

Formal verification for the masses

Testing vs Proving

Rules for transforming programs

```
if ¬true then A else B
= { applying ¬ }
  if false then A else B
= { semantics of the if statement }
  B
```

Smart contracts

- It's important to sleep at night if you're responsible for a smart contract holding valuable assets
- Formal verification can and has been applied successfully to make more secure smart contracts

Formal Land

• Ethereum, Tezos, Sui, Aleph Zero



Why it's not used more?

- Dependent types
- Niche platforms and languages
- PhD level knowledge

```
1 +1-length : ∀ {ℓ}{A : Set ℓ}(h : A)(t : L A) → length (h :: t) ≡ length t + 1
2 +1-length h [] = refl
3 +1-length h t = refl
```

A bridge

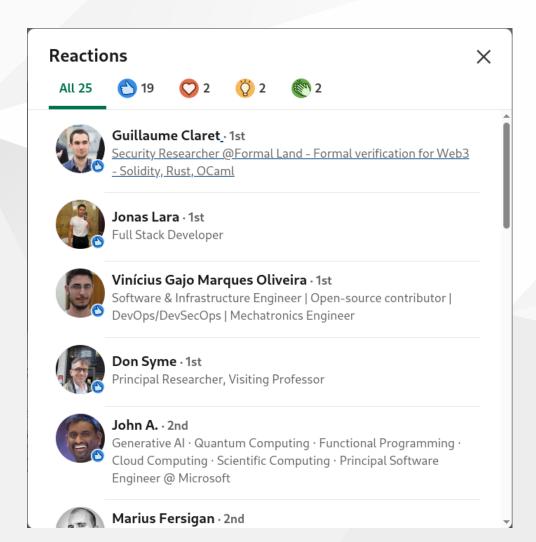
- F# and .NET
- Simpler yet powerful language
- A work in progress

```
1 let ''double negation'' =
2 proof () {
3    Theorem("double negation", !(!x) == x)
4    withLaws { ''true theorem'' }
5    !(!x) == x
6    ''='' { ''GS 3.11'' }
7    !x == !x
8    ''='' { ''= ident'' }
9    True
10 }
```

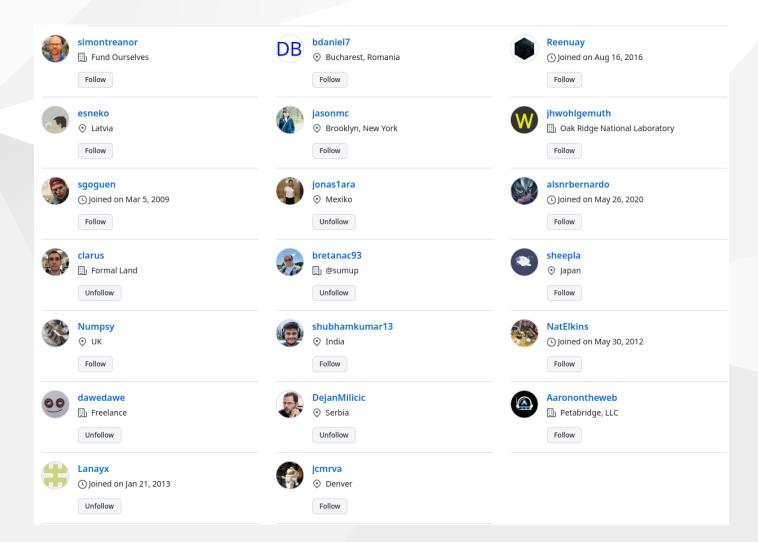
Reactions



Reactions



Reactions



10 years ago

"Llegué a este estilo por mi preocupación de escribir los programas bien, pues los errores se me escapaban al escribir un programa y notaba que estaban ahí cuando los ejecutaba", afirma.



Luis Ángel Méndez, un bicho raro entre sus colegas. Foto: Eduardo González Martínez

El método Dijkstra se opone a la práctica imperante de escribir un programa primero y someterlo después a una corrección matemática. Con su propuesta, aunque más demorada, se obtiene un programa listo para ejecutarse, correcto desde su construcción.

Questions?

• https://github.com/lamg