

Applications of **Liquid Types** for more Reliable Software

Alcides Fonseca

<http://alcidesfonseca.com/>

LASIGE & Faculdade de Ciências da Universidade de Lisboa

Alcides Fonseca et al.

PhD Students



Catarina Gamboa — Improving the Usability of Liquid Types for Reliable Software (with Jonathan Aldrich)

- ▢ Usability-Oriented Design of Liquid Types for Java at ICSE 2023
- 🛠️ Artifact Evaluation Committee at PLDI 2023



Paulo Santos — Specification-Guided Repair for Robotic Systems (with Chris Timperley)

- ▢ An Experience Report on Challenges in Learning the Robot Operating System at RoSE 2022



Guilherme Espada — Stochastic Resource Analysis of Programs

- 🛠️ Artifact Evaluation Committee at PLDI 2022
- 🛠️ Video Chair at PLDI 2022, POPL 2023, PLDI 2023, SPLASH 2023



Pedro Barbosa — Modelling the impact of deep intronic mutations on the human disease (with Carmo-Fonseca)

- ▢ Computational prediction of human deep intronic variation (preprint)
- ▢ Clinical significance of genetic variation in hypertrophic cardiomyopathy: comparison of computational tools to prioritize missense variants at Frontiers in Cardiovascular Medicine

Interns



Leon Ingelse — GeneticEngine, an hybrid grammar-guided and strongly-typed genetic programming framework

- ▢ Comparing Individual Representations in Grammar-Guided Genetic Programming for Glucose Prediction in People with Diabetes at GEWS @ Gecco 2023
- ▢ Domain-Aware Feature Learning with Grammar-Guided Genetic Programming at EuroGP 2023

Machine Learning
Pipelines

Robotics

Implementations

Future Directions

Foundations of LiquidTypes

Foundations of Liquid Types

```
var x: Int = 3
```

var x:{Int|x>0} = 3

Hey! That's Ada's NATURAL!



```
var x:{Int|x>2&& x<18} = 3
```

Hey! That's Ada's Range Types!



var x:{Int|x%2=1} = 3



$\{x:T|p\}$ is a **refined** type.

- x is the name of variable (or fresh name)
- T is a type (usually a base type)
- p is a boolean predicate that can mention x .

$\{x:\text{Int} \mid x > 0\}$
 $\{y:\text{Bool} \mid y\}$
 $\{z:\text{Float} \mid \text{true}\}$

$\{x:T \mid p\}$ is a **refined and dependent** type.

- if p mentions variables other than x .

$$\{x:\text{Int} \mid x > y\} \\ \{p:\text{Int} \mid \text{true}\} \rightarrow \{q:\text{Int} \mid q > p\}$$

function div(x:int,y:intly $y \neq 0$)

...

div(3,0) $\times$

$\{z:\text{Int} \mid z == 0\} <: \{y:\text{int} \mid y \neq 0\}$

$\forall v:\text{Int}, v == 0 \rightarrow v \neq 0$

function div(x:int,y:intly $y \neq 0$)

...

div(3,4) ✓

$\{z:\text{Int} \mid z == 4\} <: \{y:\text{int} \mid y \neq 0\}$

$\forall v:\text{Int}, v == 4 \rightarrow v \neq 0$

```
function div(x:int,y:int|y!=0)
```

...

```
function g(z:int|z>=0) {  
  div(3,z) X  
}
```

$$\{z:\text{Int} \mid z \geq 0\} \leq \{y:\text{int} \mid y \neq 0\}$$
$$\forall v:\text{Int}, v \geq 0 \rightarrow v \neq 0$$

How to prove (or disprove)

$\forall v:\text{Int}, v \geq 0 \rightarrow v \neq 0$

?

Full Dependent Types: Manually

Liquid Types: Discharging to an SMT Solver

Full Dependent Types

Often found in Interactive Theorem Provers (ITP)

- They require writing the proof manually.
- They are very expressive.

```
theorem proof :  $\forall v, v > 0 \rightarrow v \neq 0$  := by
  intro v
  intro vgt0
  induction v
  simp at vgt0
  simp
```

LiquidTypes

- They discharge the implication to an SMT Solver

Z3 **CVC5**

```
alcides@Powerbook.local /Users/alcides python 01:46:21
Python 3.11.1 (main, Jan 9 2023, 14:15:10) [Clang 13.0.1 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import z3
>>> v = z3.Int("v")
>>> s = z3.Solver()
>>> s.add(v ≥ 0)
>>> s.add(z3.Not(v ≠ 0))
>>> s.check()
sat
>>> s.model()
[v = 0]
>>>
```

```
>>> s = z3.Solver()
>>> s.add(v > 0)
>>> s.add(z3.Not(v ≠ 0))
>>> s.check()
unsat
>>>
```

$\{x:T|p\}$ is a liquid, refined and dependent type.

- if p is drawn from a decidable logic (QF_UFLIA)

QF: Quantifier-Free

UF: Uninterpreted Functions

LIA: Linear Integer Arithmetic

$\{x:\text{Tip}\}$ is a **liquid, refined and dependent** type.

$\{z:\text{Int} \mid z > 0\}$
 $\{z:\text{Int} \mid z + y > 0\}$
 $\{z:\text{Int} \mid z * y > 0\}$
 $\{z:\text{Int} \mid z * 3 > 0\}$

$\{x:\text{Tip}\}$ is a liquid, refined and dependent type.

Uninterpreted Function

predicate **size** : Array -> Int

empty : $\{z:\text{Array} \mid \text{size}(z)=0\}$

append : x:Array ->

el:Int ->

$\{y:\text{Array} \mid \text{size}(y)=\text{size}(x)+1\}$

Implementations of Liquid Types

ML(1991)

C (2012)

Haskell (2014)

Scala (2016)

Java (2018)

Javascript (2012)

Typescript (2016)

Rust (2022)



Liquid Haskell

Extension of Haskell with Liquid Types

- <https://ucsd-progsys.github.io/liquidhaskell/>
- The most mature Liquid Type implementation
 - Enforces totality and termination
 - Allows to prove theorems
 - Reflects Haskell to Logic
- <http://ucsd-progsys.github.io/liquidhaskell-tutorial/>

Liquid Haskell

```
{-@ vectorSum :: (Num a) => Vector a -> a @-}  
• vectorSum x = sum [ x ! i | i <- [ 0 .. n ] ]  
  where  
    n      = length x
```

Liquid Haskell

```
{-@ fib :: Nat -> Nat @-}  
fib 0 = 1      }  
• fib n = fib (n-1) + fib (n-2)
```

Liquid Haskell

```
{-@ type OrdList a = [a]  $\perp$  @-}
```

```
{-@ ups :: OrdList Int @-}
```

```
ups = [1, 2, 3, 40, 5]
```

Flux

Rust plugin for LiquidType-based static analysis

- <https://flux-rs.github.io/flux/>
- Combines ownership and liquid types.

```
85
86  #[flux::sig(fn () → i32[3])]
87  pub fn mk_three() → i32 {
88      let mut r: i32 = 0;
89      r
90  }
91
92
93
94
95
96
97
98
99
100
```

LiquidJava



Compiler for Java annotated with Liquid Types

- <https://catarinagamboa.github.io/liquidjava.html>

```
@Refinement("positive > 0")
int positive;
positive = 50; //Correct
positive = -1; //Error
```

```
@RefinementAlias("RGB(int x) {x >= 0 && x <= 255}")
class MyColor{
    @Refinement("RGB(r)") int r;
    @Refinement("RGB(_)" ) int g;
    @Refinement("RGB(_)" ) int b;
}
```

```
@Refinement("_ >= 0 && _ >= n")
public static int absolute(int n) {
    if(0 <= n)
        return n;
    else
        return -n;
}
```

What about **gnatprove**?



```
@RefinementAlias("Percentage(int x) { x >= 0 && x <= 100}")
public class Grade{

    @Refinement("Percentage(_)" )
    public static int addBonus(@Refinement("Percentage(_)" )
                               int grade,
                               @Refinement("Percentage(_)" && bonus < grade")
                               int bonus){

        if((bonus + grade) > 100)
            return 100;
        else
            return grade + bonus;
    }
}
```

