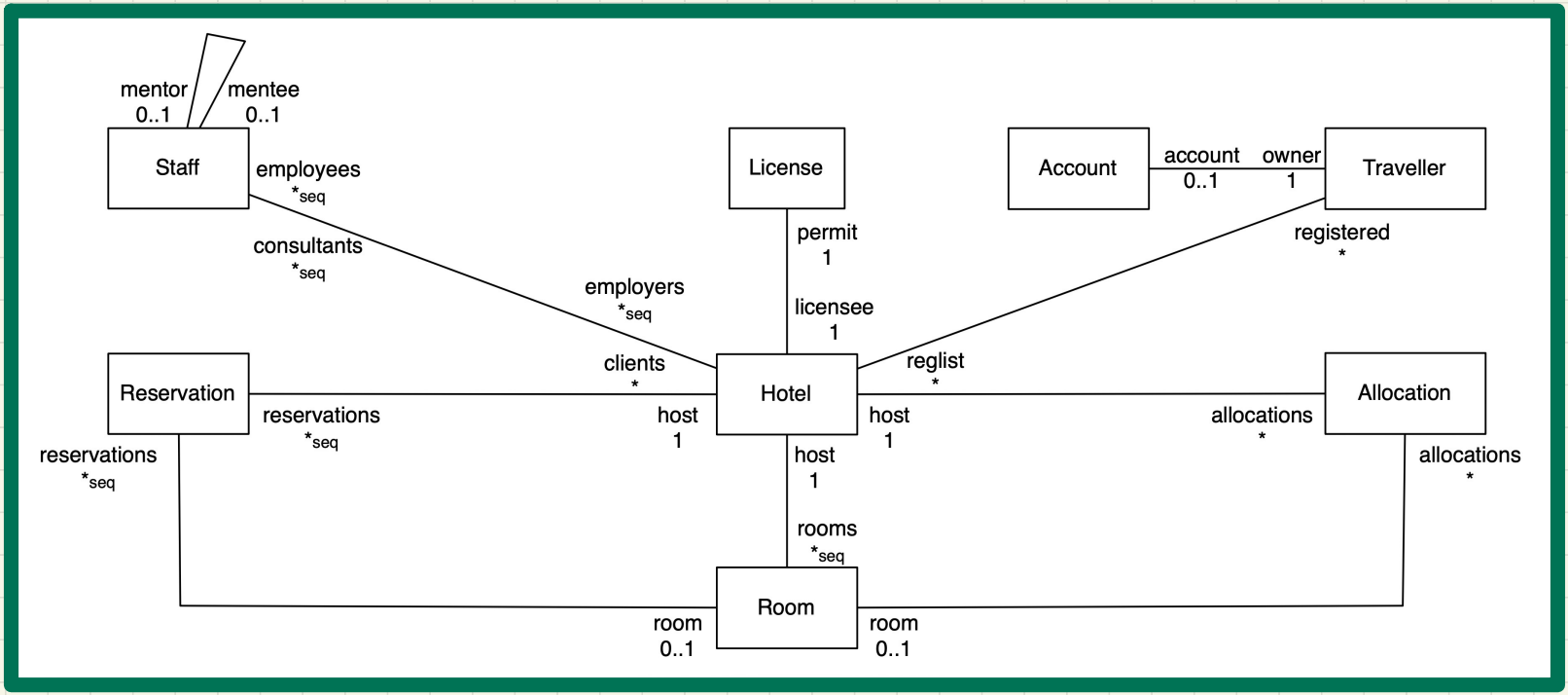
Lecture 3 - Sep. 15

Overview of Compilation

Example Compiler: Object-to-Relational
Introducing Scanner

Example Compiler 2: Data Model



Example Compiler 2: Mapping Data

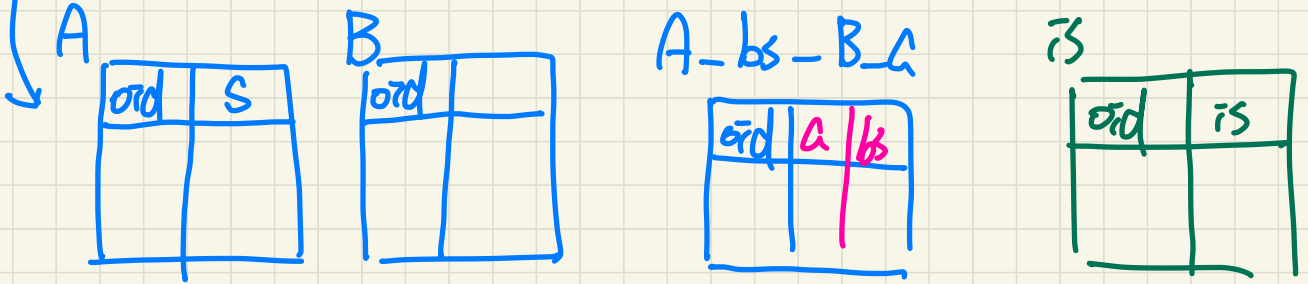
Attribute-to-Table Mapping

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

Example Transformation

```
class A {
  attributes
    s: string
    bs: set(B . a) [*] }
```

```
class B {
  attributes
    is: set(int)
    a: A . bs }
```



