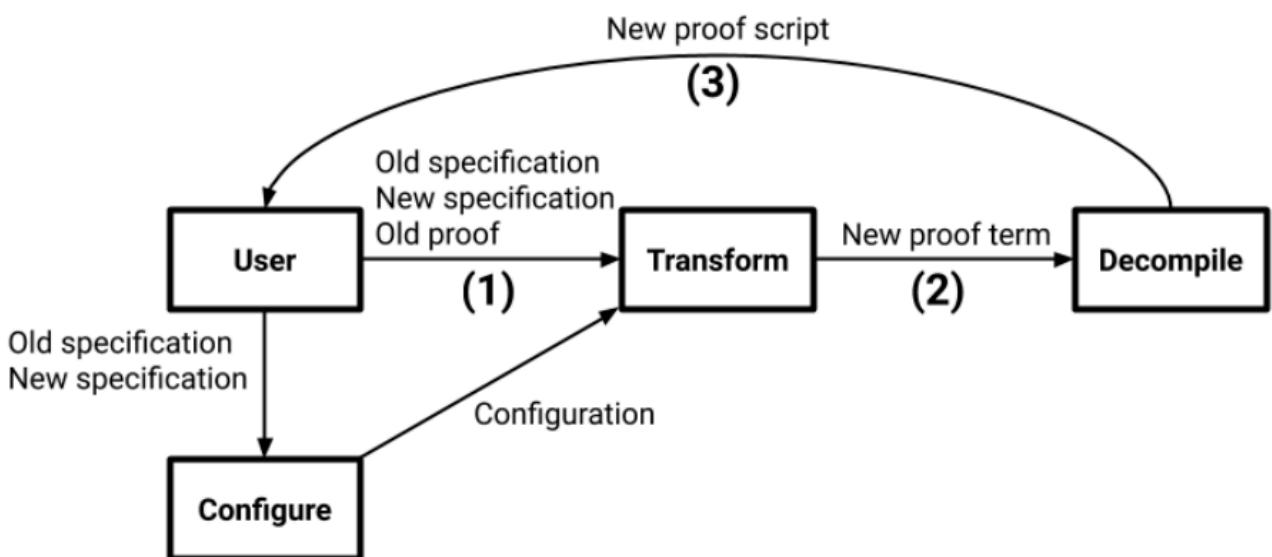


Proof Repair across Type Equivalences

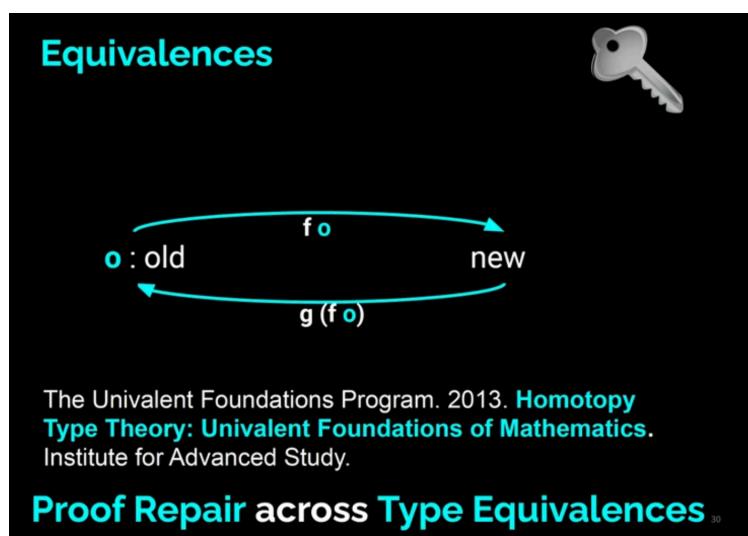
2021 PLDI - talk

摘要：

- 1 在 Coq 中自动修复 “由于变换data type而损坏的证明”。
- 2 将一个 可配置的证明term的转换 和一个 从证明term到tactic scripts的反编译器 相结合。
- 3 证明term的转换：支持type等价性，删除对旧版本type的引用，不依赖于 Coq 假设之外的公理。
- 4 Pumpkin Pi (它是 Pumpkin Patch Coq 插件套件的扩展，用于验证修复)

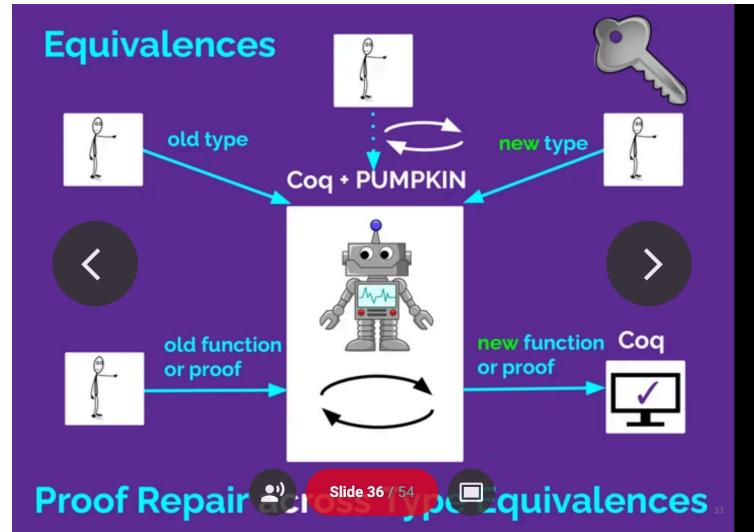


- 实现：通过在coq中加个插件实现 (pumpkin pi) <http://github.com/uwplse/pumpkin-pi>
- 支持type equivalence的任何更改，
用homotopy type theory描述就是，一组函数在两种type或一个type的两个版本之间的循环

