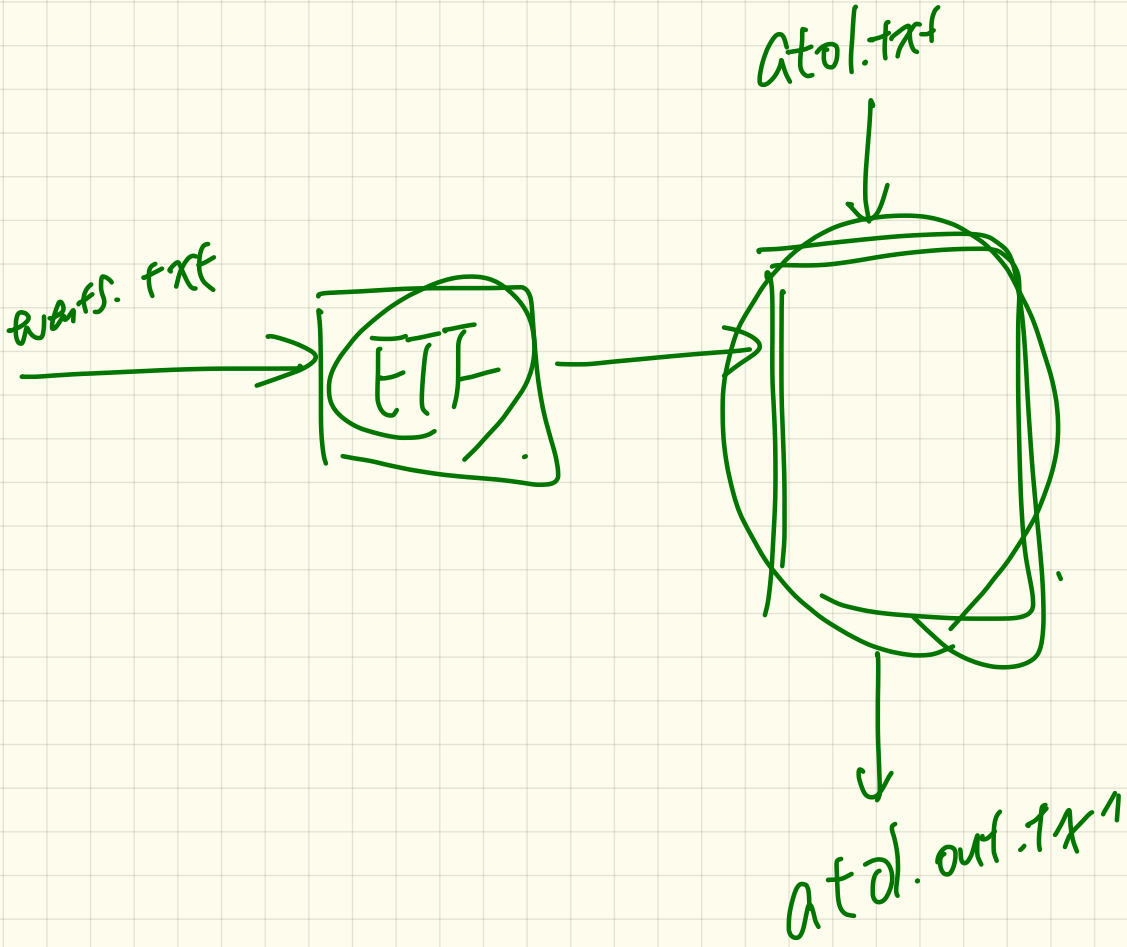
| ② *Instruction* SEMICOL *Impl*

① while (...) {
x = y;
}

② while (...) { x = y;
y = z; }



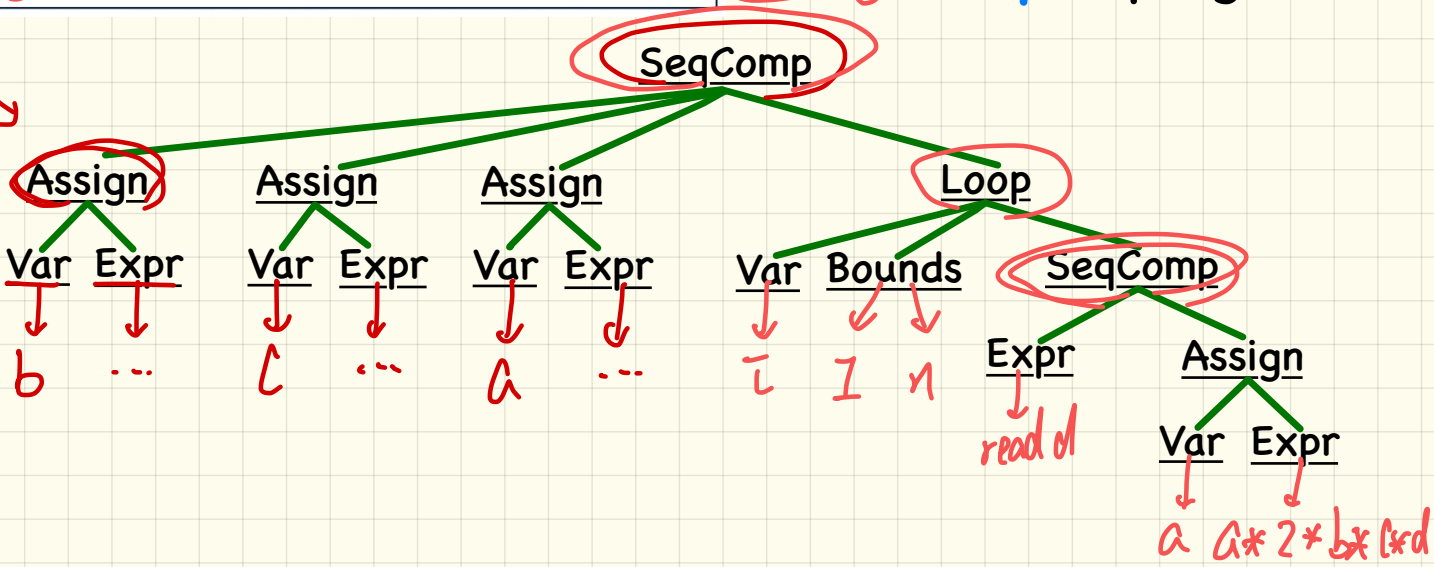

prog1.my
Java-like language
CFG my
RE my
ANTLR
Parser
Scanner
ETF


atol.txt
