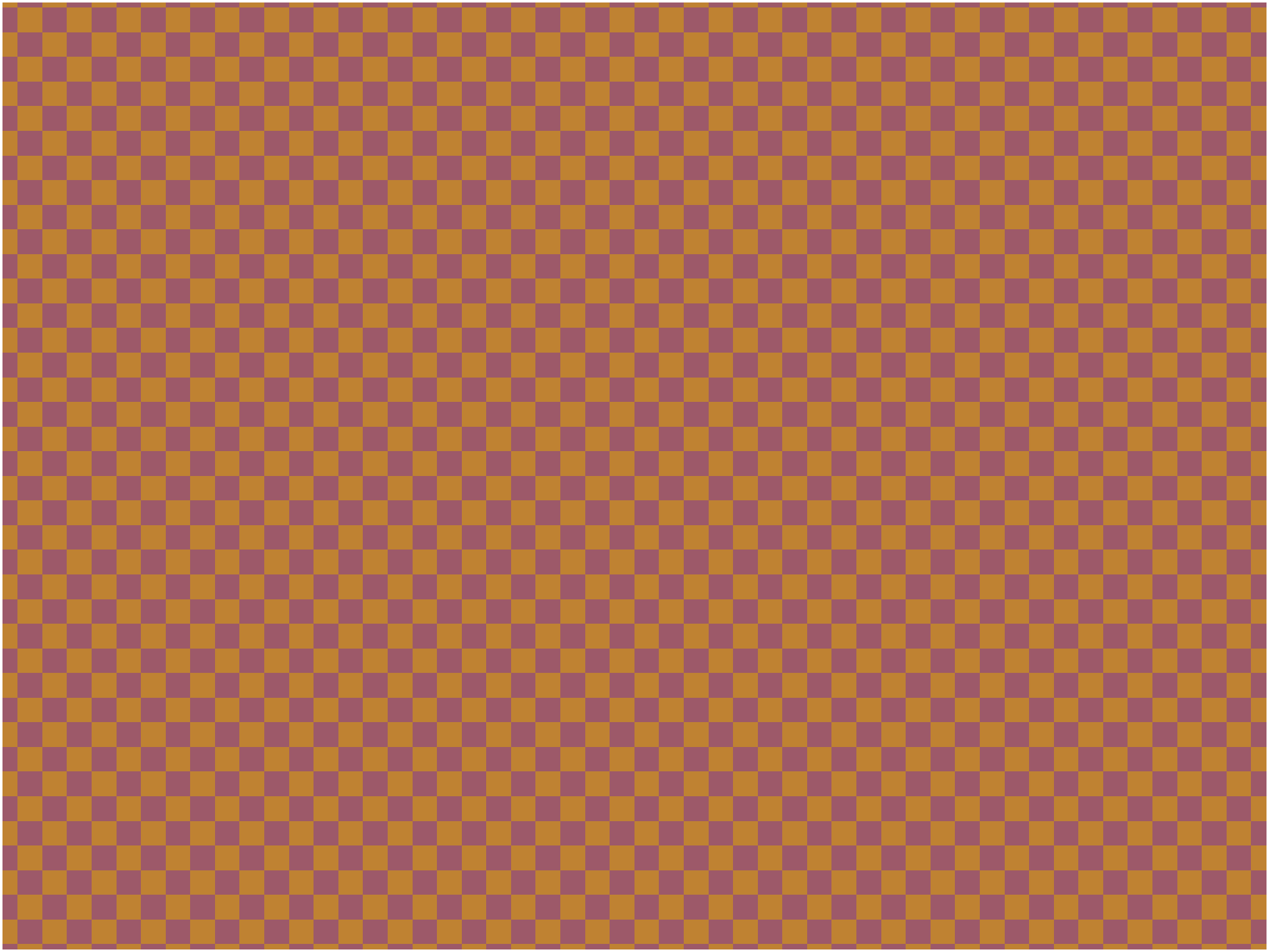


TYPES FOR ALLOY: FROM PROGRAMMING TO MODELLING

BEN GREENMAN

2022-09-09

BROWN UNIVERSITY

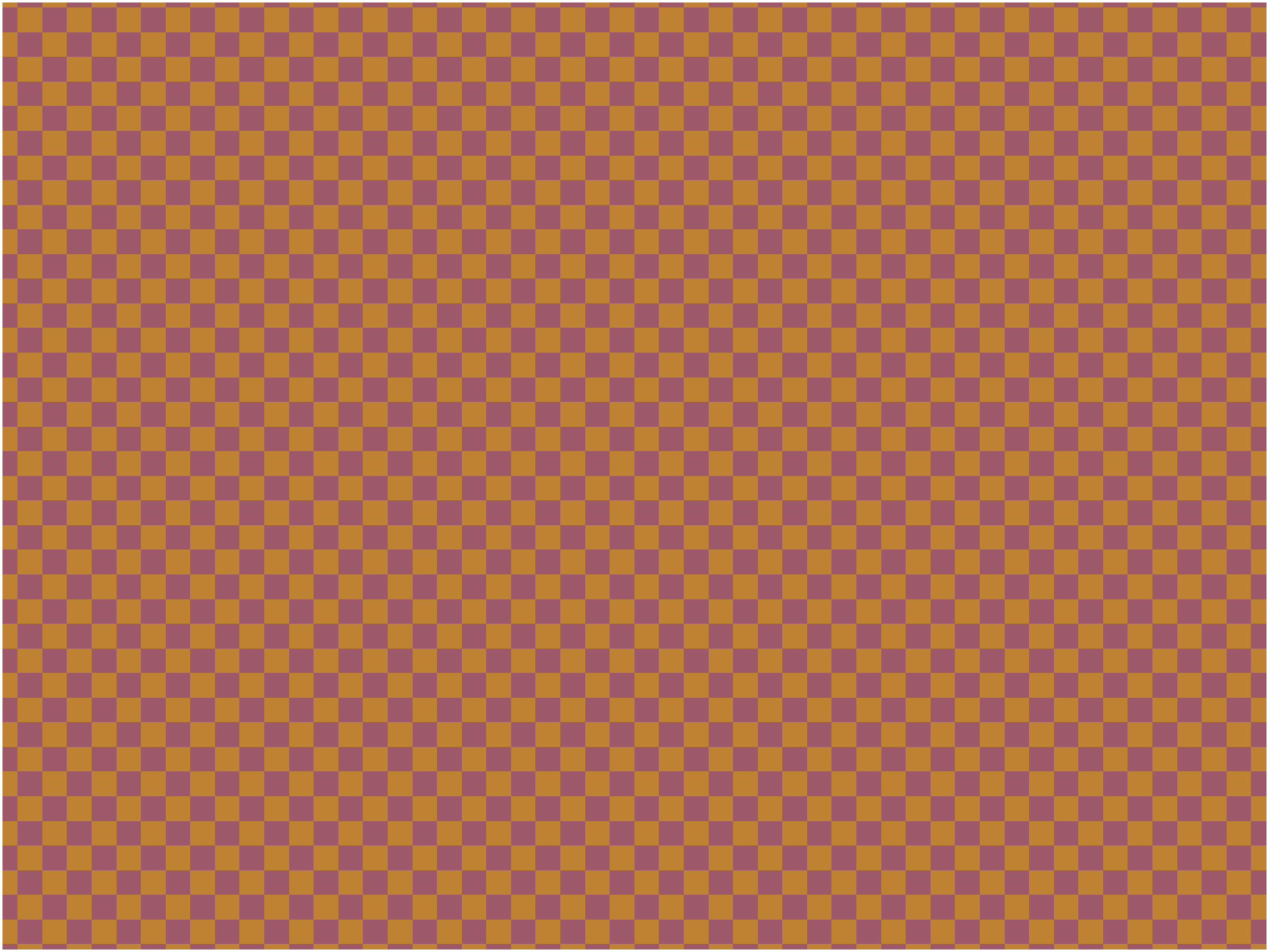
TYPES FOR ALLOY:

FROM PROGRAMMING TO MODELLING

TYPES FOR ALLOY:
FROM PROGRAMMING TO MODELLING

(More Precisely ...)

TYPES FOR FROGLET



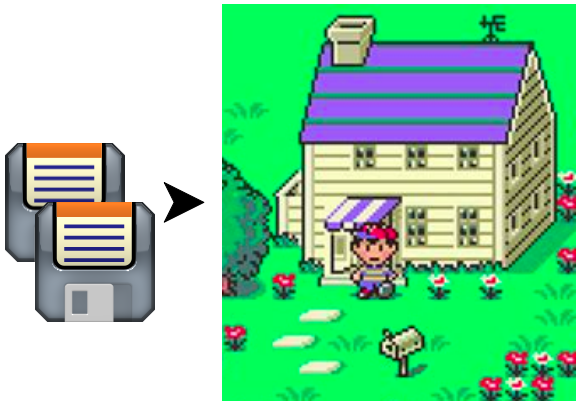
PROGRAMMING IS HARD!

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?



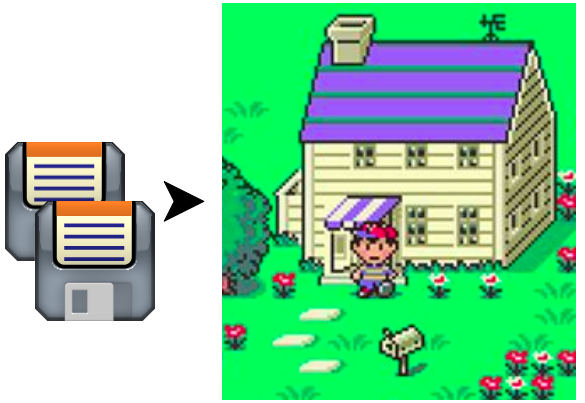
PROGRAMMING IS HARD!

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

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