LiquidJava



Liquid Types and Typestate

Typestate

Overview Package **Class** Use Tree Deprecated Index Help

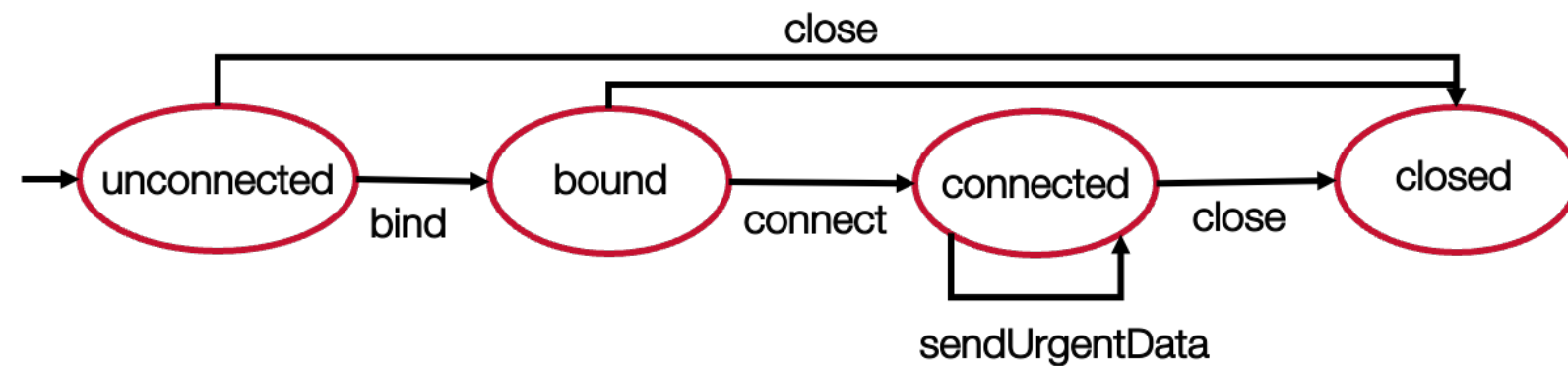
Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.net

Class Socket

java.lang.Object
java.net.Socket



```
@ExternalRefinementsFor ("java.net.Socket")
@StateSet ({ "unconnected", "bound", "connected", "closed" })
public interface SocketRefinements {

    @StateRefinement (to="unconnected(this)")
    public void Socket();

    @StateRefinement (from="unconnected(this)", to="bound(this)")
    public void bind (SocketAddress add);
```

Client code

```
Socket socket = new Socket();
socket.bind(new InetSocketAddress(...));
socket.sendUrgentData(90);
socket.close();
```

Applications of Liquid Types

Machine Learning

Data Source

Data Transformation

PCA

Feature Engineering

Temporal Aggregation

Label Encoding

Label Decoding

One Hot Encoding

One Hot Decoding

Vertical Concatenation

Horizontal Concatenation

Data Balancing

Visualization

Classification

Regression

Classifier Evaluation

Regressor Evaluation

Import from CSV 1

creditcard.csv

Rows: 284807

Columns: 31

Time Series: false

Undersampling

Random State: None

Split Dataset

Test Size: 0.25

Train Size: 0.75

Shuffle: false

Stratify by Class: false

Random Forest Classifier

Trees: 100

Max Depth: None

Criterion: gini

Evaluate Classifier

Import from CSV 1

(Dataset Port n_cols == 31)

(Dataset Port n_rows == 284807)

(Dataset Port n_rows == 284807)

(Dataset Port time_series == False)

(Dataset Port reduced == False)

(Dataset Port processed == False)

Problems

Assertions

Log

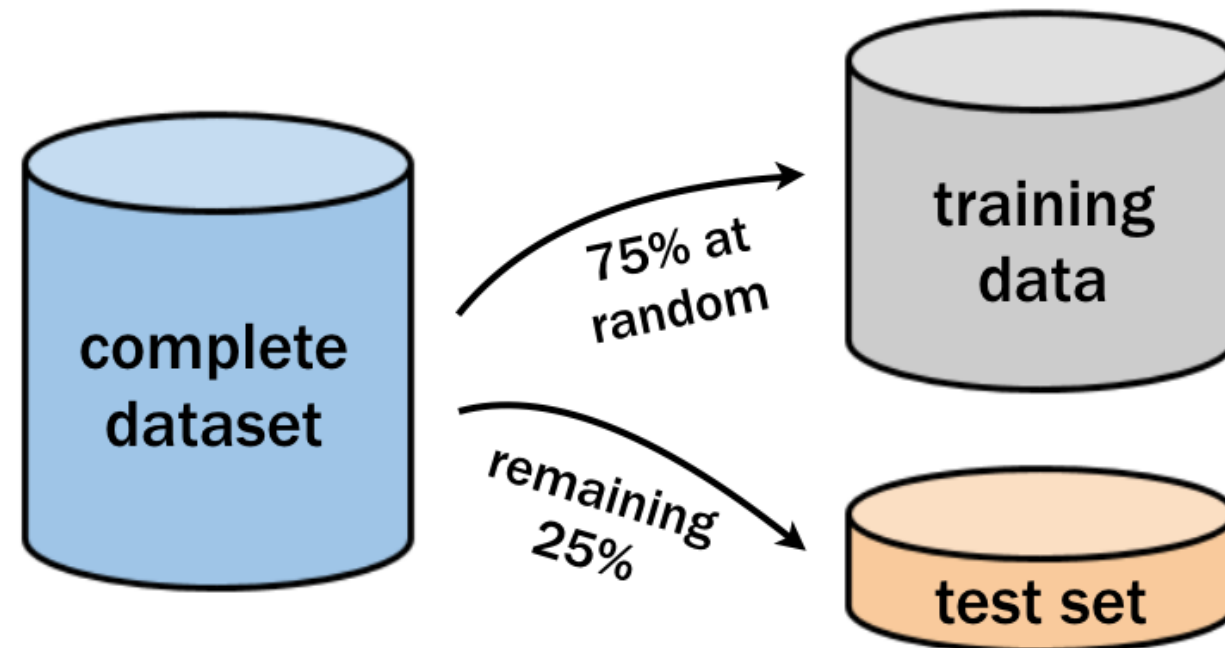
MLVP

Story #1: PHP Bytecode encoding

```
static const char *zend_vm_opcodes_names[199] = {  
    "ZEND_NOP",  
    "ZEND_ADD",  
    "ZEND_SUB",  
    "ZEND_MUL",  
    "ZEND_DIV",  
    "ZEND_MOD",  
  
    /* lots more */  
}
```

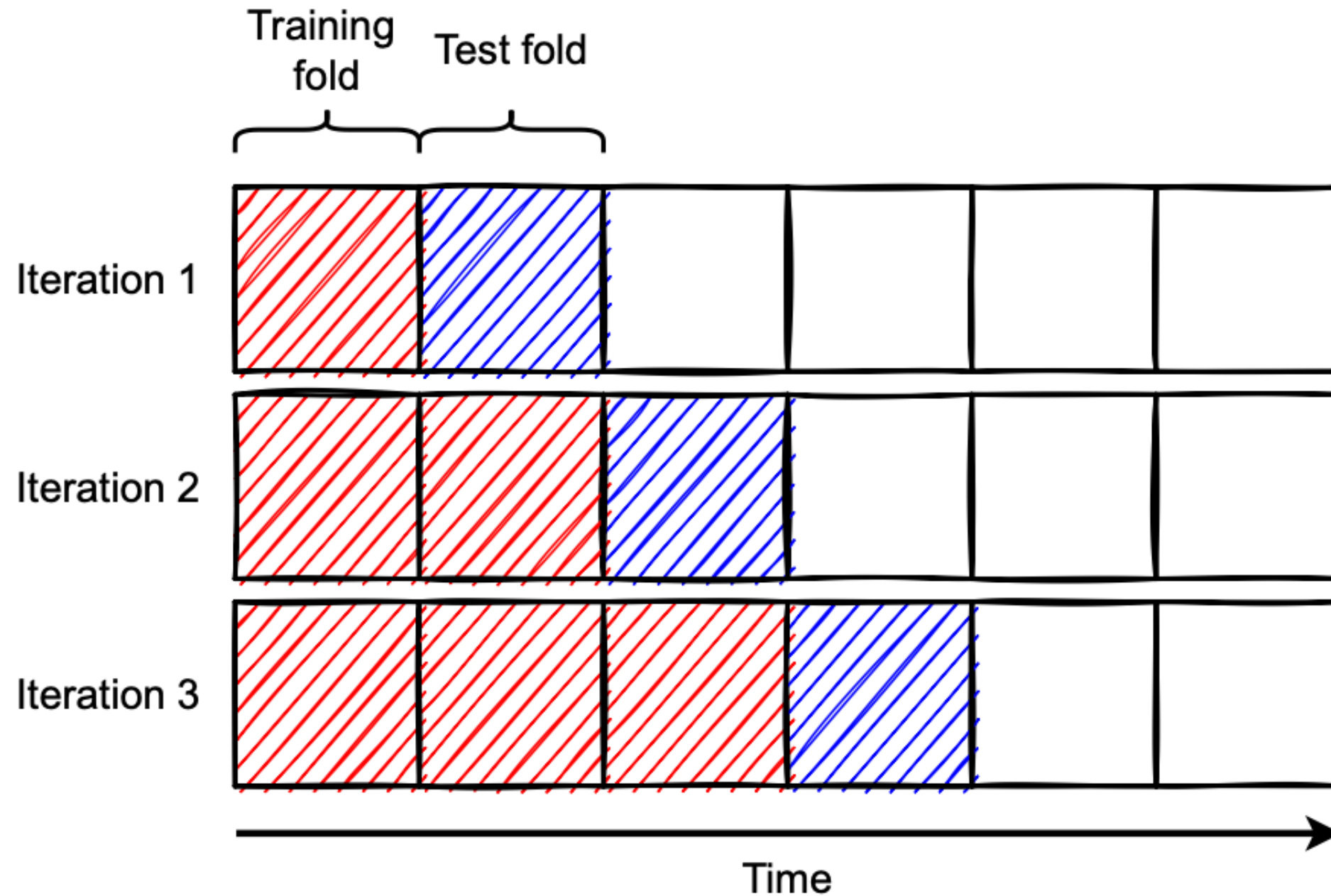
MLVP

Story #2: Sorted Dataset



MLVP

Story #3: Temporal Cross-Validation



MLVP

Story #4: Data Leakage

```
df['target'] = df['old_target'].map(  
    lambda x:1 if x == "Cat" else 0)  
X = df.drop('target')  
y = df['target']
```

⁵During our contamination check we discovered that portions of BIG-bench [48] were inadvertently mixed into the training set, and we excluded it from our reported results.