words.txt
ETL
atol.out.txt

Example Compiler One: AST-to-AST Optimizer (1)

```
b := ... ; c := ... ; a := ... ;
across 1 |..| n is i
  loop
    read d
    a := a * 2 * b * c * d
  end
```

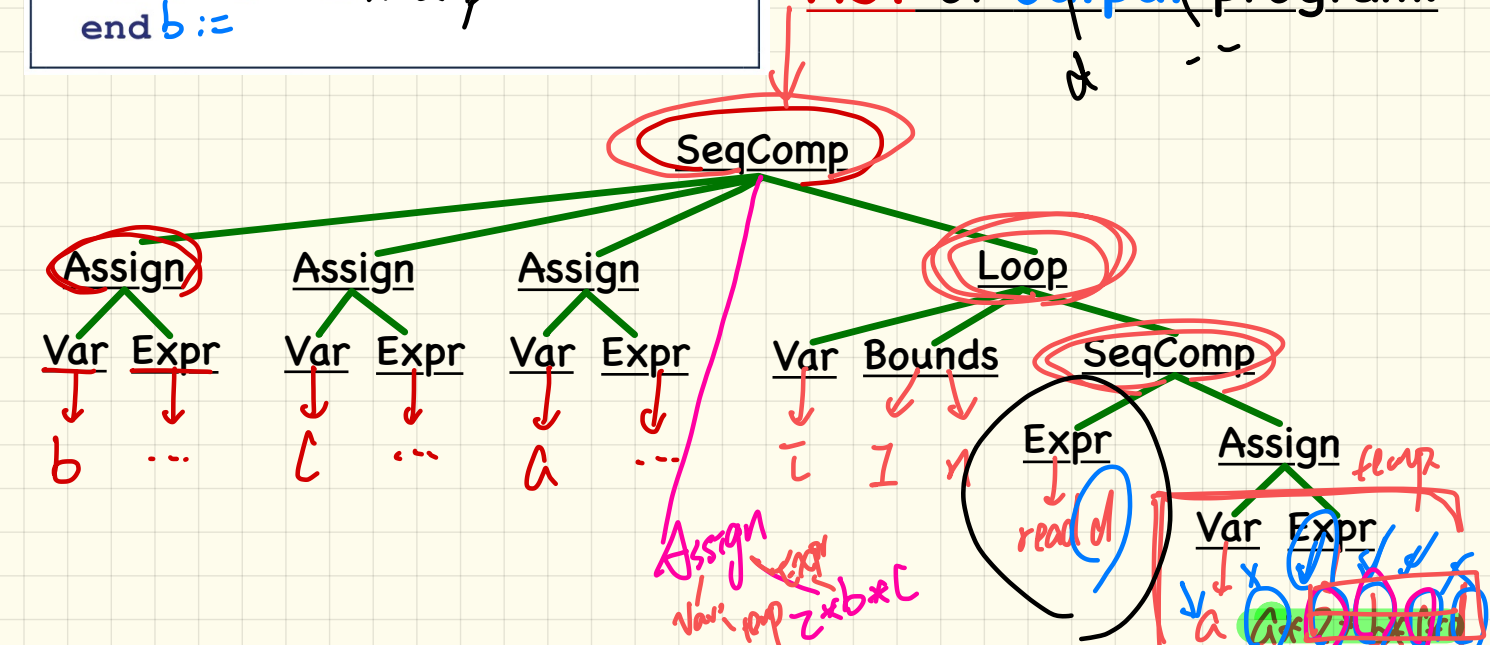
AST of input program:



