Forge:
Usable Model-Finding

IETF 120

Ben Greenman
Siddhartha Prasad
Tim Nelson
Shriram Krishnamurthi

BROWN

THE UNIVERSITY OF UTAH
Datatypes

Relations
Datatypes

Relations

Exploration + Bounded Verification
Example:
Cross-Site Request Forgery
User's auth. cookies
Problem: every request carries User's auth. cookies

Idea: add origin to requests, validate at Good Server
abstract sig EndPoint {}

sig Client
  extends EndPoint {]
abstract sig EndPoint {}

sig Client
    extends EndPoint {}

sig Server
    extends EndPoint {
        causes: set HTTPEvent
    }
abstract sig EndPoint {}

sig Client
    extends EndPoint {}

sig Server
    extends EndPoint {
        causes: set HTTPEvent
    }

abstract sig HTTPEvent {
    from : one EndPoint,
    to : one EndPoint,
    origin : one EndPoint
}

// Request, Response, Redirect
// extends HTTPEvent
Datatypes => Exploration

```scala
#lang forge
abstract sig EndPoint {
    causes: set HTTPEvent
}

sig Server extends EndPoint {
    from: one EndPoint,
    to: one EndPoint,
    origin: one EndPoint
}

abstract sig HTTPEvent {
    from : one EndPoint,
    to : one EndPoint
}

sig Request extends HTTPEvent {
    response: lone Response
}

sig Response extends HTTPEvent {
    embeds: set Request
}

sig Redirect extends Response {}

run {} for exactly 2 Server, exactly 1 Client
```
Relations

Type 1: facts about the world

```plaintext
pred RequestResponse {
    all r: Response | one response.r
    // every Response is paired with
    // a unique request
}

// ...
```
Relations

Type 2: facts about our design

pred EnforceOrigins[good: Server] {
  all r:Request | r.to = good =>
    r.origin = good // from good server
  or
    r.origin = r.from // from client
}
Checks

```run{
    // can we find (hope not)
    some good, bad: Server {
        EnforceOrigins[good]
        // ...
    }
}
```

for exactly 2 Server,

exactly 1 Client,

5 HTTPEvent

bounds
run {
    // can we find (hope not)
    some good, bad: Server {
        EnforceOrigins[good]
        // ...
    }
} for exactly 2 Server,
    exactly 1 Client,
    5 HTTPEvent
Idea: add origin to requests, validate at Good Server

Redirects can be mis-labeled

How about a set of origins??
Quickly found a bug!
Proof Assistants

Lightweight FM

Model Checkers
Insight: Most bugs have small instances

small scope hypothesis - D. Jackson
What sets Forge apart?
What sets Forge apart?

Custom Visualization

Unit Testing

Language Levels
Custom Visualization
Custom Visualization

Much more than pretty pictures!

Building on decades of CogSci research

Applying Cognitive Principles to Model-Finding Output: The Positive Value of Negative Information

TRISTAN DYER, TIM NELSON, KATHI FISLER, and SHRIRAM KRISHNAMURTHI, Brown University, USA

Model-finders, such as SAT/SMT-solvers and Alloy, are used widely both directly and embedded in domain-specific tools. They support both conventional verification and, unlike other verification tools, property-free exploration. To do this effectively, they must produce output that helps users with these tasks. Unfortunately, the output of model-finders has seen relatively little rigorous human factors study.

Conventionally, these tools tend to show new satisfaction features at a time. However, this changes from the...
Unit Testing

example  assert  test suite  test expect

How do we know the model is correct?
Unit Testing

Challenge: Programming ≠ Modeling
Language Levels

r not in r.^((response.embeds)
Language Levels

r not in r.^(response.embeds)

CS1 in prereqs.CS2

"What a travesty that would be!"
Language Levels

#lang forge/temporal
++ Linear Temporal Logic

#lang forge/relational
++ N-ary Relations

#lang forge/bsl
Functional Relations
Language Levels

#lang forge/temporal
++ Linear Temporal Logic

#lang forge/relational
++ N-ary Relations

#lang forge/bsl
Functional Relations
In what ways is LTL difficult to use?

+3 years of studies with researchers and students
Categories of LTL Errors

- Bad Prop
- Bad State Index
- Bad State Quantification
- Cycle G
- Exclusive U
- Implicit F
- Implicit G
- Implicit Prefix
- Other Implicit
- Trace Split U
- Spreading X
- Weak U
Q. Translate to LTL:
The green light turns on exactly once

\( F = \text{eventually} \quad G = \text{always} \quad X = \text{next state} \)
Q. Translate to LTL:
The green light turns on exactly once

\[ F(\text{green}) \& G(\text{green} \Rightarrow X(\neg \text{green})) \]

\(F = \text{eventually}\)   \(G = \text{always}\)   \(X = \text{next state}\)
Q. Translate to LTL:

The green light turns on exactly once

\[ F = \text{eventually} \quad G = \text{always} \quad X = \text{next state} \]

\[ F(\text{green}) \land G(\text{green} \Rightarrow X(\neg\text{green})) \]

\[ F(\text{green}) \land G(\text{green} \Rightarrow X(G(\neg\text{green}))) \]
Q. Translate to LTL:
The green light turns on exactly once

F = eventually    G = always    X = next state

Wrong:  \( F(\text{green}) \land G(\text{green} \Rightarrow X(!\text{green})) \)

Correct:  \( F(\text{green}) \land G(\text{green} \Rightarrow X(G(!\text{green}))) \)
Q. Translate to LTL:

The green light turns on exactly once

\[ F \land G \neg \neg (\neg G) \land X (G \neg G) \]

F = eventually  
G = always  
X = next state
https://ltl-tutor.xyz
https://ltl-tutor.xyz
Datatypes → Relations → Exploration + Bounded Verification