Alloy lets you focus on the WHAT

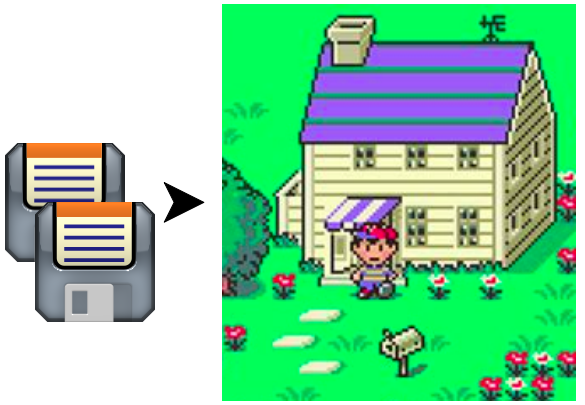
ALLOY FOR DEBUGGING DESIGNS

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 | ... }
```

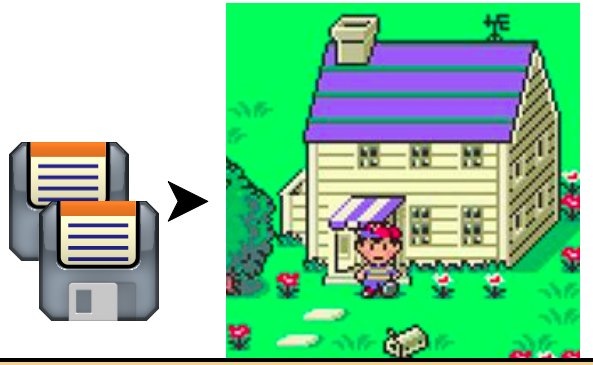


ALLOY FOR DEBUGGING DESIGNS

Alloy model

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

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

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!

Alloy gotchas

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 gotchas

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 | .... }
```