MLVP

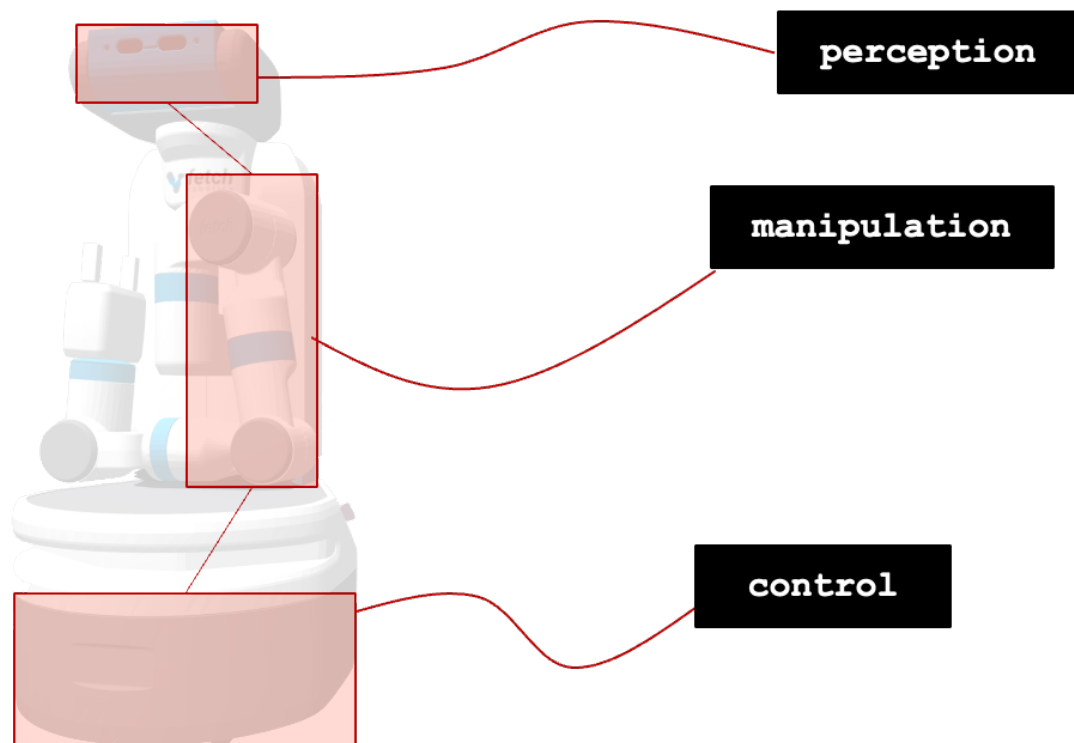
Static Verification of Semantic Properties

Categories	Defects
Feature normalization	Missing values
	Missing feature scaling
	Incorrect imputation
	Incorrect interpolation
Data balancing	Missing subsampling
	Missing supersampling
	Too many synthesized instances (e.g., SMOTE)
Training/testing instances	Use the same instances in training and testing
	Using features built from the target features in training
	Missing data shuffling
	Perform cross-validation using Time Series
Pipeline construction	Incompatible data type between components
	Unexpected data distribution
	Input dataset having an expected type or format

Applications of Liquid Types

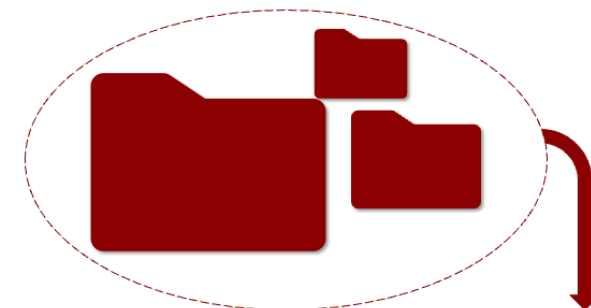
Robotics

Integrating Robotics Software Components is not easy!



High reuse of source code, libraries, and configuration files

[Garcia et al, 2020]



Developers rely on a small set of packages

[Estefo, 2019 et al; Kolak et al, 2020]

80% of dependencies

[Kolak et al, 2020]

Integrating Robotics Software Components is not easy!

Issues during development [Estefo et al, 2019; Afzal et al, 2020]

Difficulty in system configuration

Dependency Issues

Lack of documentation



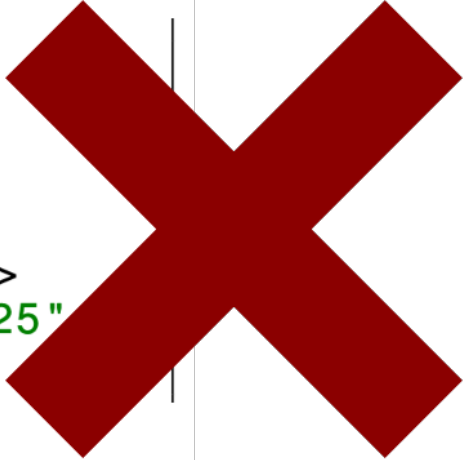
71%

reported issues when integrating new components

[Estefo et, 2019]

Integrating Robotics Software Components is not easy!

```
<!-- Launch joy node -->
<node pkg="joy" type="joy_node" name="joy">
  <param name="dev" value="$(arg_dev)" />
  <param name="deadzone" value="2" />
  <param name="autorepeat_rate" value="40" />
  <param name="coalesce_interval" value="0.025" />
</node>
```



~deadzone (double, default: 0.05)

Amount by which the joystick has to move before it is considered to be off-center. This parameter is specified relative to an axis normalized between -1 and 1. Thus, 0.1 means that the joystick has to move 10% of the way to the edge of an axis's range before that axis will output a non-zero value. Linux does its own deadzone processing, so in many cases this value can be set to zero.

ROSpec: a language for verification of robotics software architecture

Specification

```
expects {  
    cmdvel.linear.x > 0.0,  
    cmdvel.linear.y > 0.0,  
    cmdvel.linear.z > 0.0,  
} where {  
    cmdvel: Twist := topic /cmd_vel  
}
```

```
ensures {  
    topic_scan.range_max - topic_scan.range_min < ranges_size,  
    topic_scan.angle_min < topic_scan.angle_max,  
    topic_scan.range_min < topic_scan.range_max,  
    topic_scan.angle_increment >= 0,  
} where {  
    ranges_size: int := 120,  
    topic_scan: nav_msgs/LaserScan := topic /base_scan  
}
```

Native Functions

```
ensures {  
    tag(topic_scan, "metric"),  
    tag(t1, "metric"),  
} where {  
    t1: PoseWithCovariance := topic /initialpose  
}
```

