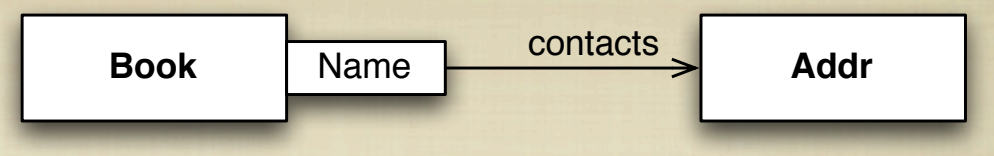
# Alloy Operators

Curt Clifton
Rose-Hulman Institute of Technology

# Goals for Today

- Recall more set theory ideas and see how they're used in Alloy
- Learn some Alloy operators that we'll use to describe designs
- Get more comfortable experimenting with Alloy to see how it behaves with an example

# Constants




- **none** – the empty set
- **univ** – the universal set
  - Contains one of every atom in the instance
- **iden** – the identity relation
  - Maps every atom to itself
  - Including Int atoms -8, -7, ..., -1, 0, 1, ..., 6, 7

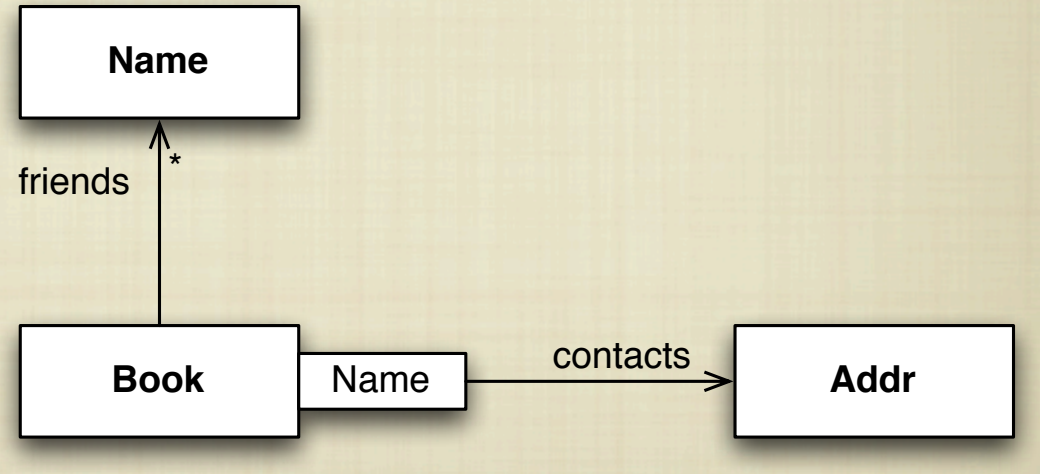
In every instance, unless we change "bit width"

Q1,2

# Set Operators

- Set valued:
  - + – union
  - & – intersection
  - - – difference
- Boolean valued:
  - in – subset
  - = – equality

All of these apply to any pair of relations with the same arity




Q3

# Relational Operators

- Combining relations
  - -> – arrow product
  - . – dot join
  - [] – box join
- Reachability
  - ^ – transitive closure
  - * – reflexive-transitive closure
- "Modifying" relations
  - ~ – transpose
  - <: – domain restriction
  - :> – range restriction
  - ++ – override

# Combining Relations: Arrow Product

- `p -> q` – every possible way to concatenate a tuple from p and a tuple from q
- Examples in Alloy...

# Combining Relations: Dot Join


- Also called "composition"
- Written: p.q

# Dot Join on Tuples

- {(N0,**A0**)} . {(**A0**,D0)} = {(N0,D0)}
- {(N0,**D0**)} . {(**N0**,D0)} = {}
- {(N0,**D0**)} . {(**D1**)} = {}
- {(N0,**D0**)} . {(**D0**)} = {(N0)}
- {(**N0**)} . {(**N0**,D0)} = {(D0)}
- {(**B0**)}.{(**B0**,N0,D0)} = {(N0,D0)}

Recall: this is the "math" font

Q5

# Dot Joins on Relations

- Suppose **p** and **q** are relations
- **p.q** is the result of taking every combination of a tuple from **p** and a tuple from **q** and including their join

Q6

# Combining Relations: Box Join

- Exactly the same meaning as dot join
- Syntactic sugar—looks like array access
- Written: q[p], means p.q
  - Typically p will be a scalar

# "Modifying" Relations: Transpose

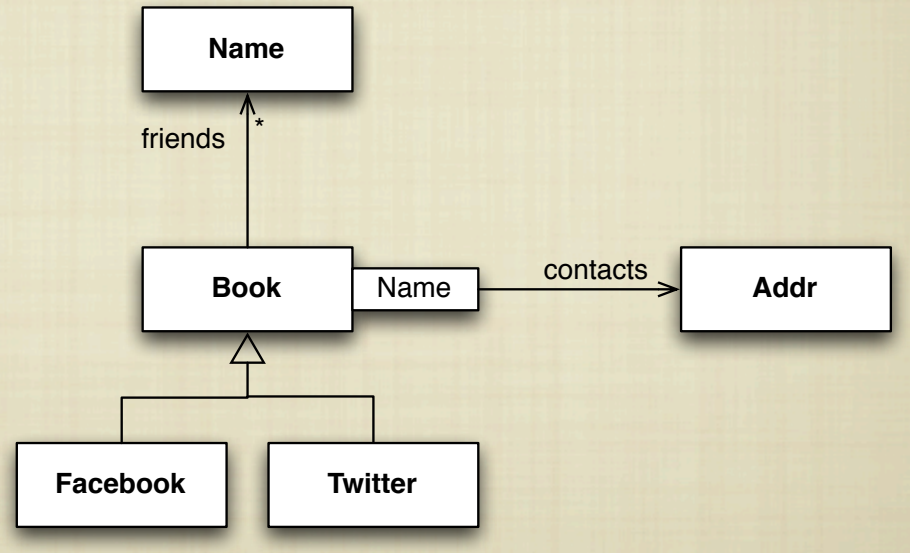
- ~p – transpose
  - Yields a new relation by flipping order of tuples in p
  - p must be binary
  - Useful for getting order right before a join

Example: Of whom is Name$2 a friend?

Q7

# "Modifying" Relations: Restrict domain/range

- s <: r, domain restriction
  - Yields a new relation by throwing out tuples in r that don't start with an element of s
  - s must be a set
  - Useful for focusing on "subclasses"
  - Unlike dot join, it keeps all columns
- r :> s, range restriction
  - like <: but matches last element of r's tuples



# Next Time

- Techniques for telling Alloy what must be true about a design:
  - Override operator
  - Reachability operators
  - Constraints

# Alloy Models for Reference

# Model from class

```
abstract sig Book {
	contacts: Name -> lone Addr,
	friends: set Name,
	owner: lone Name
}
sig Name, Addr {}

sig Twitter, Facebook extends Book {}

pred show[b1, b2:Book] {
	b1 != b2
	#b1.contacts > 1
	#b2.contacts > 1
}

run show for 3 but 2 Book

pred addContact[b, b': Book, n: Name, a: Addr] {
	b != b'
	some Facebook & b iff some Facebook & b'
	b'.contacts = b.contacts ++ n->a
}

pred addContacts[b, b': Book, c: Name -> lone Addr] {
	b != b'
	some Facebook & b iff some Facebook & b'
	b'.contacts = b.contacts ++ c
}

run addContacts for 4 but 2 Book
```