TYPES FOR ALLOY:
FROM PROGRAMMING TO MODELLING
TYPES FOR ALLOY: FROM PROGRAMMING TO MODELLING

(More Precisely ...)

TYPES FOR FROGLET
Imagine .... the year is 19XX

Your DVD rental website is doing great business
Imagine .... the year is 19XX

Your DVD rental website is doing great business

Until evil hackers find a way to redirect orders (CSRF)

What to do?
Imagine ..., the year is 19XX

Your DVD rental website is doing great business

Until evil hackers find a way to redirect orders (CSRF)

What to do?

Idea: track the origin of every HTTP Event

Q. Does it work?
PROGRAMMING IS HARD!

"Does the idea work?" is a difficult question to answer!
PROGRAMMING IS HARD!

"Does the idea work?" is a difficult question to answer!

Code has to say:
- WHAT to do, and
- HOW to do it
"Does the idea work?" is a difficult question to answer!

Code has to say:
- WHAT to do, and
- HOW to do it

Alloy lets you focus on the WHAT
Alloy for Debugging Designs

Alloy model

// Data definition
sig Request extends HTTPEvent
  response: lone Response

// Predicate
pred Acyclic [r: Request]
  r not in r.^{response.embeds}

// Conjecture
check { no good, evil: Server | .... }
Alloy for Debugging Designs

Alloy model

// Data definition
sig Request extends HTTPEvent
    response: lone Response

// Predicate
pred Acyclic [r: Request]
    r not in r.^(response.embeds)

// Conjecture
check { no good, evil: Server | .... }

Running a model kicks off a search for counterexamples

For our DVD website, a counterexample SHOWS how a forgery can happen – even if we track one origin
Step 1: Outline your data structures
Step 2: Write formal properties
Step 3: Study the counterexamples
ALLOY FOR DEBUGGING DESIGNS

Step 1: Outline your data structures
Step 2: Write formal properties
Step 3: Study the counterexamples

Lots of successes: Amazon, AT&T, Netflix
ALLOY FOR DEBUGGING DESIGNS
ALLOY FOR DEBUGGING DESIGNS

Great!
Alloy for Debugging Designs

Great!

... Why doesn't ALL production software come with an Alloy model?

One reason: models are tricky to write!
/\* Data definition \*/
sig Request extends HTTPEvent  
  response: lone Response

/* Predicate */
pred Acyclic [r: Request]  
  r not in r.^({response.embeds})

/* Conjecture */
check { no good, evil: Server | .... }
// Data definition
sig Request extends HTTPEvent
    response: lone Response

// Predicate
pred Acyclic [r: Request]
    r not in r.^{(response.embeds)}

// Conjecture
check { no good, evil: Server | .... }

- r is a set
- response is a relation
- . is relational join
- ^ is transitive closure
ALLOY GOTCHAS

In short:

1. Alloy is scripting for set theory
   - Very forgiving of type mixups

2. Alloy is NOT a programming language
   - Intuitions can be misleading
logic FOR systems
RQ. How to teach Alloy to programmers?
RQ. How to teach Alloy to programmers?

Important for pros, too
VISION: LANGUAGE LEVELS
VISION: LANGUAGE LEVELS

Why? Tailored feedback!

Alloy
VISION: LANGUAGE LEVELS

Why? Tailored feedback!
Vision: Language Levels

Why? Tailored feedback!
A functions-first subset of Alloy

- Models contain only:
  + data structures
  + functional relations

- a.b is a field access

- No relational algebra
A functions-first subset of Alloy

- Models contain only:
  + data structures
  + functional relations

- a.b is a field access

- No relational algebra

Q. Easy to learn?
FROGLET 2022

A functions-first subset of Alloy

- Models contain only:
  + data structures
  + functional relations

- `a.b` is a field access

- No relational algebra

Q. Easy to learn?

Ok so far, for Java ppl
FROGLET 2022

A functions-first subset of Alloy

- Models contain only:
  + data structures
  + functional relations
- a.b is a field access
- No relational algebra

Q. Easy to learn?
Ok so far, for Java ppl

Q. Too restrictive?
A functions-first subset of Alloy

- Models contain only:
  + data structures
  + functional relations

- a.b is a field access

- No relational algebra

Q. Easy to learn?
Ok so far, for Java ppl

Q. Too restrictive?
Yes ... needs support
FROGLET 2022

A functions-first subset of Alloy

- Models contain only:
  + data structures
  + functional relations

- a.b is a field access

- No relational algebra

Q. Easy to learn?
  Ok so far, for Java ppl

Q. Too restrictive?
  Yes ... needs support
Types to catch and explain mistakes

- Goal: be like Java
  + build on intuitions

- a.b checks for valid fields

- Allow untyped libraries
/ Data definition
sig Request extends HTTPEvent
  response: lone Response

// Predicate
pred Acyclic [r: Request]
  r not in r.^{response.embeds}

// Conjecture
check { no good, evil: Server | .... }
// Data definition
sig Request extends HTTPEvent
response: lone Response

// Predicate
pred Acyclic [r: Request]
  r not in r.^((response.embeds)

// Conjecture
check { no good, evil: Server | .... }
// Data definition
sig Request extends HTTPEvent
    response: lone Response

// Predicate
pred Acyclic [r: Request]
    r not in r.^("response.embeds")

// Conjecture
check { no good, evil: Server | .... }
Typed Froglet (2023)

Alloy model

// Data definition
sig Request extends HTTPEvent
  response: lone Response

// Predicate
pred Acyclic [r: Request]
  r not in r.^((response.embeds)

// Conjecture
check { no good, evil: Server | .... }
Instead, use a library:
not reachable[r, r, response, embeds]
Challenge: What's the type for reachable?

Instead, use a library:

`not reachable[r, r, response, embeds]`
Challenge: What's the type for reachable?
**Challenge:** What’s the type for reachable?

**Challenge:** Are simple types enough?
Challenge: What's the type for reachable?

Challenge: Are simple types enough?

Challenge: Will types help us migrate to Alloy?
FROGLET TO ALLOY
Froglet to Alloy

Pop Quiz: What should this do?

CS2 in prereqs.CS1
(Valid in Alloy)
FROGLET TO ALLOY

Pop Quiz: What should this do?

CS2 in prereqs.CS1
(Valid in Alloy)
FROGLET TO ALLOY

Pop Quiz: What should this do?

CS2 in prereqs.CS1 (Valid in Alloy)

"Should totally have an error, because this syntax makes no sense"
FROGLET TO ALLOY

Pop Quiz: What should this do?

CS2 in prereqs.CS1 (Valid in Alloy)

"Should totally have an error, because this syntax makes no sense"

"This one just seems like it is wrong"
FROGLET TO ALLOY

Pop Quiz: What should this do?

CS2 in prereqs.CS1

(Valid in Alloy)

"Should totally have an error, because this syntax makes no sense"

"This one just seems like it is wrong"

"Good riddance! What a travesty that would be"
FROGLET TO ALLOY

Pop Quiz: What should this do? CS2 in prereqs.CS1 (Valid in Alloy)

"Should totally have an error, because this syntax makes no sense"

"This one just seems like it is wrong"

"Good riddance! What a travesty that would be"

"No I just... no.... what's even happening...."
FROGLET TO ALLOY

Typed Froglet

Froglet

Alloy
FROGLET TO ALLOY

Types 'n Sets

Typed Froglet

Alloy
THE END
Goal: Java-style types to help programmers learn Alloy

Some Challenges:
- How to type libraries?
- Are Java types enough?
- Will types help teach sets?

Big Idea: Language Levels