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

Alloy Gotchas

Alloy gotchas

Takeaways:

1. Alloy is a scripting tool for set theory
   – Very forgiving of type mixups

2. Alloy is NOT a programming language
   – Intuitions can be misleading


logic for systems

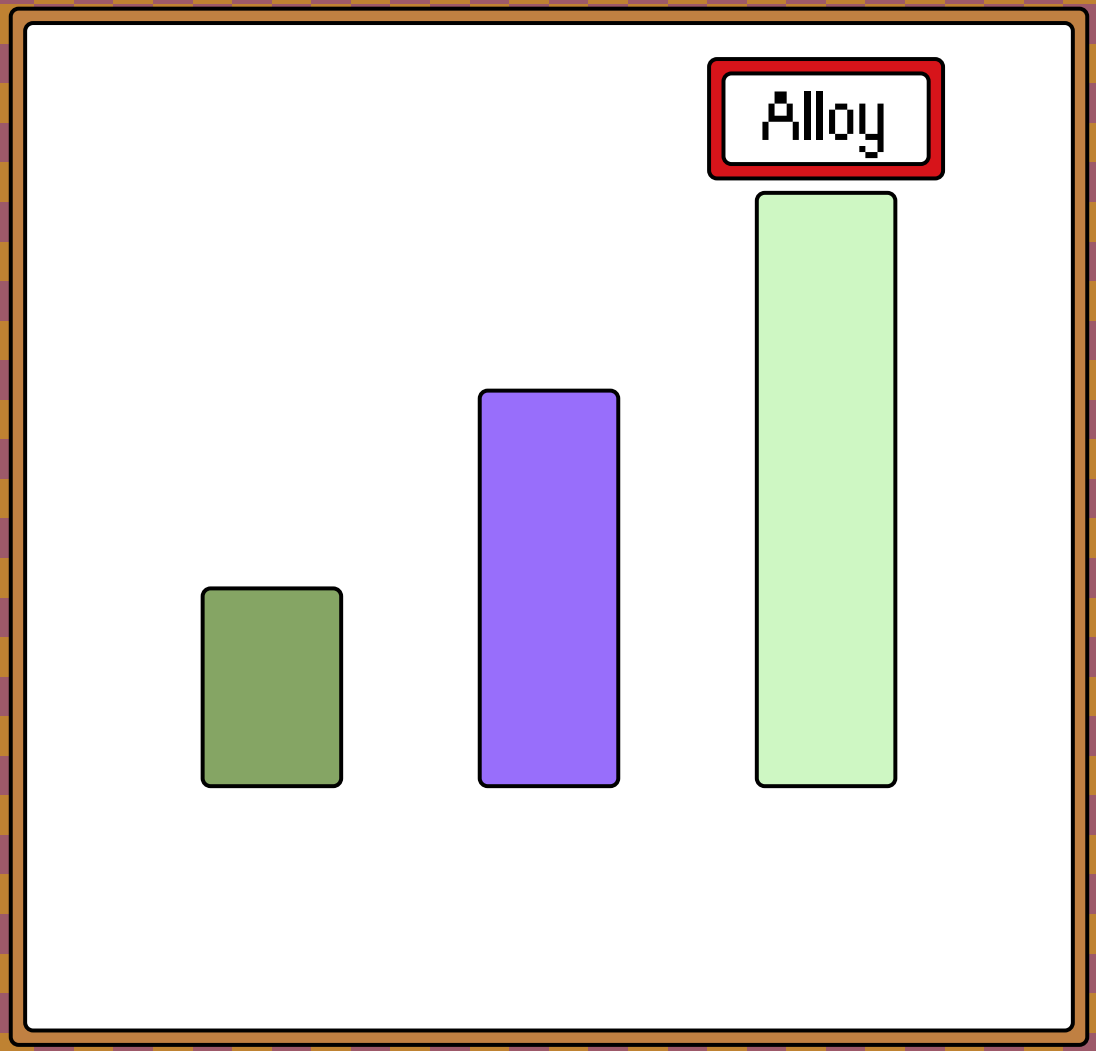
RQ. How to teach Alloy to programmers?

RQ. How to teach Alloy to programmers?

