Proving and Disproving Equivalence of Functional Programming Assignments

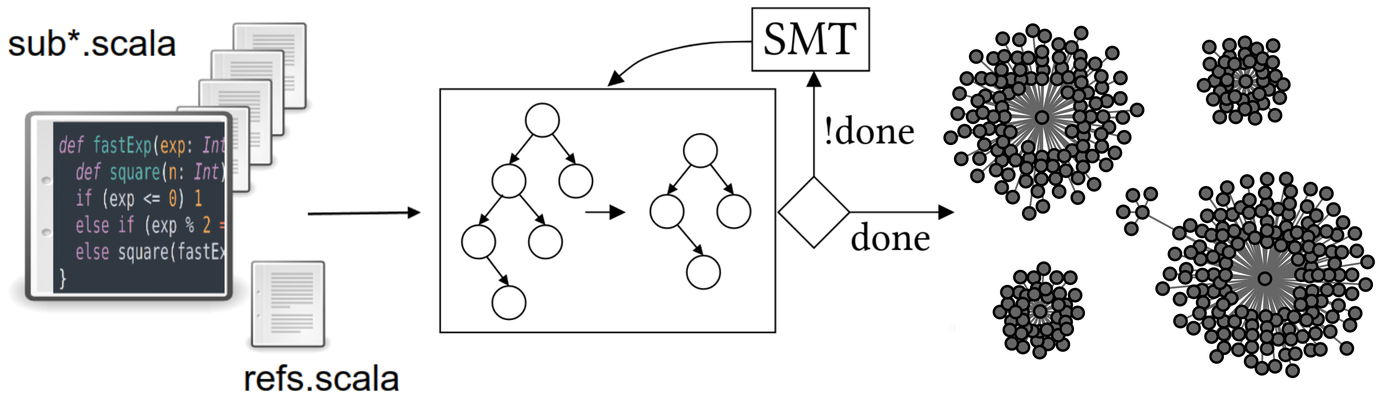
Dragana Milovančević, Viktor Kunčak

Is this solution correct?

```
def uniq(lst: List[Int]): List[Int] =
  distinct(List(), lst)

def distinct(a: List[Int], b: List[Int]): List[Int] =
  b match
    case Nil() => a
    case Cons(x, xs) =>
      if isin(x, a) then distinct(a, xs)
      else distinct(a ++ List(x), xs)

def isin(n: Int, lst: List[Int]): Boolean =
  lst.foldRight(false){ (e, acc) =>
    (e == n || acc)
  }
```

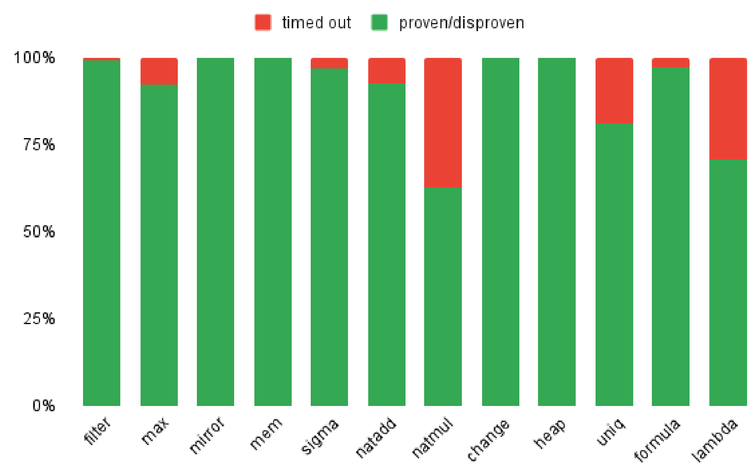
```
def uniqM(lst: List[Int]): List[Int] =
  distinctM(lst, Nil())

def distinctM(l: List[Int], r: List[Int]): List[Int] =
  l match
    case Nil() => r
    case Cons(x, xs) =>
      if !isinM(r, x) then distinctM(xs, r ++ List(x))
      else distinctM(xs, r)

def isinM(lst: List[Int], n: Int): Boolean =
  if lst.isEmpty then false
  else if lst.head == n then true
  else isinM(lst.tail, n)
```

Rigorous Automated Grading



Effective in Practice

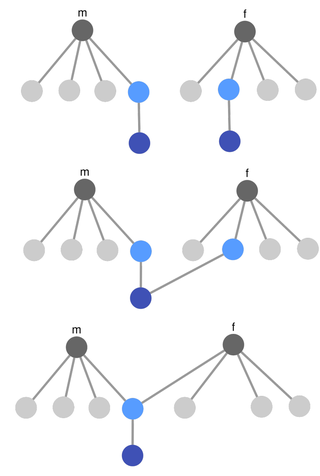
- Evaluation on over 4000 student submissions
- 86% overall success rate
- 96% success rate for single-function programs



Functional Induction

- Proofs by induction on the function's execution trace

Function Call Matching



Clustering Algorithm

- Scaling to hundreds of student submissions
- Discovery of tests and intermediate solutions

Going Further

- Stainless: a tool for verifying Scala programs: github.com/epfl-lara/stainless
- PLDI'23 paper: "Proving and Disproving Equivalence of FP Assignments"
- Interested in joint work or a project? Contact us!

Contact
dragana.milovancevic@epfl.ch
viktor.kuncak@epfl.ch

EPFL

Laboratory for Automated Reasoning and Analysis