LiquidJava for MAVSDK



- MAVSDK is one of the two most popular open-source drone controller frameworks (with a version in Java)



```
@ExternalRefinementsFor("io.mavsdk.mission.Mission")
@StateSet({...,"connectionEstablished", "missionUploaded", "missionStarted"})
public interface MissionControllerRefinements {
    @StateRefinement(from="connectionEstablished(this)",to="missionUploaded(this)")
    public void uploadMission(MissionPlan missionPlan);

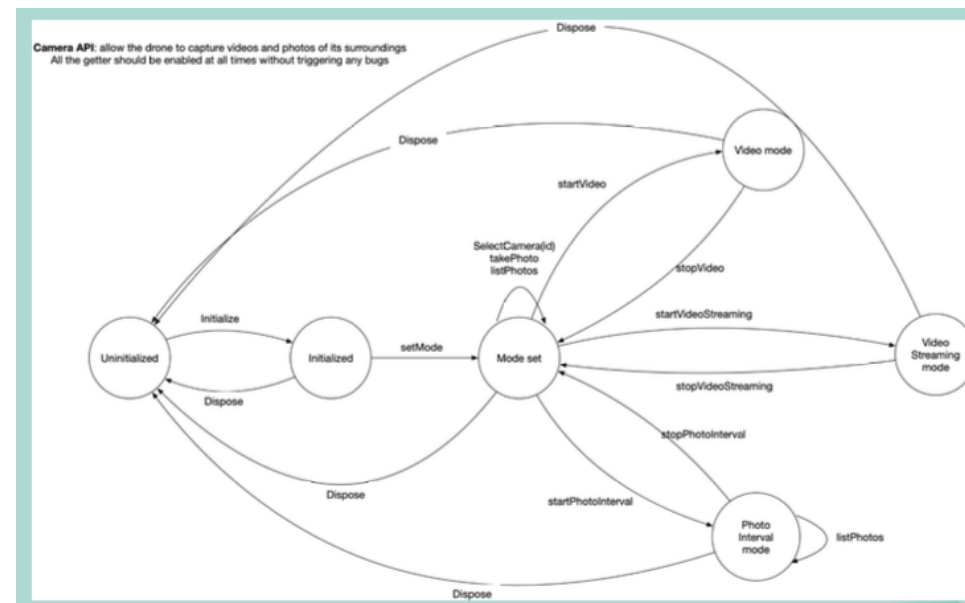
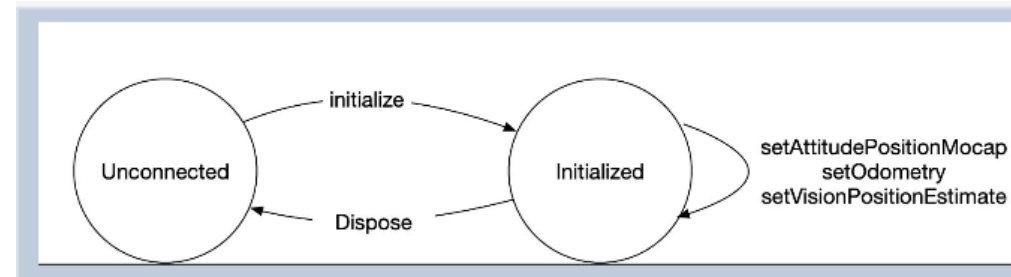
    @StateRefinement(from="missionUploaded(this)", to="missionStarted(this)")
    public void startMission();

    ...
}
```

```
Mission mission =
    drone.getMission();
//mission.uploadMission(...);
✗ mission.startMission();
```

LiquidJava for MAVSDK

	States	Transitions
Mocap	2	6
Geofence	3	5
Mission	4	14
Offboard	4	17
Camera	6	15
FTP	5	19
FollowMe	6	20



4 bugs were found!

2 in **Camera**, 1 in **FTP** and 1 in **FollowMe**

Why are
Liquid Types
not being adopted?

Participative Design of LiquidJava

Syntax Survey with 50 collected answers

A

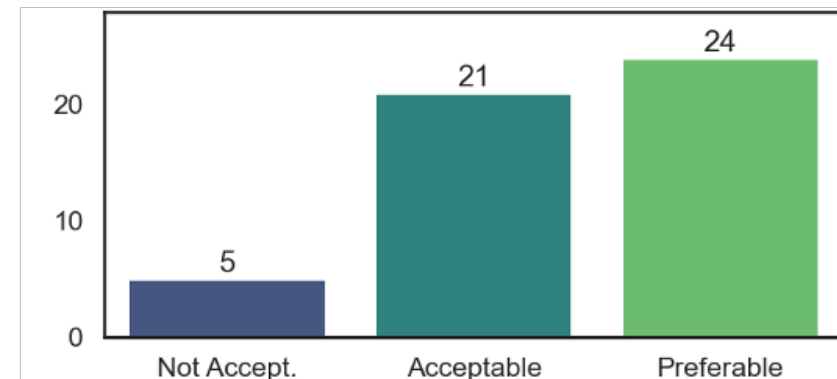
```
@Refinement("_ >= 0 && _ <= 100")
public static int percentageFromGrade(@Refinement("grade >= 0") int grade,
                                       @Refinement("scale > 0") int scale)
```

B

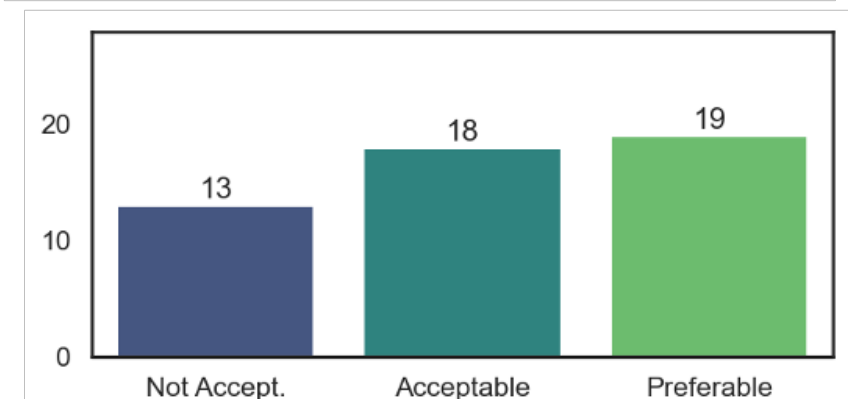
```
@Refinement("{grade >= 0} -> {scale > 0} -> {_ >= 0 && scale > 0}")
public static int percentageFromGrade(int grade, int scale)
```



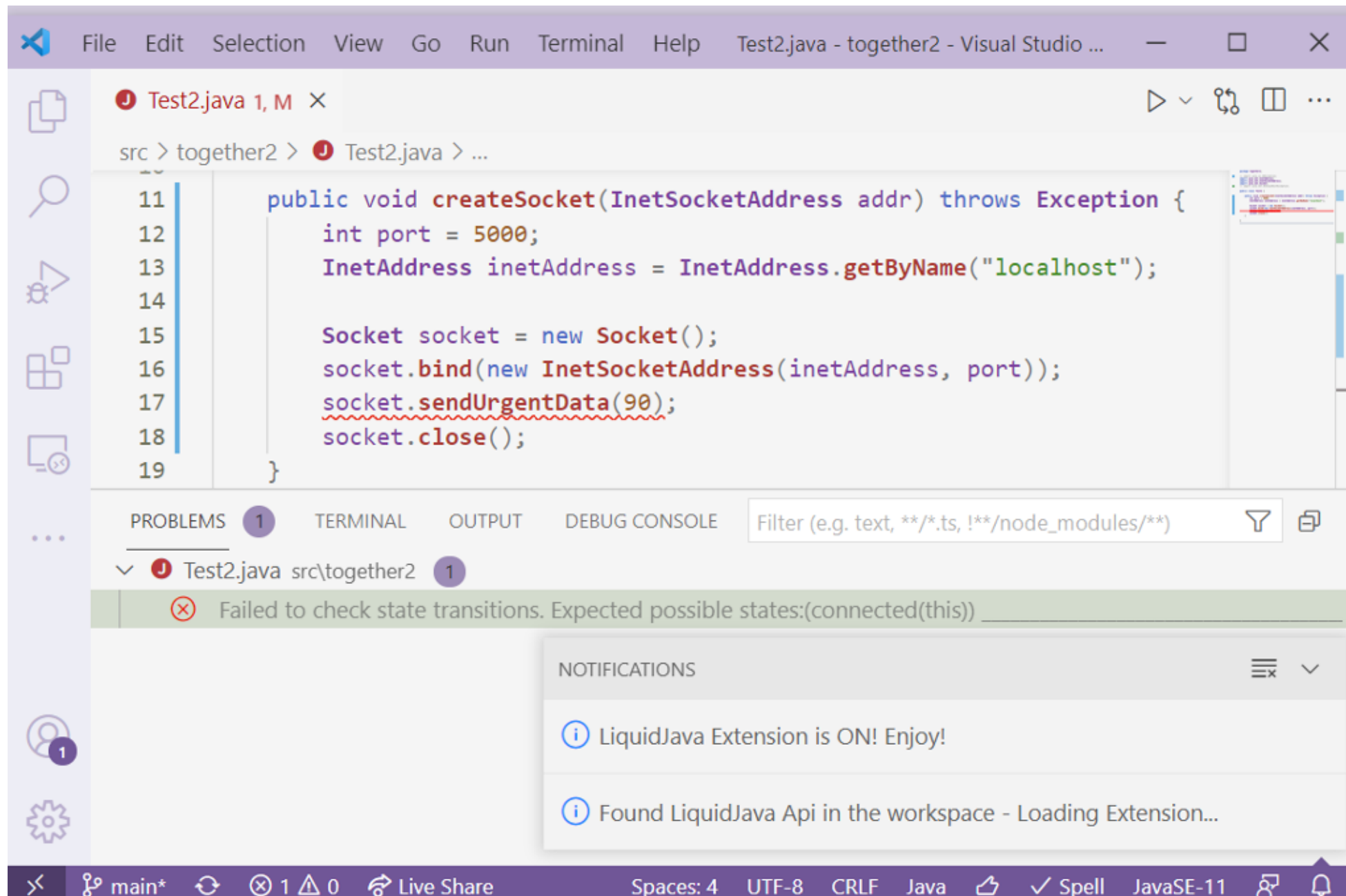
A



B



Editor Support for LiquidJava



User Study on LiquidTypes

- 1 Are refinements easy to **understand**?
- 2 Is it **easier and faster** to find implementation errors using **LiquidJava** than with plain Java?
- 3 How hard is it to **annotate** a program with refinements?
- 4 Are developers **open to using LiquidJava** in their projects?