Example Compiler 2: 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}
    }
}
```

Scanner
- delimiter
- keywords

CFG for parser.

Method ::=
Id LCB
Exp
IMP
Exp

Example Compiler 2: Target Program

Account	
oid	balance
1	100

Traveller	
oid	name
2	alan
3	mark

Hotel	
oid	name
4	GLAD

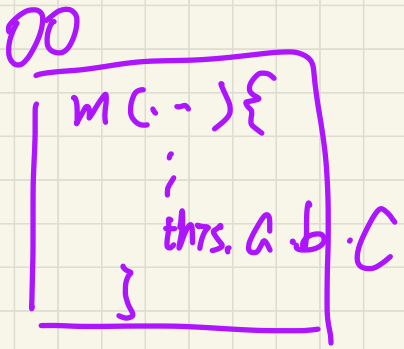
Account_owner_Traveller_account		
oid	owner	account
5	3	1

Hotel_registered_Traveller_reglist		
oid	registered	reglist
6	2	4
7	3	4

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`));
```

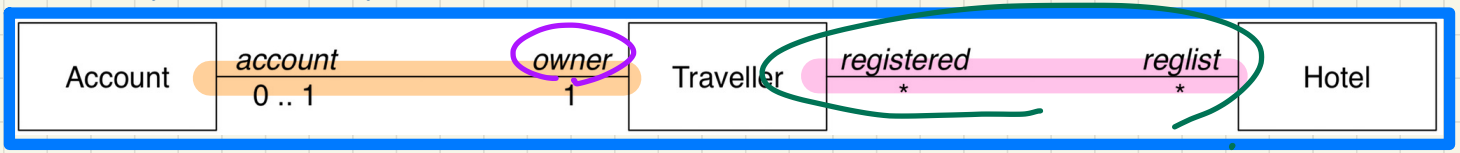
MySql.




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

Stored Procedures

Example Compiler 2: Path Transformation



Object Path

```
this.owner.reglist
```

oid == 4

oid == 3

Table Queries

```
SELECT (VAR 'reglist')
       (TABLE 'Hotel_registered_Traveller_reglist')
       (VAR 'registered' = (SELECT (VAR 'owner')
                           (TABLE 'Account_owner_Traveller_account')
                           (VAR 'owner' = VAR 'this')))
```