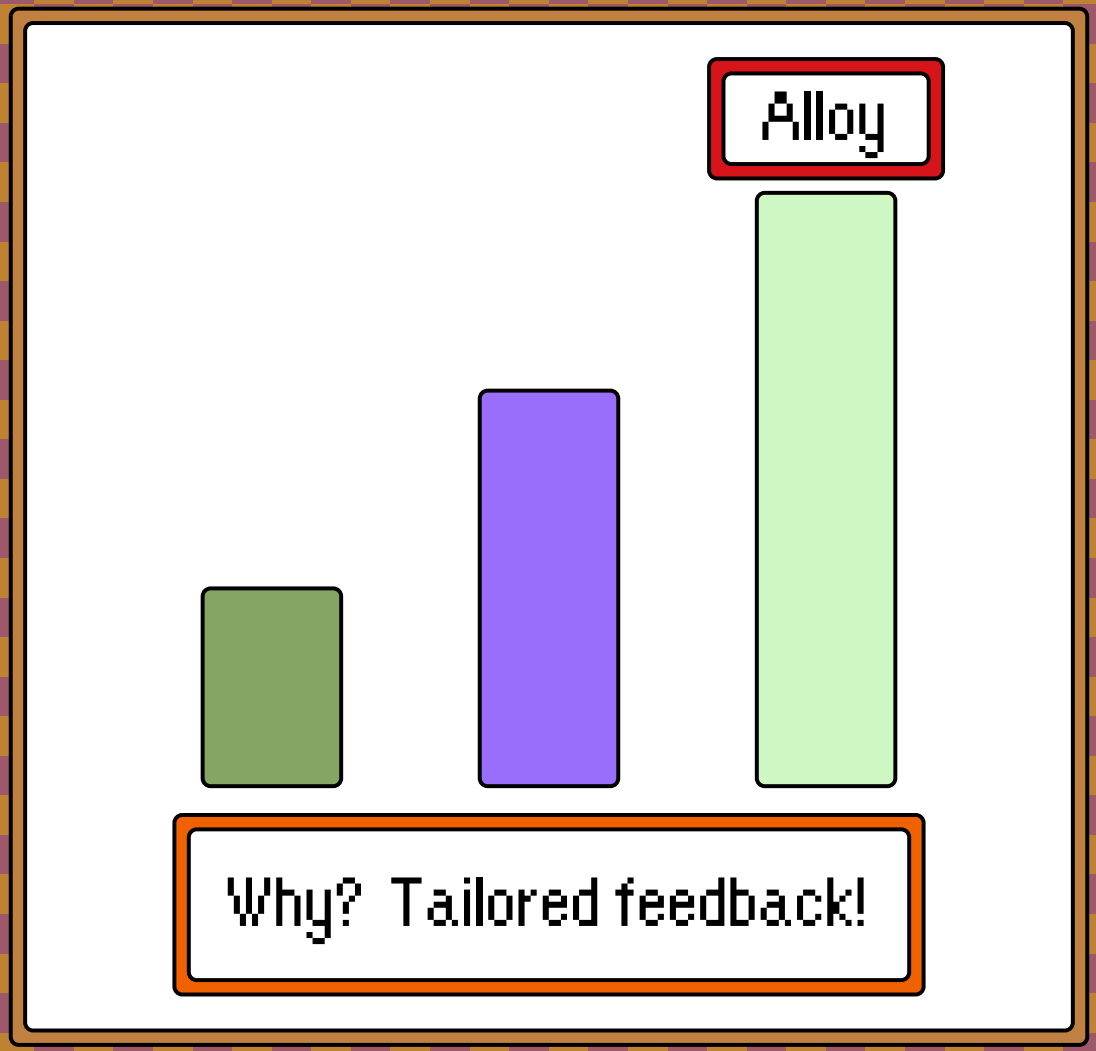
Important for pros, too

VISION: LANGUAGE LEVELS

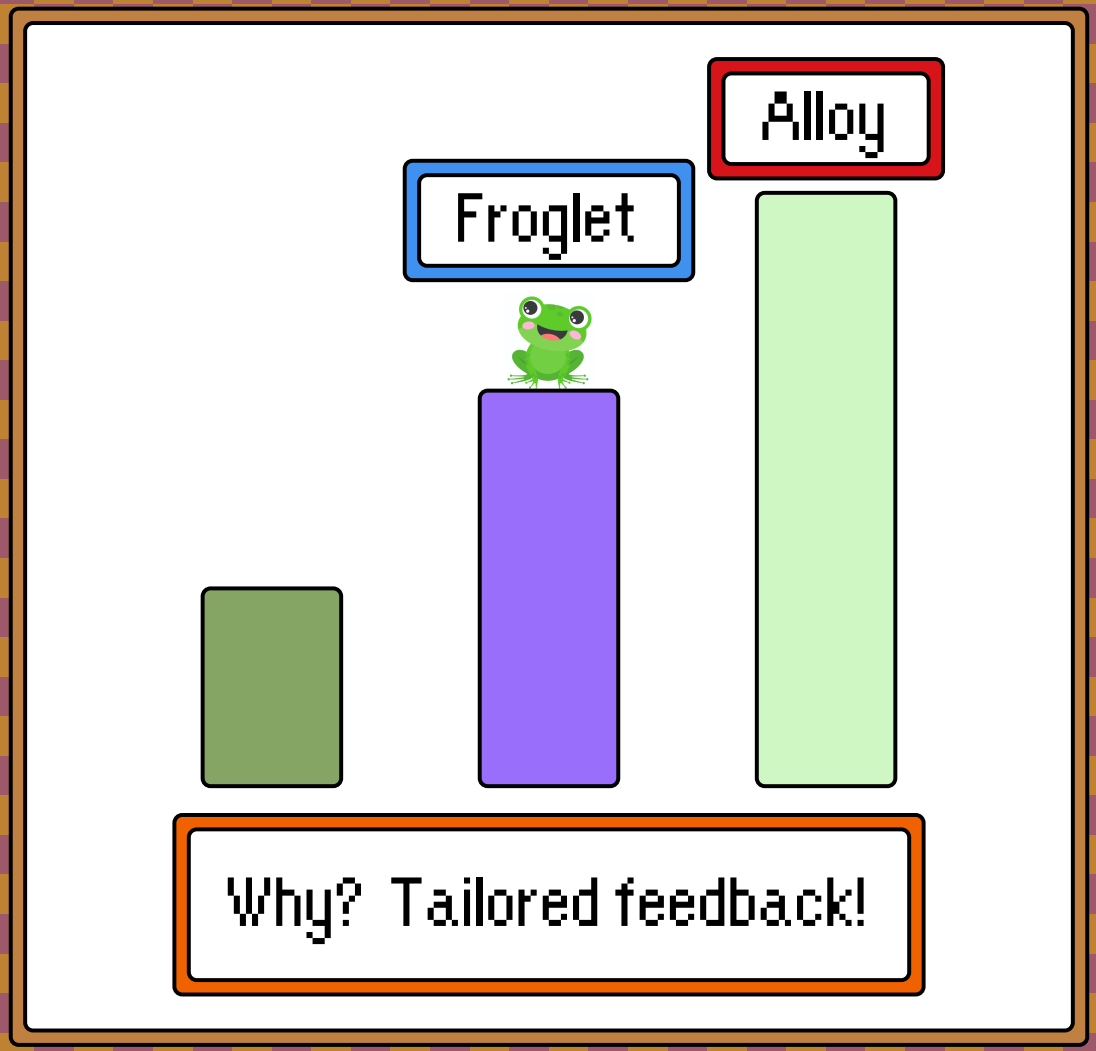
VISION: LANGUAGE LEVELS



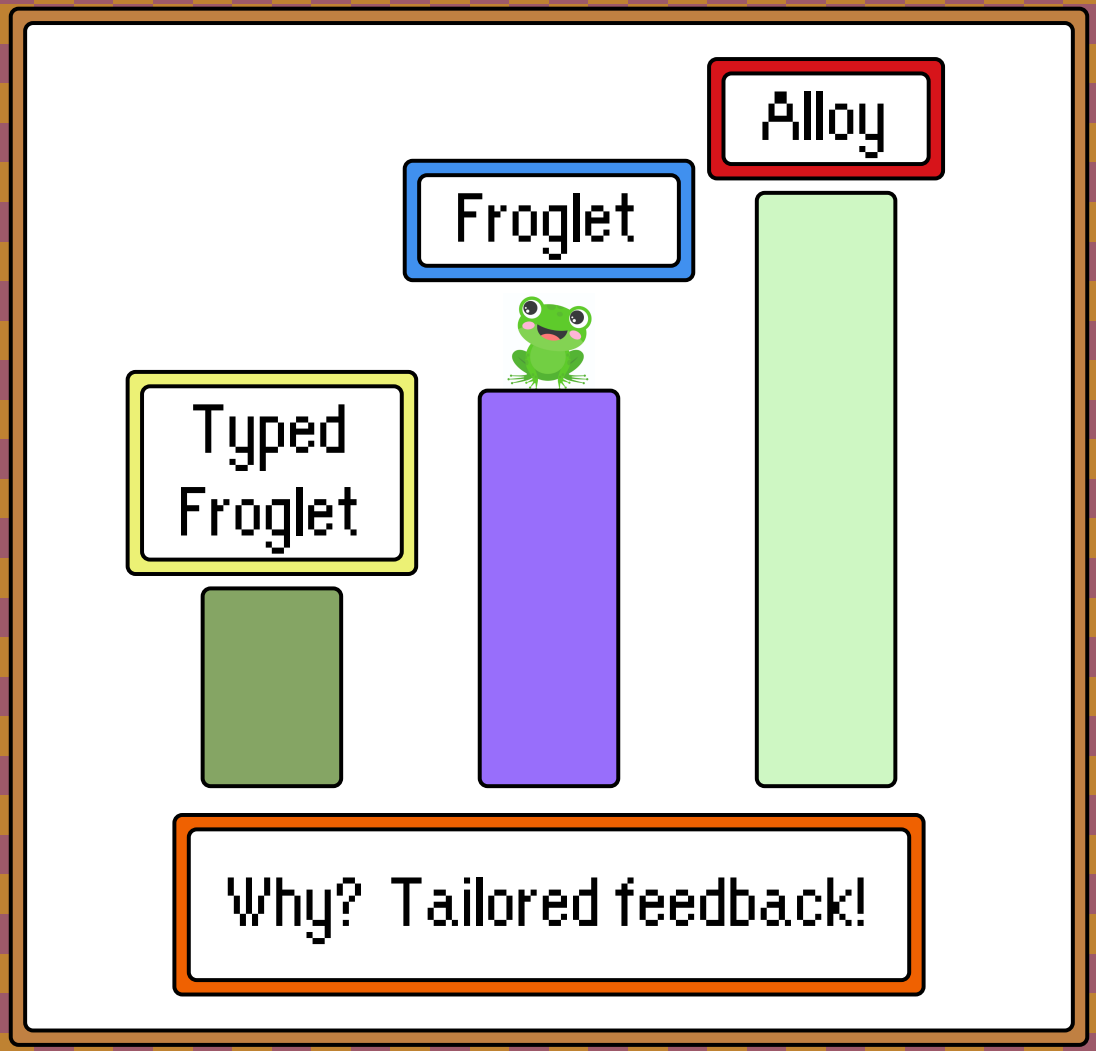