30

56% *Very Familiar* with Java

80% *Vaguely or Not Familiar* with Refinement Types

User Study on LiquidTypes

Study configuration



Group 1

Task 1 - Find the error in plain Java RQ2

Sum
Socket

Exercises

Fibonacci
ArrayDeque



Task 2

RQ1

Understand Refinements without prior explanation



Overview of LiquidJava



Task 3 - Find the error in LiquidJava RQ2

Fibonacci
ArrayDeque

Exercises

Sum
Socket



Analysed Data

 Answers
(Correct, Incorrect,
Compiler Correct,
Unanswered)

 Time spent

 Long Answers

Task 4

RQ1+3

Annotate Java programs with LiquidJava

Comments

RQ4

Describe what they liked and disliked about LiquidJava

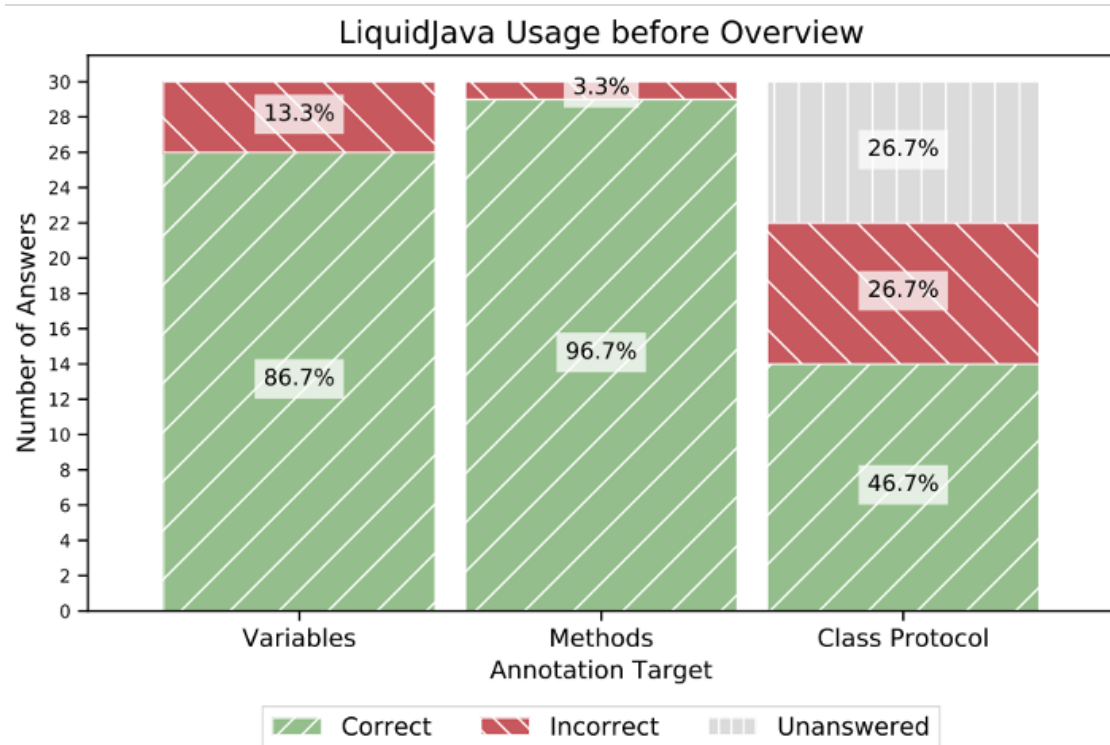


Group 2

User Study on LiquidTypes

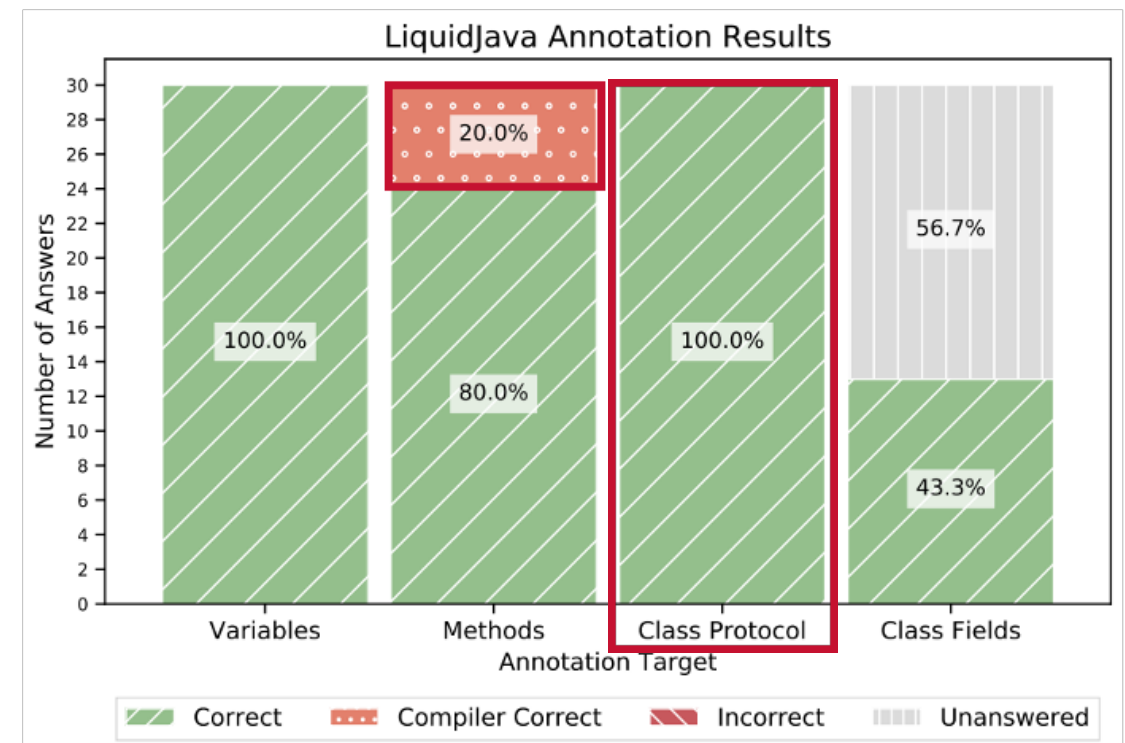
Intuitive refinements

```
@Refinement ("-25 <= x && x <= 45")
int x;
//Correct:
x = 0;
//Incorrect:
x = 46;
```