this == 1

Account	
oid	balance
1	100

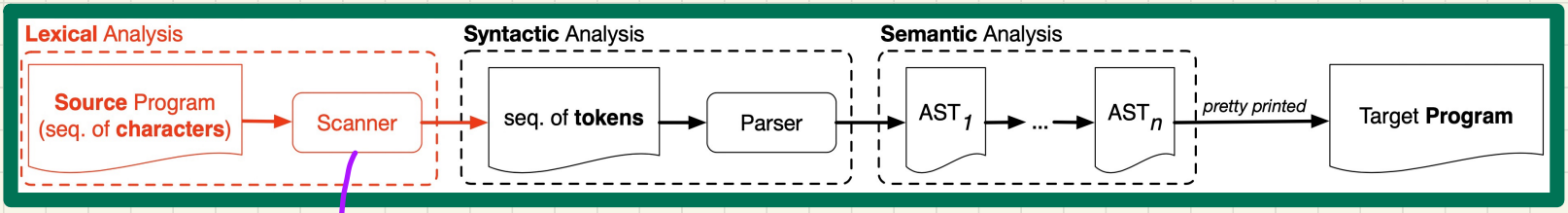
Traveller	
oid	name
2	alan
3	mark

Hotel	
oid	name
4	GLAD

Account_owner_Traveller_account		
oid	owner	account
5	3	1

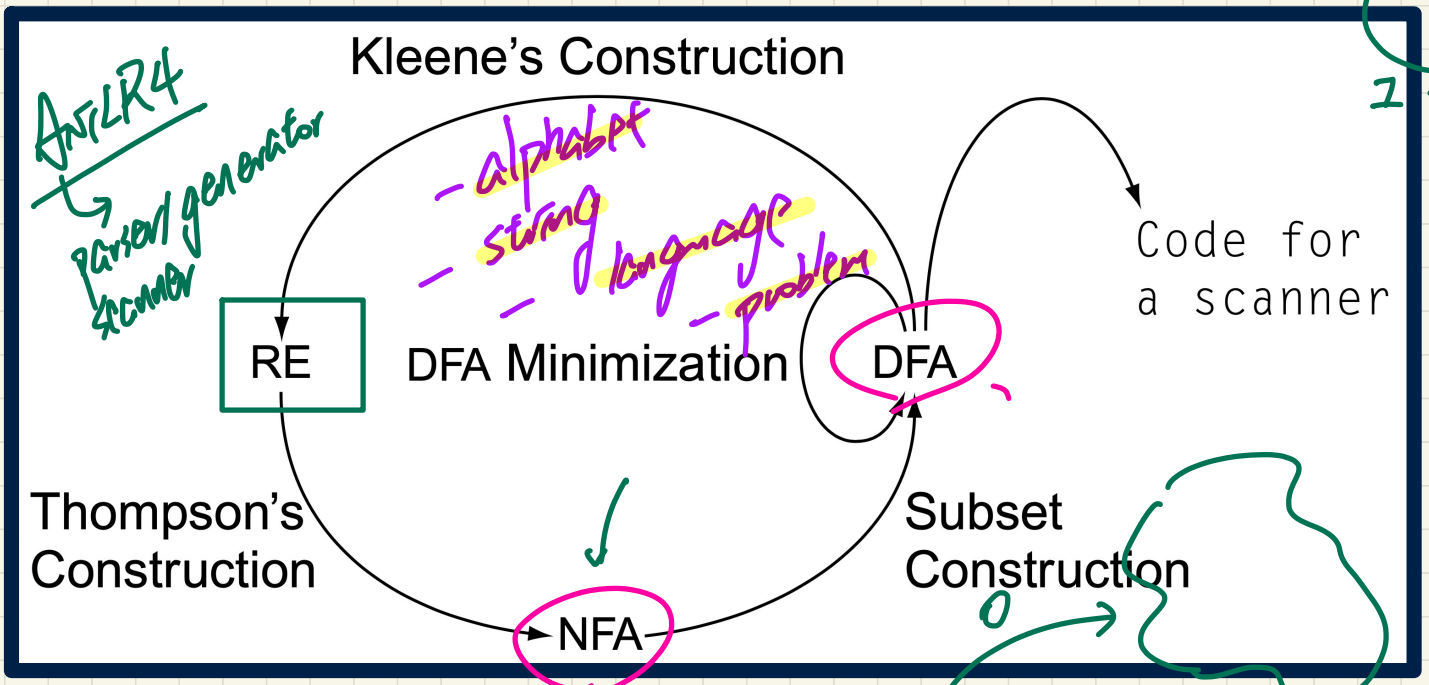
Hotel_registered_Traveller_reglist		
oid	registered	reglist
6	2	4
7	3	4

Scanner in Context



may also report error if there's any invalid char. seq.

Scanner: Formulation & Implementation



Set Comprehension

ε epsilon
↳ empty string.

$$\{ \underbrace{\quad}_{\text{expression}} \mid \underbrace{\quad}_{\text{predicates}} \}$$

↳ $\wedge$, $\vee$, $\neg$, $\Rightarrow$
↳ $\forall$, $\exists$

$$\Sigma_{dec} = \{ d \mid 0 \leq d \leq 9 \}$$

$\{0, 1\}$

$01010 \in \Sigma_{bin}$

string

alphabet

$\mathbb{N} = \{0, 1, \ldots\}$

natural

#

$P \wedge True \equiv P$

$P \vee false \equiv P$

op	identity
$+$	0
$*$	1
concat	ε
$\wedge$	true
$\vee$	false

$$\Sigma^k = \{ xy \mid x \in \Sigma^1 \wedge y \in \Sigma^{k-1} \}$$

$$\Sigma^k = \{ |w| \mid 0 \le |w| \le k \wedge ? \}$$

$|w|$ → #

not right ∵ the resulting set is a set of #'s

$$\Sigma^k = \{ \Sigma x \mid x \in \Sigma^{k-1} \}$$

not right ∵ concatenation only applies to two strings

w is a string of length k over Σ

$\equiv$

$$w = c_0 c_1 c_2 \cdots c_{k-1} \wedge \left(\forall i \cdot 0 \leq i \leq k-1 \Rightarrow c_i \in \Sigma \right)$$

$$\bigwedge_{0 \leq i < k} c_i \in \Sigma$$