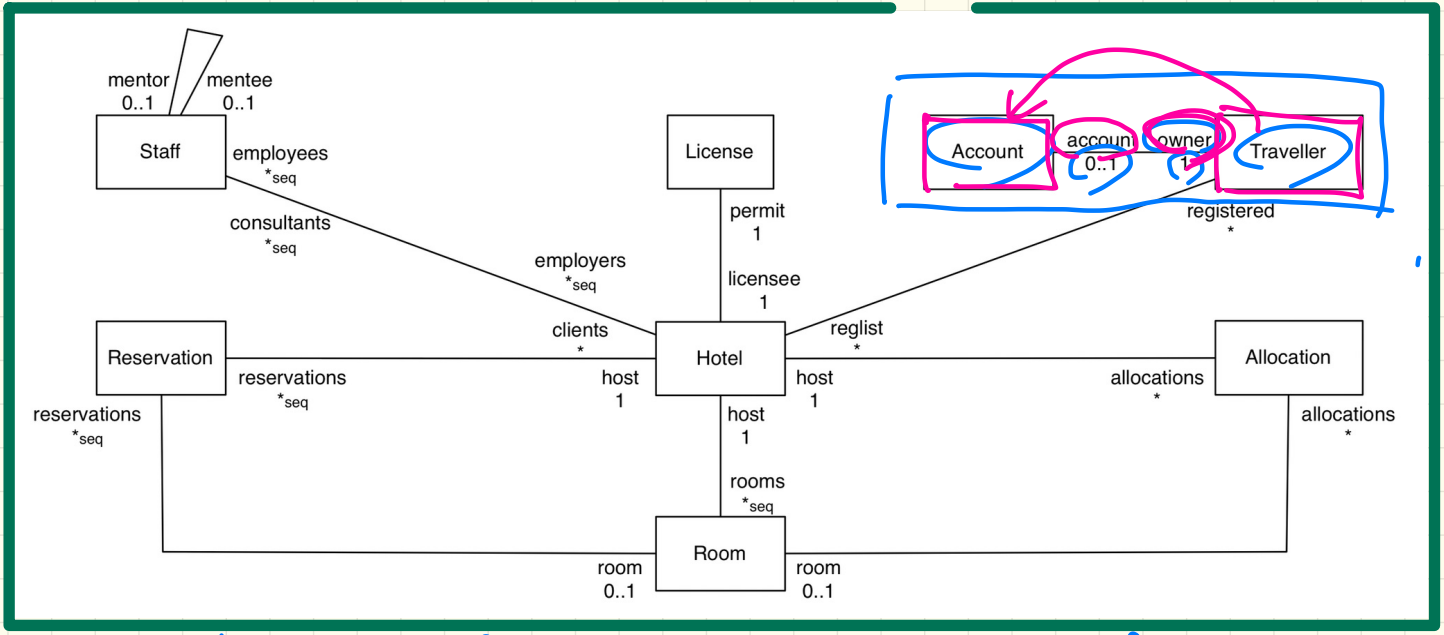
Example Compiler One: AST-to-AST Optimizer (2)

```
b := ... ; c := ... ; a := ...
temp := 2 * b * c
across 1 |..| n is i
  loop
    read d
    a := a * d * temp
  end
```