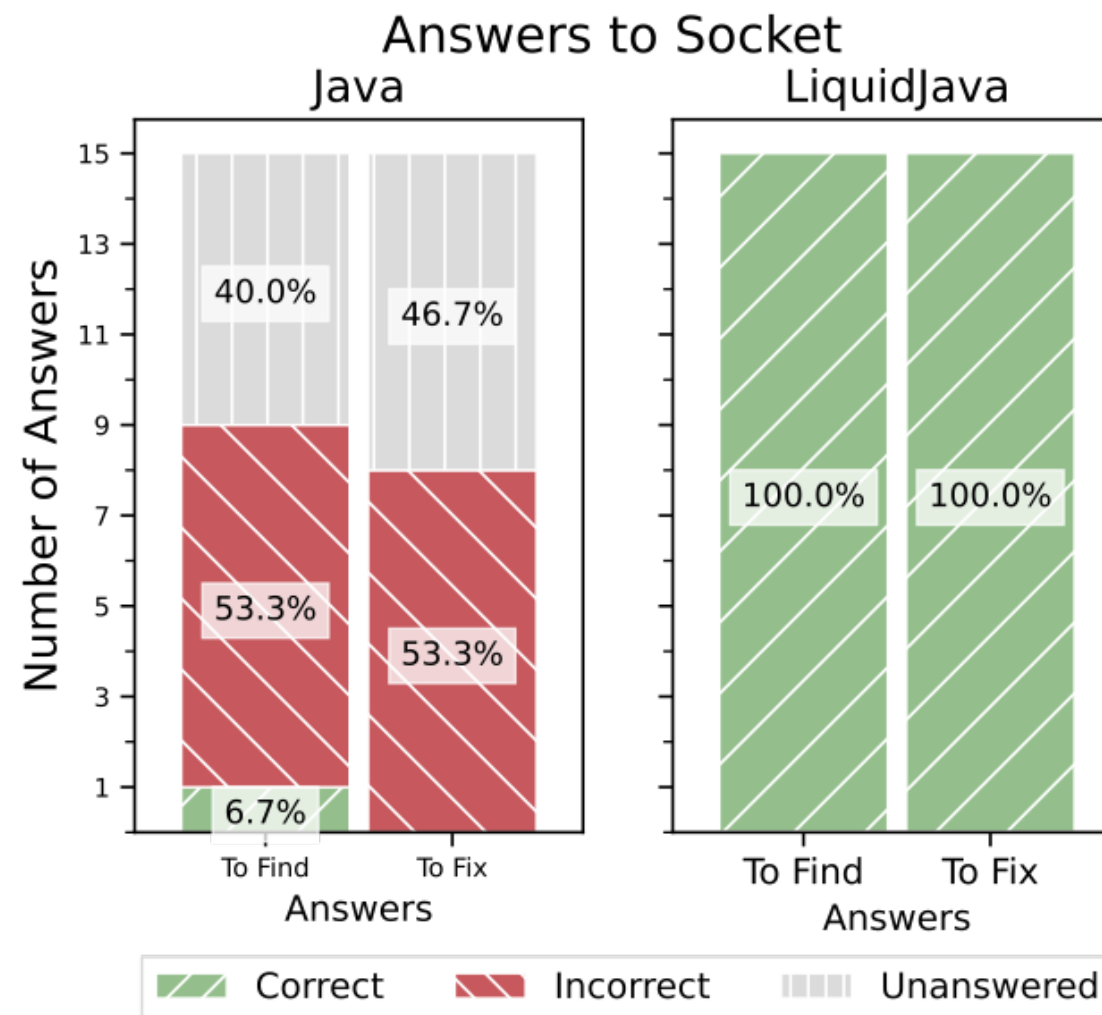
Overview Video and website

Add annotations



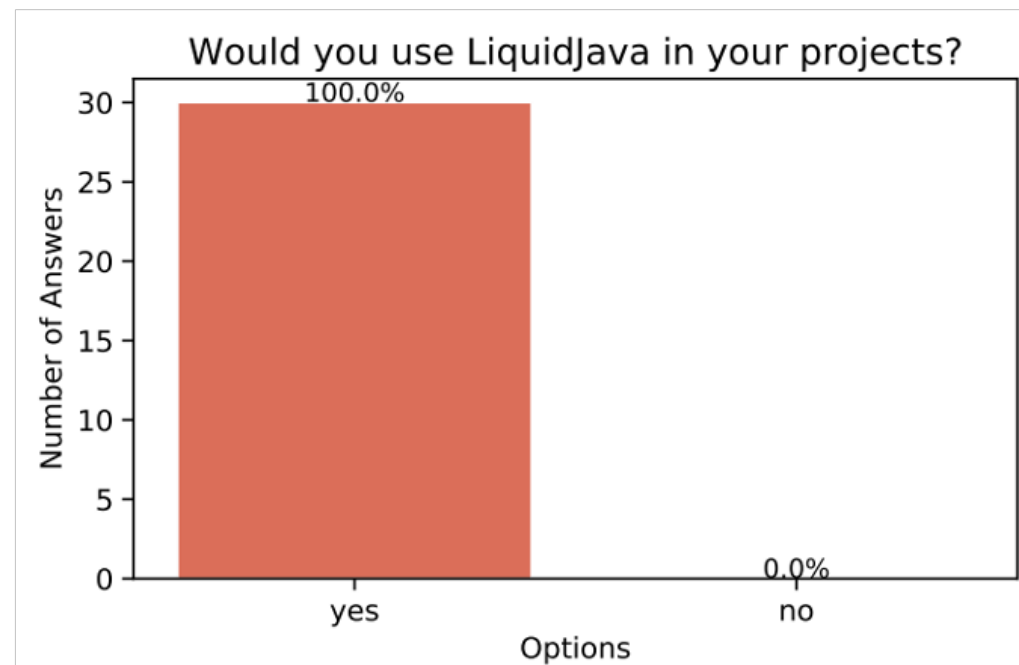
User Study on LiquidTypes

- ❑ LiquidJava **helped** developers find the bugs in code
- ❑ Best results for **lesser-known** classes



User Study on LiquidTypes

- ❑ Developers liked the **error reporting** and **state refinements** but want better **plugin integration** and **error messages**
- ❑ Participants are **interested** in using LiquidJava