VISION: LANGUAGE LEVELS

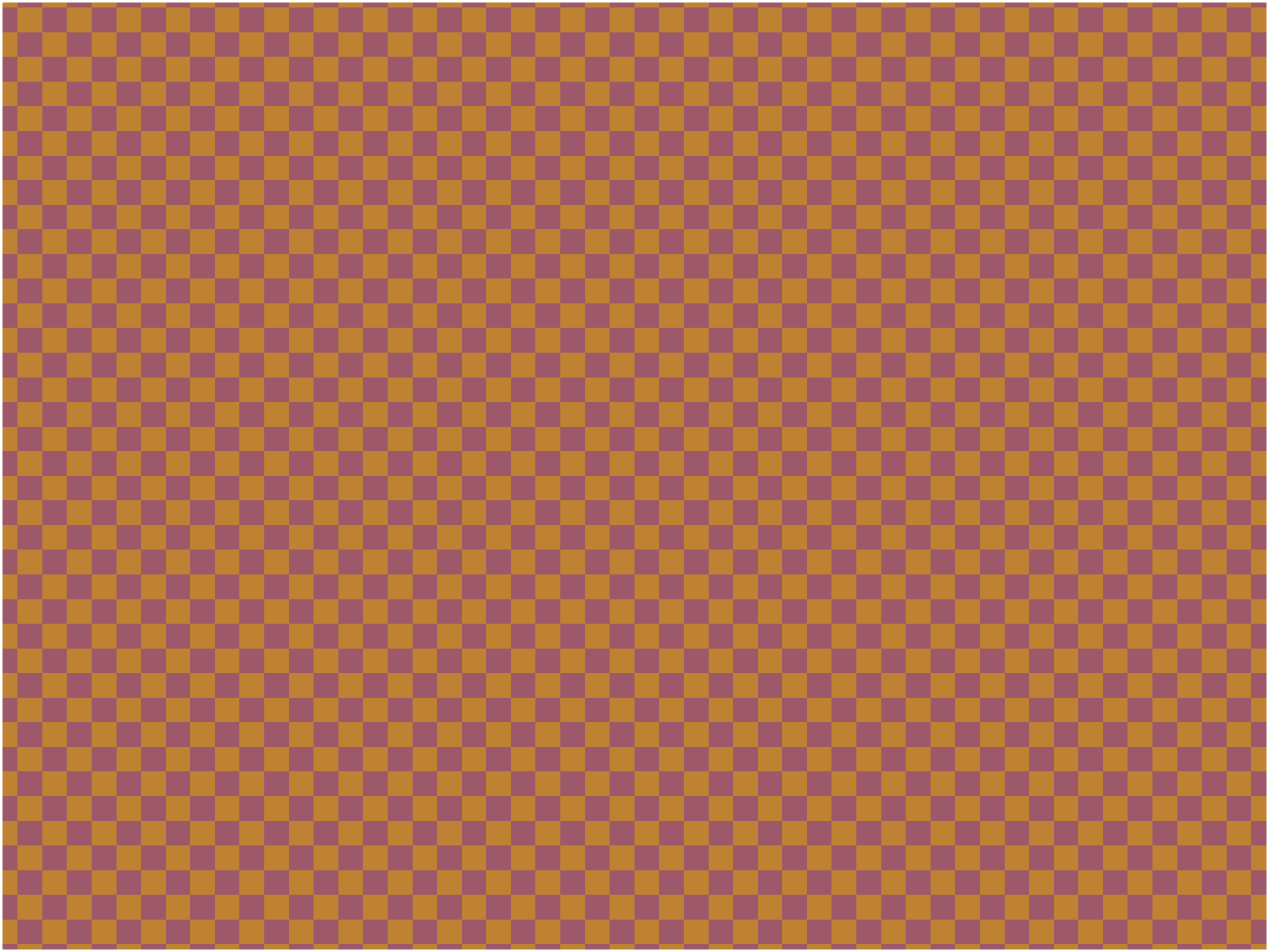


VISION: LANGUAGE LEVELS



VISION: LANGUAGE LEVELS





FROGLET 2022

FROGLET 2022

