# **Evolutionary Program Synthesis** from Refined and Dependent types

## Paulo Santos, Andreia Mordido, Vasco Vasconcelos, Sara Silva, Alcides Fonseca

reliable software systems

LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

## **Motivation**

The software market revenue worldwide in 2018 was estimated in 456 billion U.S. Dollars and it is projected to grow to 507 until 2021. The focal activity of this industry is the maintenance and development of software.

#### Synthesis of low polygon Mona Lisa in ÆON

original : Image = loadImage("monalisa.jpg", 75, 150) -

evolvelmage() : {img : Image | width(original) == width(img) and





Program Synthesis is the task of generating a program that fulfils partially or completely a specification, allowing the development of more reliable, secure, faster and cheaper software.

## height(original) == height(img) and polygons(img) < 500 and @minimize (difference(original, img)) {

length of the output image.

#### **Predicate 3**

Limits the amount of polygons to ensure it is a low polygon image.

#### **Predicate 4**

Minimizes the difference between both images.

## Synthesize complete and partial programs from refined and dependent types

#### ÆON



+

**ÆON** is a functional programming language that uses refined types to generate valid expressions and dependent types to synthesize correct individuals.

cipher(x : Integer, key : Integer) : {y: Integer | y > 0 and ← **Refined Type** x == decypher(y, key)**Dependent Type** 

#### **Fitness function extractor**

Each assertion is recursively  $\mathbf{1}$ converted (f) into an objective 2 which each individual will try to 3 minimize.

Boolean		Continuous
true, false	$\rightarrow$	0, 1
x = y	$\rightarrow$	$ x-y _N$
$x \neq y$	$\rightarrow$	$1 -  x - y _N$
$a \lor b$	$\rightarrow$	min(f(a), f(b))
$a \wedge b$	$\rightarrow$	(f(a) + f(b))/2
$a \rightarrow b$	$\rightarrow$	$f(\neg a \lor b)$
$\neg a$	$\rightarrow$	1 - f(a)



Optimization

2

3

4



Solution found!

#### **ÆON to ÆONCORE**

ÆON was created with a userfriendly syntax, allowing new programmers to engage with the language. It acts like a syntactic frontend to its core, **ÆONCORE**, which exhibits a more complex syntax as it follows the type system rules.

# Integer → Integer → Integer {x : Integer | x != 0} Integer

**Non-deterministic Synthesis** 

### 1. Initialization

Providing the type of the hole operator to the nondeterministic synthesizer allows the generation of a random initial population.

## 2. Evaluation

The objective is to minimize the error between the user intention and the randomly tested fitness functions.

### 5. Mutation

Mutation randomly selects a node of a subtree and uses its type to synthesize a compatible replacement.

## 3. Selection

**Traditional Selection** Lexicase Selection Filter the individuals according to their performance on test cases.

#### fitness == 0 or generation == maximum generations

We propose a sound and complete synthesis algorithm which non-deterministically generates programs that use and define new polymorphic functions and makes recursive calls.

$$\Gamma \vdash \{x : Integer \mid x \neq 0\} \leadsto_d e$$

generate an expression with maximum depth *d*. -^//>*d* 





Acknowledgements: This work was supported by the Fundação para a Ciência e a Tecnologia (FCT) under LASIGE Research Unit UIDB/00408/2020 and the CONFIDENT project (PTDC/EEI-CTP/4503/2014).