Future Directions for Liquid Types

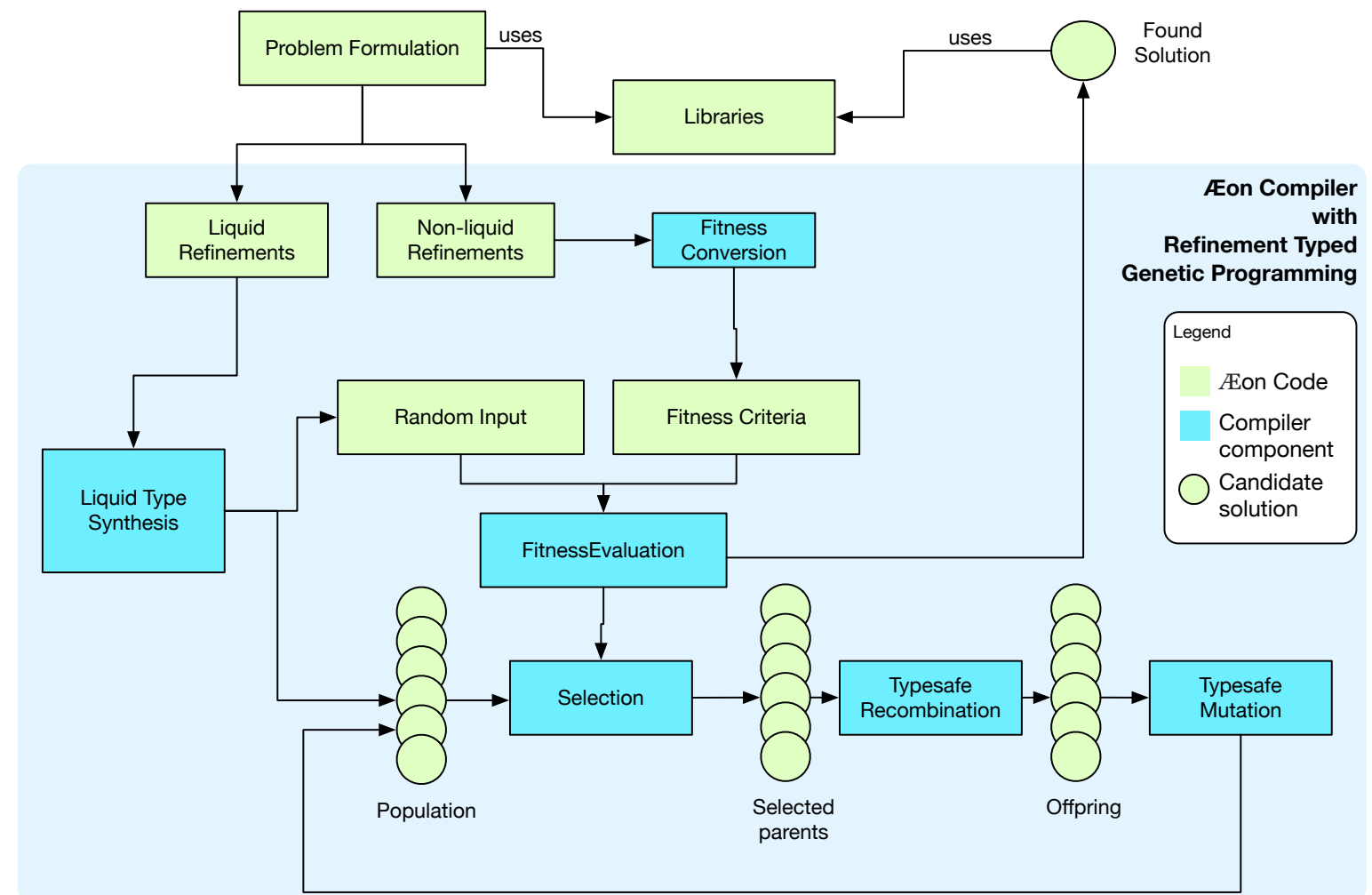
Program Synthesis

Program Synthesis with Refinements

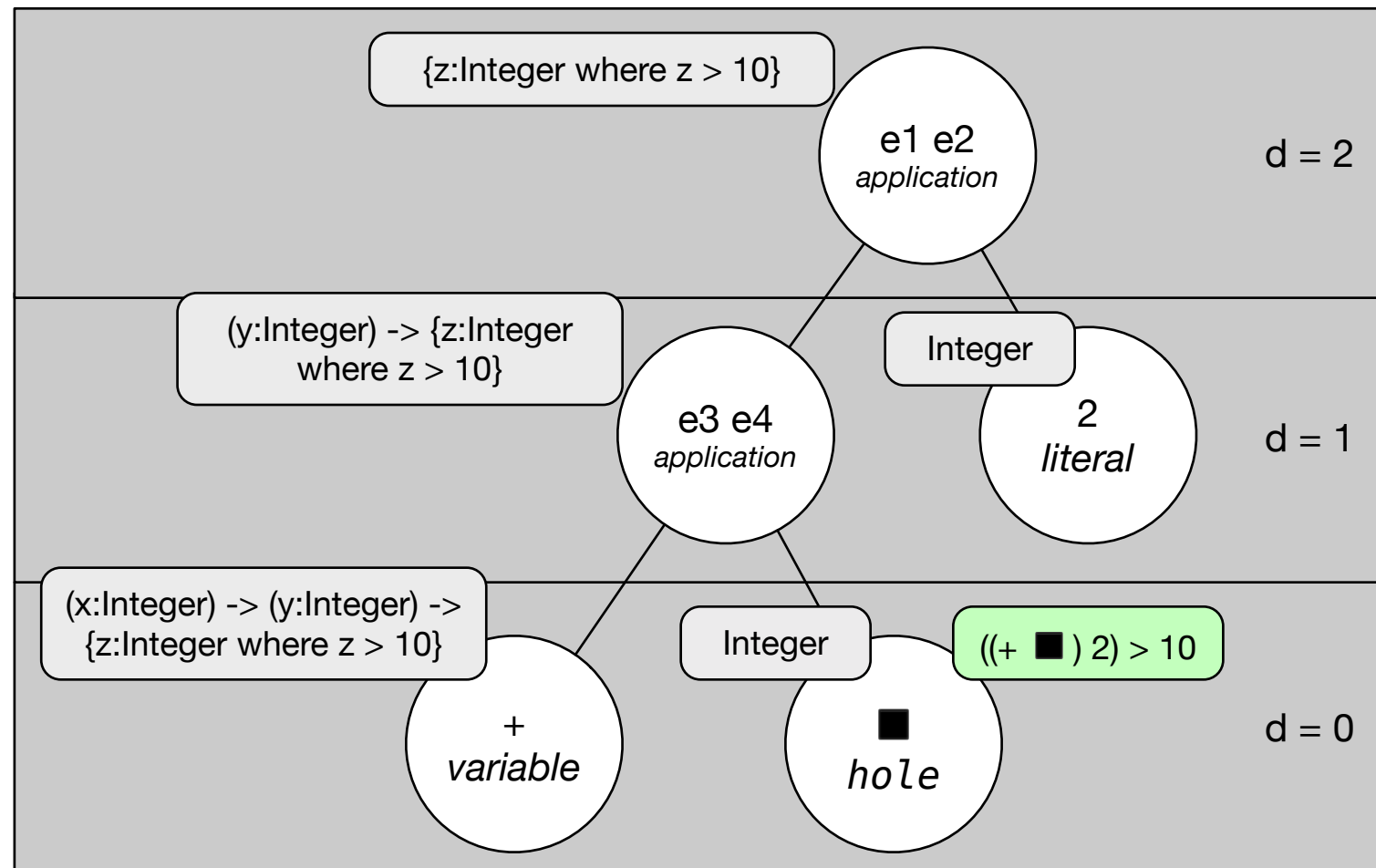
SMT-based Program Synthesis

Fitness Function via
Property-Based Testing

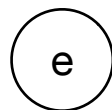
```
def f(i: Int, j: Int) -> k: Int | k > i+j and isPrime(k) {  
  ??  
}
```



Tree Generation



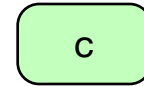
Legend



AST node



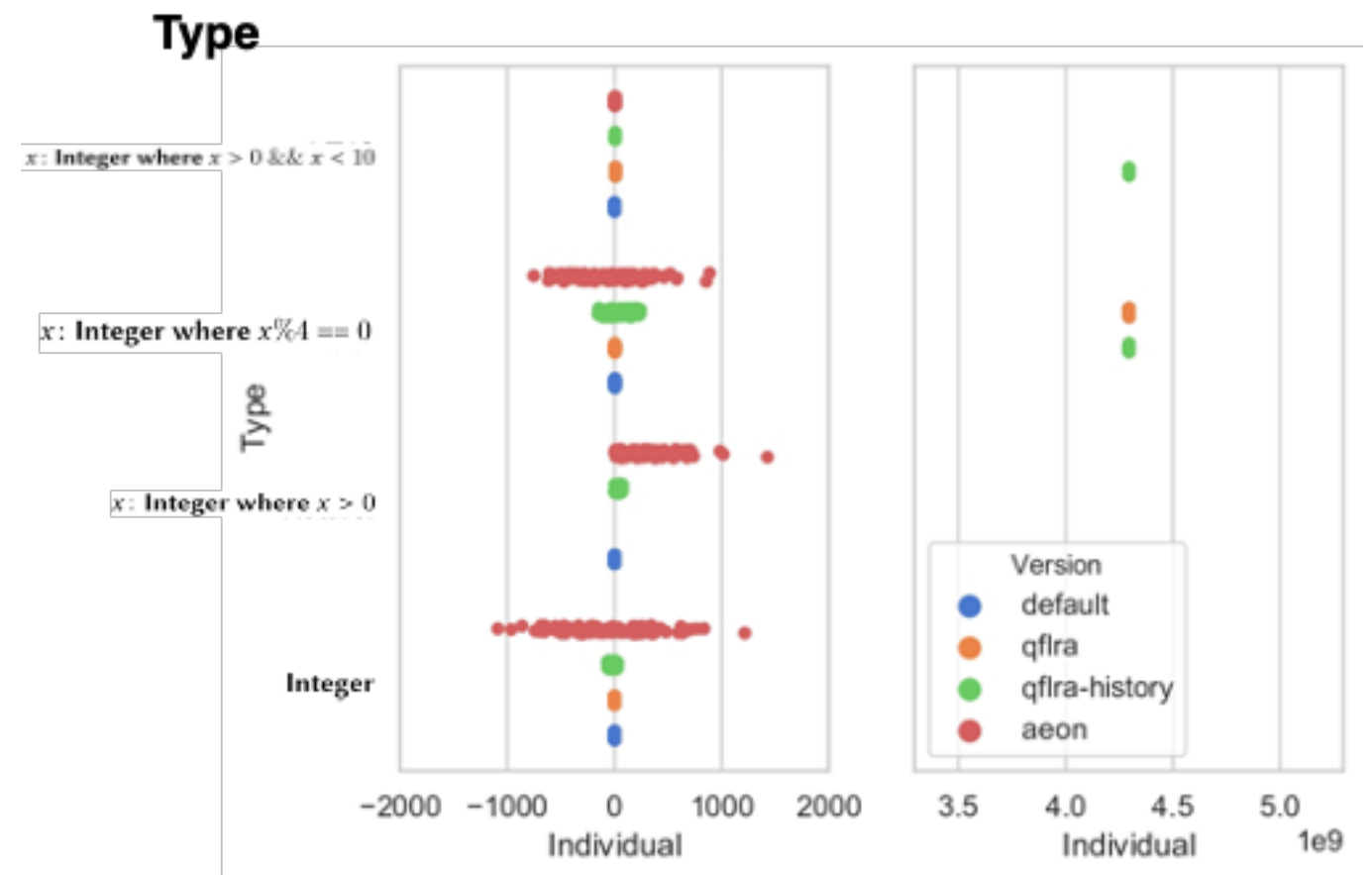
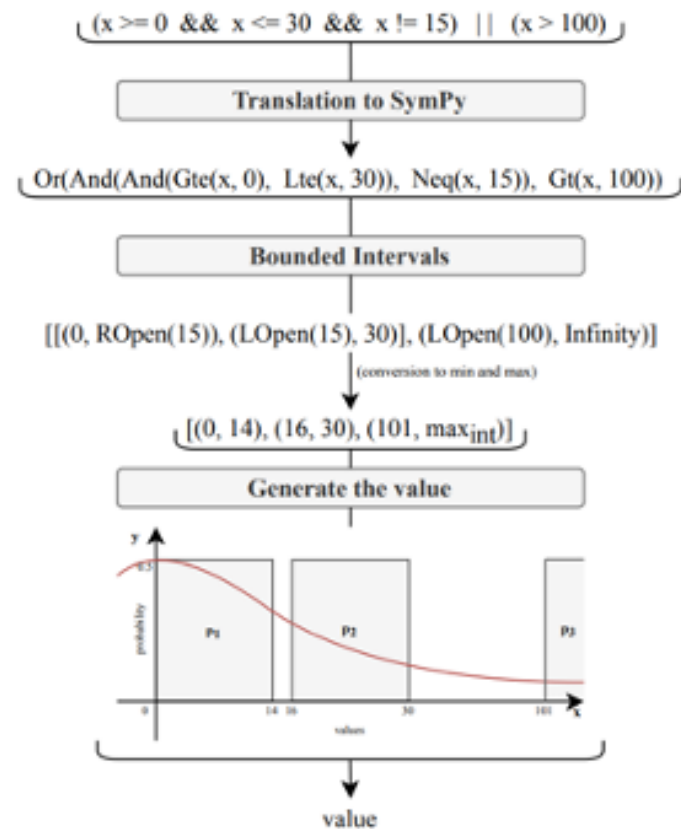
Expected Type