A functions-first subset of Alloy

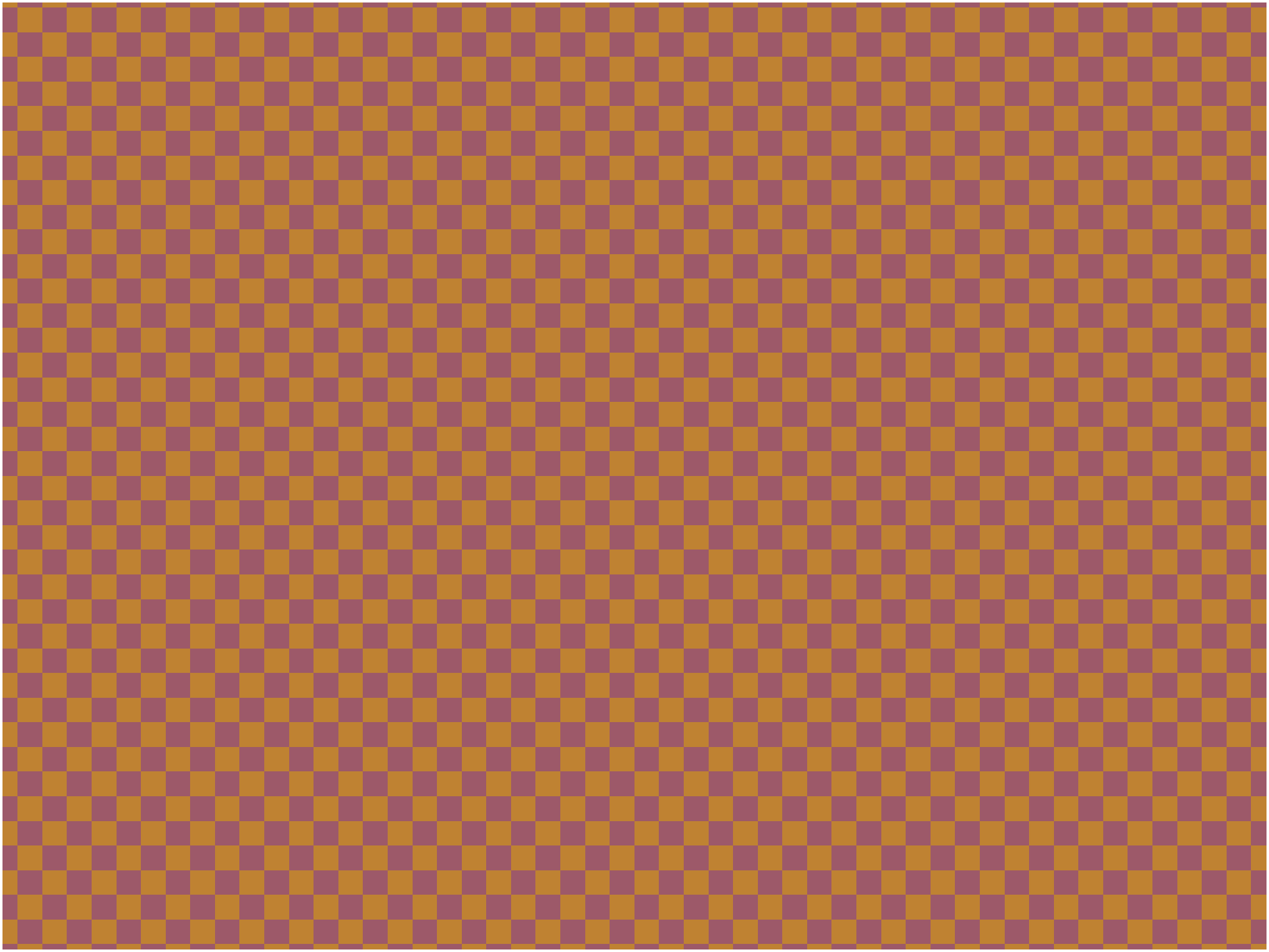
- Models contain only:
  - + data structures
  - + functional relations
- a.b is a field access
- No relational algebra