AST of output program:



Example Compiler Two: Data Model



class Account{

owner : Traveller.account [1]

}

class Traveller{

account : Account.owner [0..1]

}

Example Compiler Two: Mapping Data

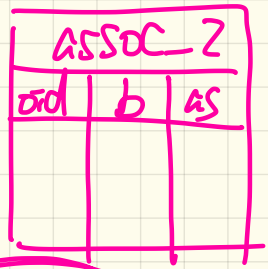
Attribute-to-Table Mapping

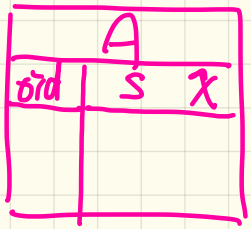
	SINGLE-VALUED	MULTI-VALUED
PRIMITIVE-TYPED	column in *class table*	*collection table*
REFERENCE-TYPED	*association table*	

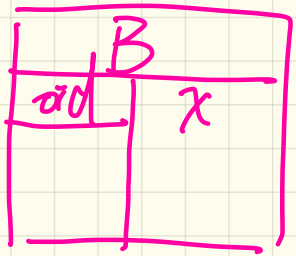
Example Transformation

```
class A {
  attributes
    s: string
    as: set(A . b) [*] }
```

```
class B {
  attributes
    is: set (int)
    b: B . as }
```







Example Compiler Two: Source Program


```
class Account {
  attributes
    owner: Traveller . account
    balance: int
}
```

```
class Traveller {
  attributes
    name: string
    reglist: set(Hotel . registered)[*]
}
```

```
class Hotel {
  attributes
    name: string
    registered: set(Traveller . reglist)[*]
  methods
    register {
        t? : extent(Traveller)
      & t? /: registered
      ==>
          registered := registered \/ {t?}
      || t?.reglist := t?.reglist \/ {this}
    }
}
```

Example Compiler Two: Target Program

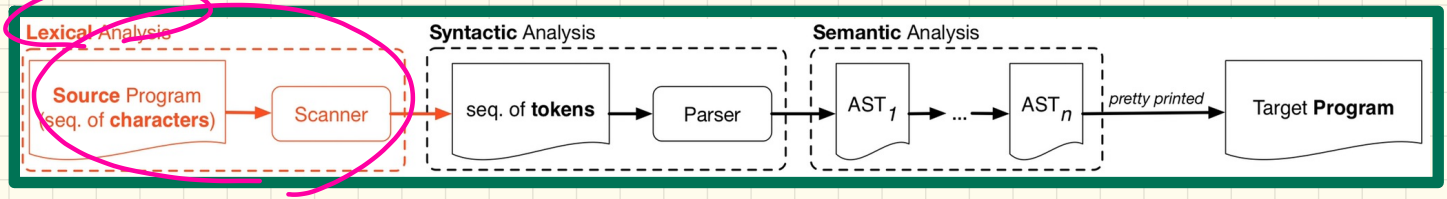
Table Schemas

```
CREATE TABLE `Account`(
  `oid` INTEGER AUTO_INCREMENT, `balance` INTEGER,
  PRIMARY KEY (`oid`));
CREATE TABLE `Traveller`(
  `oid` INTEGER AUTO_INCREMENT, `name` CHAR(30),
  PRIMARY KEY (`oid`));
CREATE TABLE `Hotel`(
  `oid` INTEGER AUTO_INCREMENT, `name` CHAR(30),
  PRIMARY KEY (`oid`));
CREATE TABLE `Account_owner_Traveller_account`(
  `oid` INTEGER AUTO_INCREMENT, `owner` INTEGER, `account` INTEGER,
  PRIMARY KEY (`oid`));
CREATE TABLE `Traveller_reglist_Hotel_registered`(
  `oid` INTEGER AUTO_INCREMENT, `reglist` INTEGER, `registered` INTEGER,
  PRIMARY KEY (`oid`));
```

Stored Procedures

```
CREATE PROCEDURE `Hotel_register`(IN `this?` INTEGER, IN `t?` INTEGER)
  BEGIN
    ...
  END
```

Scanner in Context



Scanner: Formulation & Implementation