Expected condition

Problem: SMT Solvers lack consistent non-determinism

Solution 1: Sample from Intervals



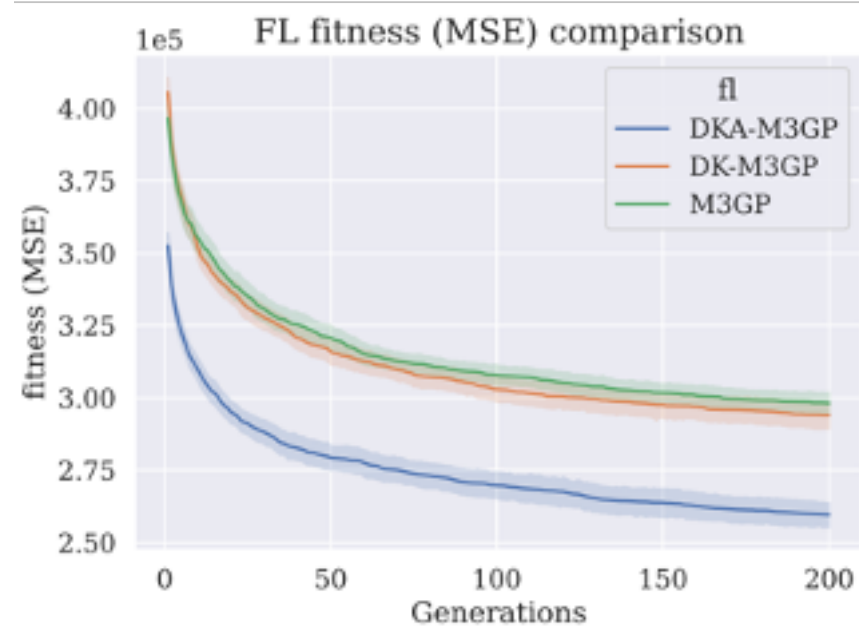
Solution 2: MetaHandlers

```
@weight(3)
@dataclass
class Container(Parent):
    age : Annotated[int, IntRange[0, 200]]
    height : Annotated[float, FloatRange[0.10, 2.5]]
    children : Annotated[List[Container], ListSizeBetween[1,30]]
    var_name : Annotated[str, VarRange["x", "y", "z"]]
```

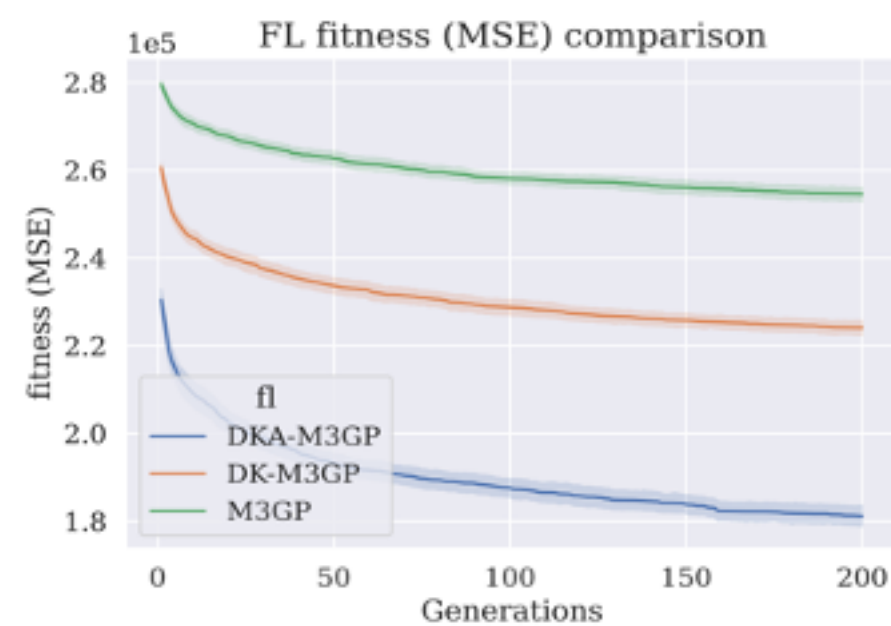
- GeneticEngine is a typed approach for Genetic Programming
 - <https://github.com/alcides/GeneticEngine>
 - pip install geneticengine
- GeneticEngine is being used for interpretable machine learning:
 - Glucose Prediction in Diabetes
 - Bike sharing usage
 - Understanding RNA Splicing

Genetic Programming improvement using Refined Types

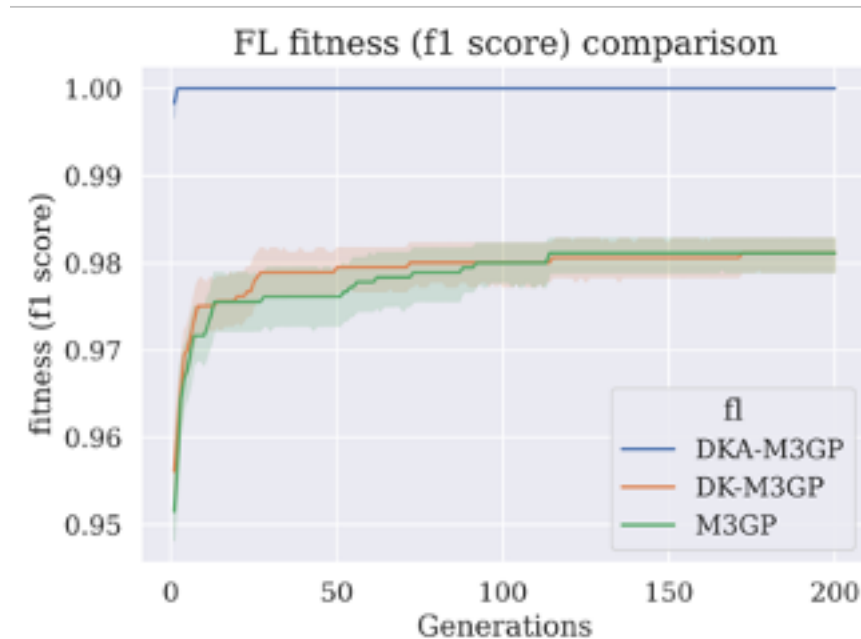
Boom bikes



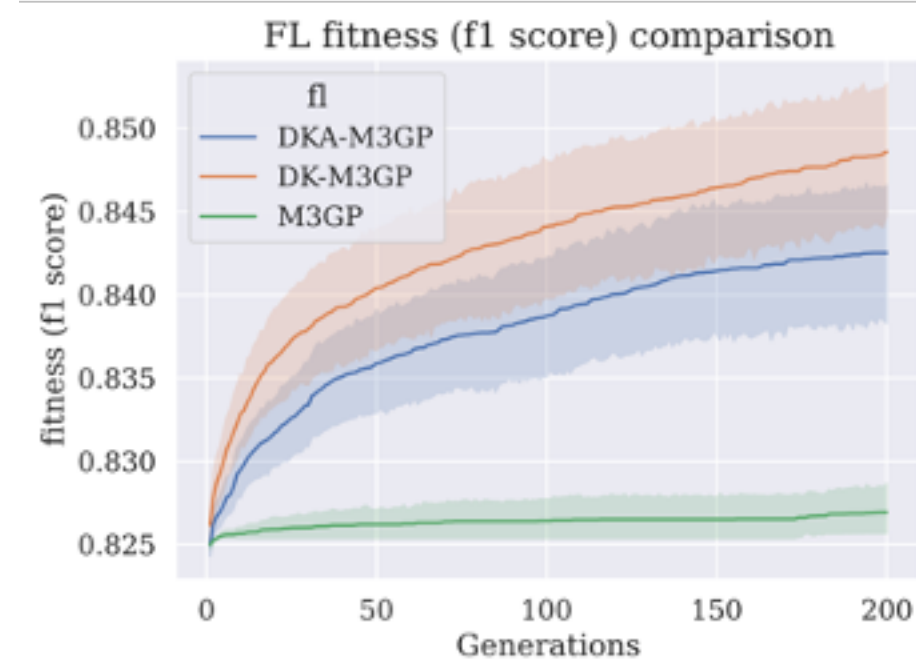
Website visitors



Caesarean



Credit Risk



Overview of Liquid Types

Liquid Types

- Types annotated with predicated from a decidable logic.
- Can be used to encode semantic restrictions that are validated during compilation
 - Useful for complex domains with lots of constraints (Machine Learning, Robotics)
 - Requires verification-style annotations, but more lightweight
- Can be used to guide search of programs (including ExplainableAI)
- Editor integration and error messages need to improve.

Questions?