Q. Easy to learn?

Ok so far, for Java ppl

Q. Too restrictive?

Yes ... needs support



TYPED FROGLET (2023)

TYPED FROGLET (2023)

Types to catch and explain mistakes

- Goal: be like Java
  + build on intuitions

- a.b checks for valid fields

- Allow untyped libraries

TYPED FROGLET (2023)

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 | ... }
```

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 | .... }
```

Type Error

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 | .... }
```

Type Error
response does not have fields

TYPED FROGLET (2023)

Alloy model

```
// Data d
sig Reque
  respons

// 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]

Type Error

response does not have fields

TYPED FROGLET (2023)

Instead, use a library:

not reachable[r, r, response, embeds]

TYPED FROGLET (2023)

Challenge: What's the type for reachable?

```
instead, use a library.

not reachable[r, r, response, embeds]
```

TYPED FROGLET (2023)

Challenge: What's the type for reachable?

Challenge: Are simple types enough?

TYPED FROGLET (2023)

Challenge: What's the type for reachable?

Challenge: Are simple types enough?

Challenge: Will types help us migrate 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"

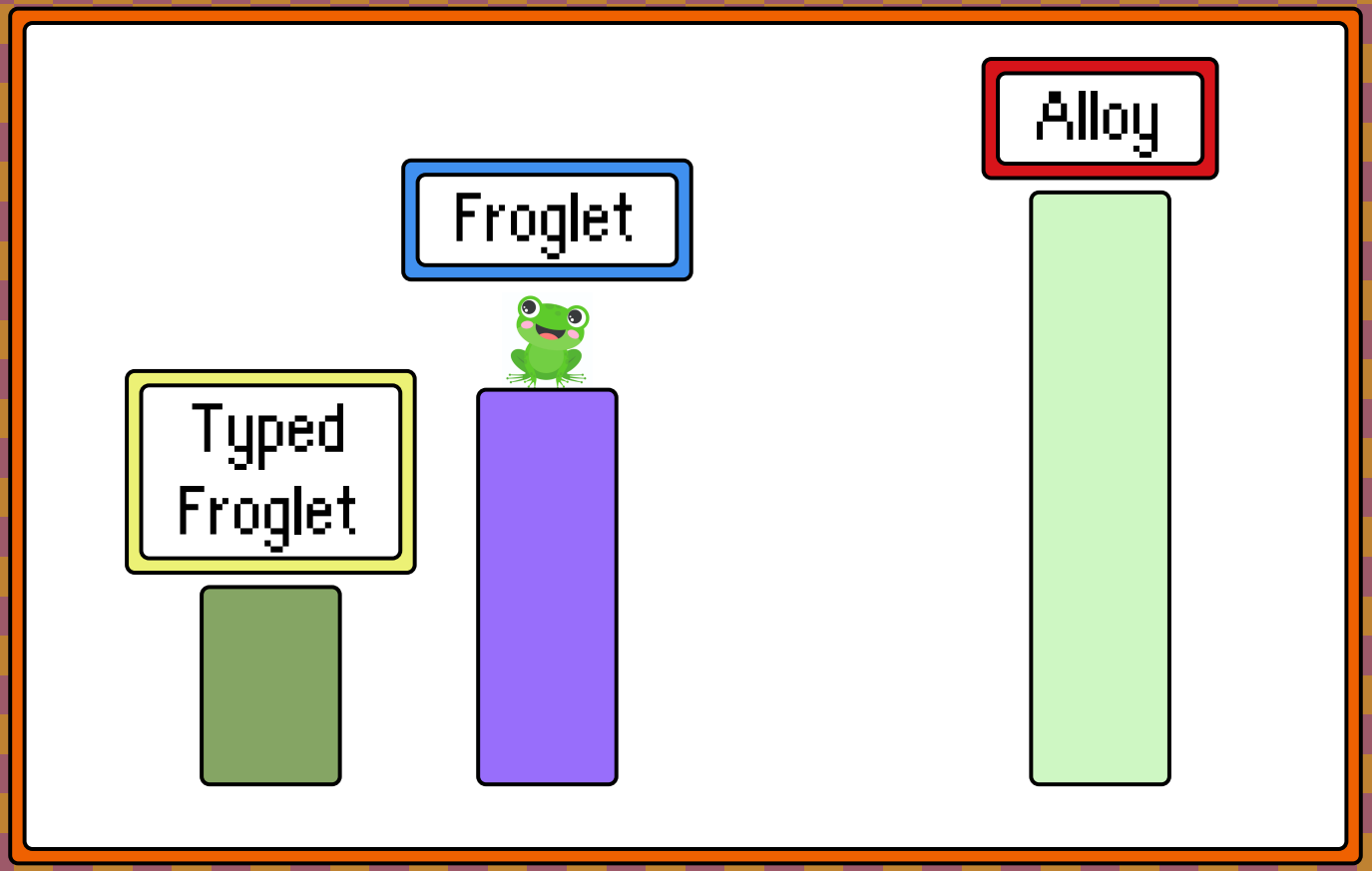
"This one just seems like it is wrong"

