LASIGE reliable software systems

Extending Java with Refinements

Catarina Gamboa, Paulo Santos and Alcides Fonseca

U LISBOA Ciências ULisboa FCT Fundação para a Ciência e a Tecnologia

Refinement Types

- Division by Zero
- Array access

$$\{v : T \mid p(v)\}$$

```
@Refinement("y > 0 && y < 50")
int y;
y = 10; // okay
y = 100; // okay in Java, refinement type error
```

Introduction

Catarina Gamboa, Paulo Santos and Alcides Fonseca

WHY?

- First languages with refinements were not popular in industry
- Overhead for developers

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

LiquidJava

Refinement Types

Introduction

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

1. Refinements are optional.
2. Refinement type-checking works on top of existing Java type-checker
3. Expressive Refinements
4. Refinement type-checking should be decidable

RELATED WORK

Related Work

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

$$\{v : T \mid p\ (v)\ \}$$

Static verification with SMT solvers

Runtime checks

Related Work

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

Strongly typed functional languages

Mainstream languages

Streams – restrict notation

JML – Java Modeling Language

APPROACH

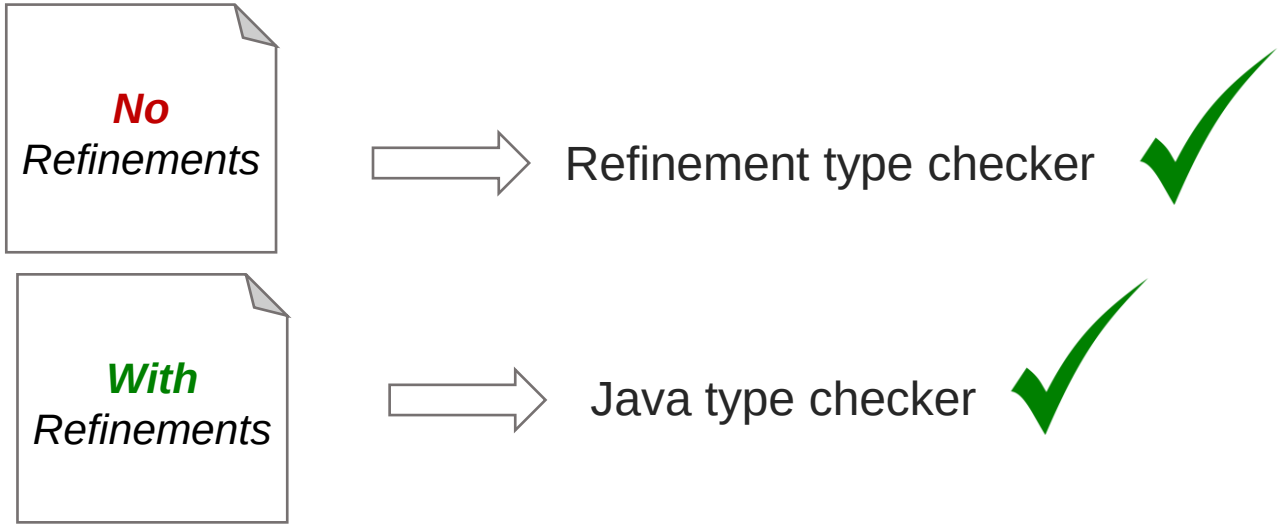
Approach

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

1

Refinements are optional



Annotations:

`@Refinement(…)`

Approach

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

2

Refinement type-checking on top of existing Java type-checker

Java type checker   Refinement type checker 

Only valid Java programs are accepted in LiquidJava

Approach

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

3

Expressive Refinements

```
@Refinement("y > 0 && y < 50")
```

4

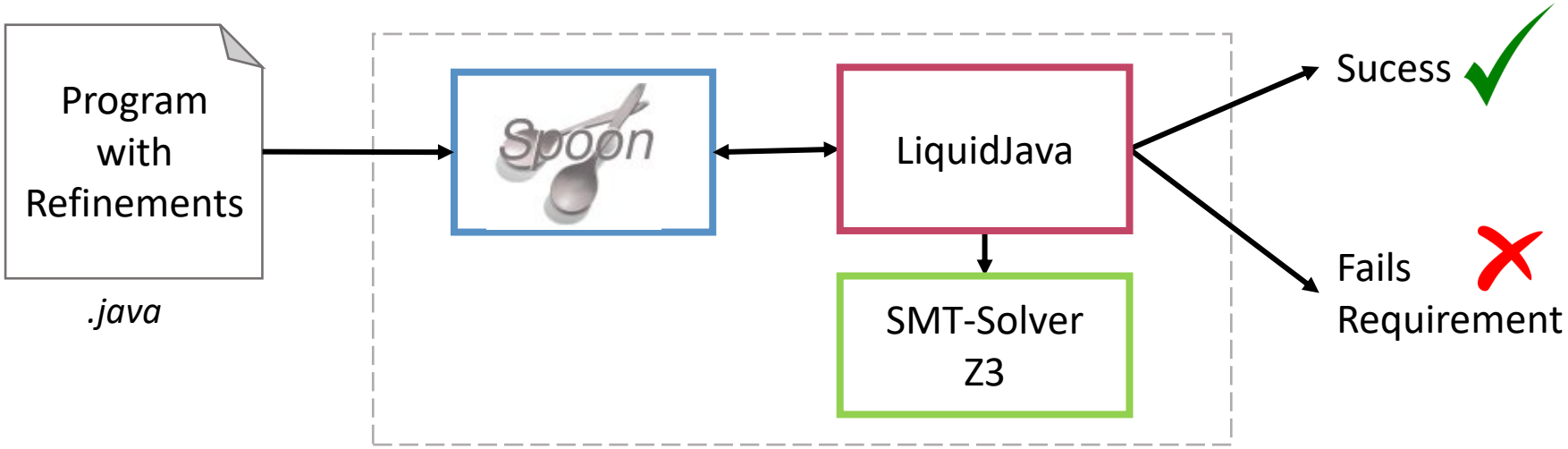
Refinement type-checking should be decidable

Limit language to Liquid Types

Approach

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems



Expressions are checked against their expected types ⇨ **Subtyping**

Approach

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

```
1  @Refinement("{x > 0} -> {\\v == 3 * x}")
2  private static int triplePositive(int x) {
3      return x + x + x;
4  }
5
6  public static void main(String[] args) {
7      @Refinement("k >= 10 && k < 100")
8      int k = 30;
9      @Refinement("t > 0")
10     int t = triplePositive(k);
11 }
```

Approach

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

```
1  @Refinement("{x > 0} -> {\\v == 3 * x}")
2  private static int triplePositive(int x) {
3      return x + x + x;
4  }
5
6  public static void main(String[] args) {
7      @Refinement("k >= 10 && k < 100")
8      int k = 30;
9      @Refinement("t > 0")
10     int t = triplePositive(k);
11 }
```

$$\{v{:}int \mid v == x + x + x\} <: \{v{:}int \mid v == 3 * x\}$$

$$\forall x : int.x > 0 \Rightarrow \forall v : int.v = x + x + x \Rightarrow v = 3 * x$$

Approach

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

```
1  @Refinement("{x > 0} -> {\\v == 3 * x}")
2  private static int triplePositive(int x) {
3      return x + x + x;
4  }
5
6  public static void main(String[] args) {
7      @Refinement("k >= 10 && k < 100")
8      int k = 30;
9      @Refinement("t > 0")
10     int t = triplePositive(k);
11 }
```

```
{k:int | k == 30} <: {k:int | k >= 10 && k < 100}
```

Approach

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

```
1  @Refinement("{x > 0} -> {\\v == 3 * x}")
2  private static int triplePositive(int x) {
3      return x + x + x;
4  }
5
6  public static void main(String[] args) {
7      @Refinement("k >= 10 && k < 100")
8      int k = 30;
9      @Refinement("t > 0")
10     int t = triplePositive(k);
11 }
```

```
{k:int | k >= 10 && k < 100} <: {k:int | k > 0}
```

Approach

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

```
1  @Refinement("{x > 0} -> { \\v == 3 * x}")
2  private static int triplePositive(int x) {
3      return x + x + x;
4  }
5
6  public static void main(String[] args) {
7      @Refinement("k >= 10 && k < 100")
8      int k = 30;
9      @Refinement("t > 0")
10     int t = triplePositive(k);
11 }
```

$$\forall k : int.k \geq 10 \land k < 100 \Rightarrow \forall t : int.t = 3 * k \Rightarrow t > 0$$

CHALLENGES

Challenges

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

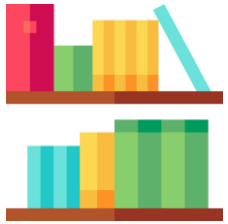
- Expressiveness
- Object state
  - Uninterpreted functions
- Dynamic behaviour of method invocations
  - Overrider subtype of Overridden

Challenges

Catarina Gamboa, Paulo Santos and Alcides Fonseca

LASIGE reliable software systems

- Annotate standard library
  - Fault Localization



- Evaluate patches in software repair context

Thank you!

Contact us:
cgamboa@lasige.di.fc.ul.pt