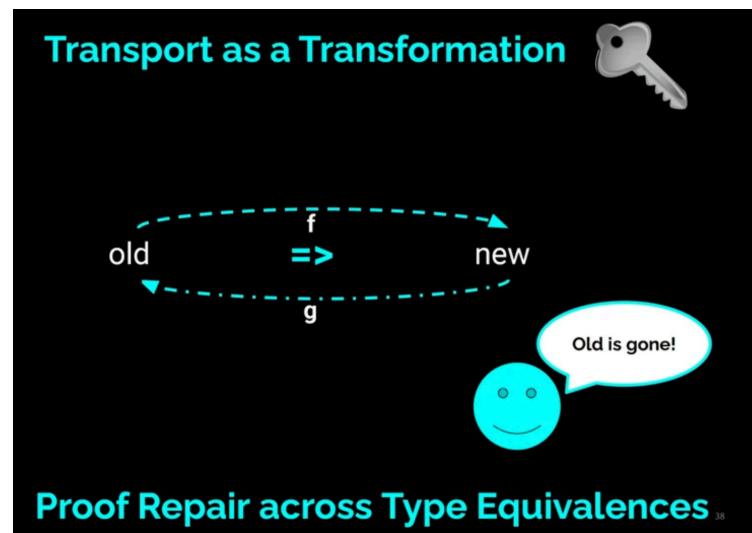


- 有search步骤，给新旧版本的改变过的data type，会寻找并且自动证明这种描述改变的等价性。

- 当没有search时，用户可以自己pass这个equivalence。
- 最后会用新的type修复函数和证明



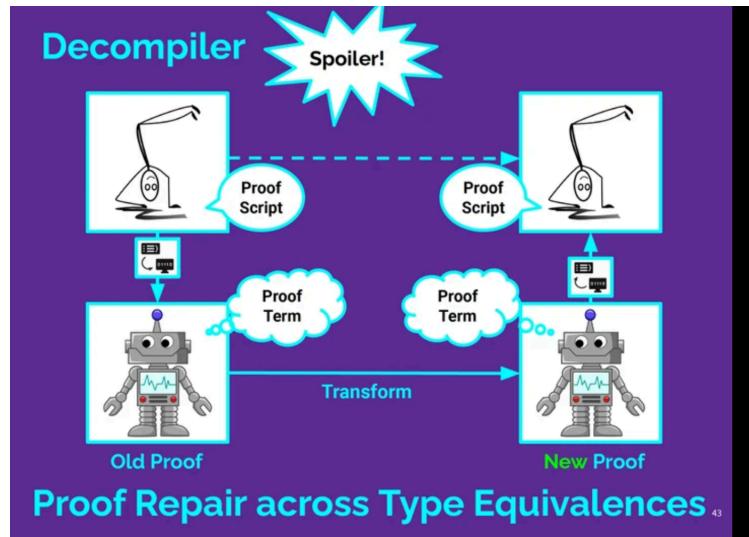
- transport: 以homo理论就是通过这些equivalence来rewriting
 - 实际做的是把transform当做一个proof term transformation来实现
 - 可以在每次change之后删除旧的版本



- 把definition的等价性转换为proposition的等价性



- decompiler



- add new indices to data type

Adding New Indices

```
Inductive list T : Type := 
| nil : list T
| cons : T →
  list T →
  list T.
```

```
Inductive vector T : nat → Type := 
| nil : vector T 0
| cons : T → ∀ (n : nat),
  vector T n →
  vector T (S n).
```

Spoiler! $\Sigma(l:\text{list } T).\text{length } l = n \simeq \text{vector } T n$

Proof Repair across Type Equivalences

vector 是 list 的长度索引(orange).

- change the inductive structure of data type

Changing Inductive Structure

```
Inductive positive :=
| xI : positive → positive
| x0 : positive → positive
| xH : positive.
```

```
Inductive nat :=
| 0 : nat
| S : nat → nat.
```

```
Inductive N :=
| NO : N
| Npos : positive → N.
```

Spoiler! $\text{nat} \simeq \text{N}$

Proof Repair across Type Equivalences

一元自然数 nat

二进制自然数 N (0:N0, Npos: positive的 (1:xH、左移加1 (xI) 、左移加0 (xO)))

nat和N是等价的，但是不同的inductive structure

definitional equalities over nat -> propositional equalities over N.

- 实验：在coq标准库，25秒修复

Saving Work

1	Classic benchmark
2	User study benchmark, part 1
3	User study benchmark, part 2
4	Prohibitive change
5	External example 1
6	External example 2
7	External example 3
8	Industrial use ... and more!

Proof Repair across Type Equivalences 47

verifying a modified program can be easier than verifying the original the first time around, in practical use cases.