"Good riddance! What a travesty that would be"

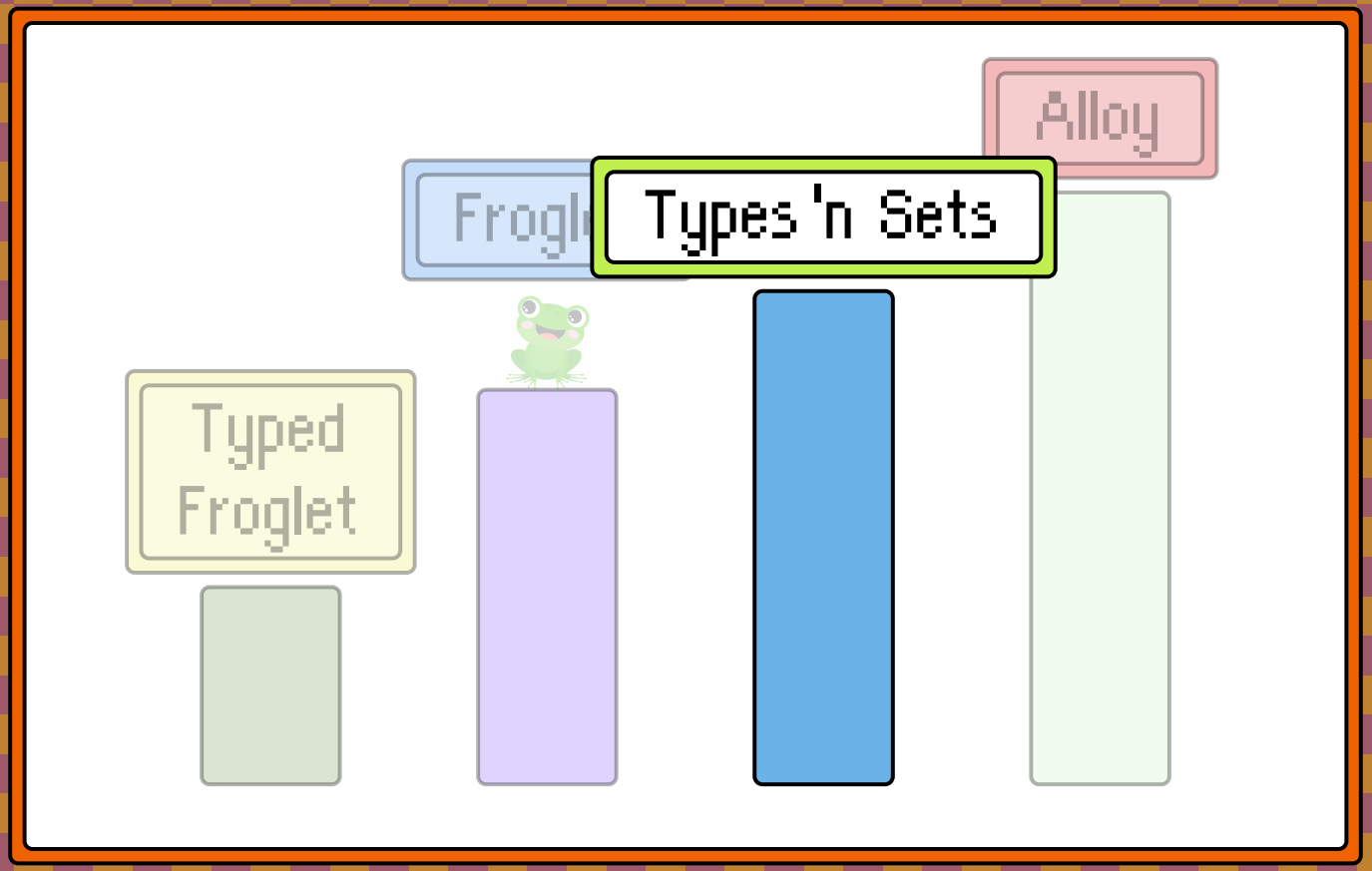
"No I just... no.... what's even happening...."

FROGLET TO ALLOY

FROGLET TO ALLOY



FROGLET TO ALLOY




THE END

TYPES FOR FROGLET

Goal: Java-style types to help programmers learn Alloy

Some Challenges:

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