syntax summary relations

\[ R \subseteq A \times B \]

over time, each \( A \) is mapped by \( R \) to the same \( B \)s

over time, \( R \) maps the same \( A \)s to each \( B \)

\[ R \text{ maps each } A \text{ to } n \text{ Bs} \]

\[ R \text{ maps } m \text{ As to each } B \]

+ one or more
* zero or more
! exactly one
? at most one
omitted = *
semantics examples
semantics examples
semantics examples
semantics examples
semantics examples
semantics examples
semantics examples

Digital Photography (Group)

Indian Food (Group)

Javascript Hackers (Group)

members

members

Bob (Member)

ε

Group

Member

Moderated Group

Moderator

messages

posts

respondsTo

? approved

? approves

members

members
semantics examples

Digital Photography (Group) members
Indian Food (Group) members
Javascript Hackers (Group) members

Bob (Member)

\[ \in \]

Diagram:

```
Group -> Member

Moderated Group -> Moderator

Moderator ? approves

posts ! messages

Member +

messages !

respondsTo ?
```

Members

Moderators
semantics examples
semantics examples

Ceramics (Moderated Group)

Alice (Moderator)
  posts
  respondsTo
  M1 (Message)

Bob (Member)
  posts
  respondsTo
  M2 (Message)

Moderated Group

Group
  members
  messages
  posts
  ?

Member
  members
  +
  posts
  !

Moderator
  ?
  ? approves
  +
  moderators

Message
  *
semantics examples

Ceramics (Moderated Group)

- moderators
- members

Alice (Moderator)
Bob (Member)

M1 (Message)
M2 (Message)

∈

Group
- members

Member

Moderated Group

Moderator

Message
- posts

respondsTo

messages

approves
semantics examples
common mistakes

#1. designation confusion
   arrivesAt: Elevator -> Floor
   elevator serves floor?
   is currently at floor?
   will arrive at, or has arrived at floor?

#2. should be split into multiple relations
   lines: Address -> AddressLine

#3. relates >2 atoms
   salary: Student -> Amount
   for which job?
mistake: including operations

a bad smell
› using verbs, especially in present tense

bad:

User \(\rightarrow\) reads \(\rightarrow\) writes \(\rightarrow\) Post

better:

User \(\leftarrow\) author \(!\) \(\rightarrow\) Post
mistake: inadequacy

think about each feature
› is there enough state to answer each query?
› to decide which actions are allowed?

bad:

better:
mistake: lack of generalization

are two sets really subsets of a common set?
› look for duplication of relations

bad:

better:
mistake: implementation details

beware of
› collection objects: replace by relation?
› singleton objects: replace by set?
› status: replace by subsets?

bad:

better:
**composite**

- **FSObject**
  - File
  - Directory

- **Element**
  - Shape
    - Rectangle
    - Oval
  - Group

- **Document**
  - Head
  - Body
  - Element
    - Block
      - Div
      - Para
    - Inline

- **Contains**

- **Elements**
tuple

employee-company-position
› a 3-way relationship, expressed with Job tuple
lowering

users with IP addresses
› strengthen multiplicity
› define subset of active users that have IPs
another lowering

before

Member ! posts Message

? approves Moderator

Member ! posts Message

Moderator ! approves Approved Message

Pending Message

after
lifting

generalize and move relation to larger set
› from Alum/Student to Member
› from Shape/Slide to Component
graphical limitations
graphical limitations

consider this instance

› Alice is not a member of Ceramics
graphical limitations

consider this instance
› Alice is not a member of Ceramics

can we say moderators should be members?
› not in this graphical notation
delving into semantics

Diagram:

- **Group**
  - **Moderated Group**
  - **Moderators**
  - **Members**

- **Member**
  - **Moderator**
  - **Members**
  - **Moderators**
delving into semantics

A group's moderators must be in the set Moderator.
Only a group in ModeratedGroup can have moderators. A group's moderators must be in the set Moderator.
only a group (ie, not a message, eg) can have members

only a group in ModeratedGroup can have moderators

a group’s moderators must be in the set Moderator
only a group (ie, not a message, eg) can have members

only a group in ModeratedGroup can have moderators

a group’s members must be in the set Member

a group’s moderators must be in the set Moderator

delving into semantics
only a group (ie, not a message, eg) can have members

only a group in ModeratedGroup can have moderators

a group’s members must be in the set Member

every Moderator is a Member

a group’s moderators must be in the set Moderator

delving into semantics
only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

every Moderator is a Member.

a group’s members must be in the set Member.

only a group in ModeratedGroup can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.

only a group (i.e., not a message, e.g.) can have members.

every ModeratedGroup is a Group.

only a group in ModeratedGroup can have moderators.

a group’s moderators must be in the set Moderator.

a group’s members must be in the set Member.
only a group (i.e., not a message, eg) can have members

a group’s members must be in the set Member

every Moderator is a Member

but not every moderator of g is a member of g

only a group in ModeratedGroup can have moderators

a group’s moderators must be in the set Moderator

every ModeratedGroup is a Group

only a group in ModeratedGroup can have moderators
textual constraints

- moderators must be members of the group
- member only posts message in group she belongs to
- moderators approve messages in groups they moderate
- message only responds to message in same group

in this course, just informal text; more advanced: express in Alloy
in mongo: still choices!

**embedded**

```json
{
    title: "Fury",
    time: "7:00pm",
    theater: {
        name: "West Newton Cinema",
        location: "Newton"
    }
}
```

one document in the collection Movies

**relational**

```json
{
    title: "Fury",
    time: "7:00pm",
    theater: 1
}
```

one document in the collection Movies

```json
{
    _id: 1,
    name: "West Newton Cinema",
    location: "Newton"
}
```

one document in the collection Theaters
repertoire of design moves

basic moves
› reverse relation
› add/remove object
› nest objects
› choose key

other moves
› add redundancy
› add index
making a nested structure

```
{title: "Fury"}
{zip: "02139"}
{location: {zip: "02139"},
  name: "Kendall"}
{
  theater: {location: {zip: "02139"},
            name: "Kendall"},
  movie: {title: "Fury"},
  startTime: "7pm",
}
```
reversing a relation

```json
{title: "Fury"}

{zip: "02139"}

{location: {zip: "02139"},
  name: "Kendall"}

{
  theater: {location: {zip: "02139"},
                       name: "Kendall"},
  movie: {title: "Fury"}
}

{
  time: "7pm",
  showings: [
    theater: {location: {zip: "02139"},
                       name: "Kendall"},
    movie: {title: "Fury"}
  ]
}
```
relating structures

{title: "Fury"}

{zip: "02139"}

{location: {zip: "02139"},
  name: "Kendall"}

{
  startTime: "7pm",
  theater: 2,
  movie: 1
}

{__id: 1, title: "Fury"}

{__id: 2, location: {zip: "02139"},
  name: "Kendall"}