

Formal Verification for Ethereum

Amsterdam, 3 May 2017
Yoichi Hirai

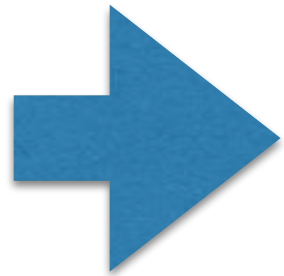
“formal” verification?



<http://www.telegraph.co.uk/culture/tvandradio/10803323/Why-cant-we-make-drama-like-The-Pallisers-anymore.html>

“formal” comes from formalised math

Usual Math

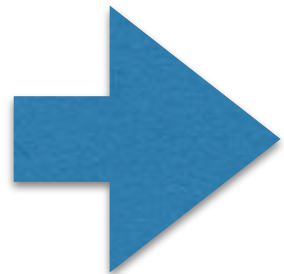


idea

correct but boring

https://en.wikipedia.org/wiki/Emmy_Noether#/media/File:Noether.jpg

Formalised Math

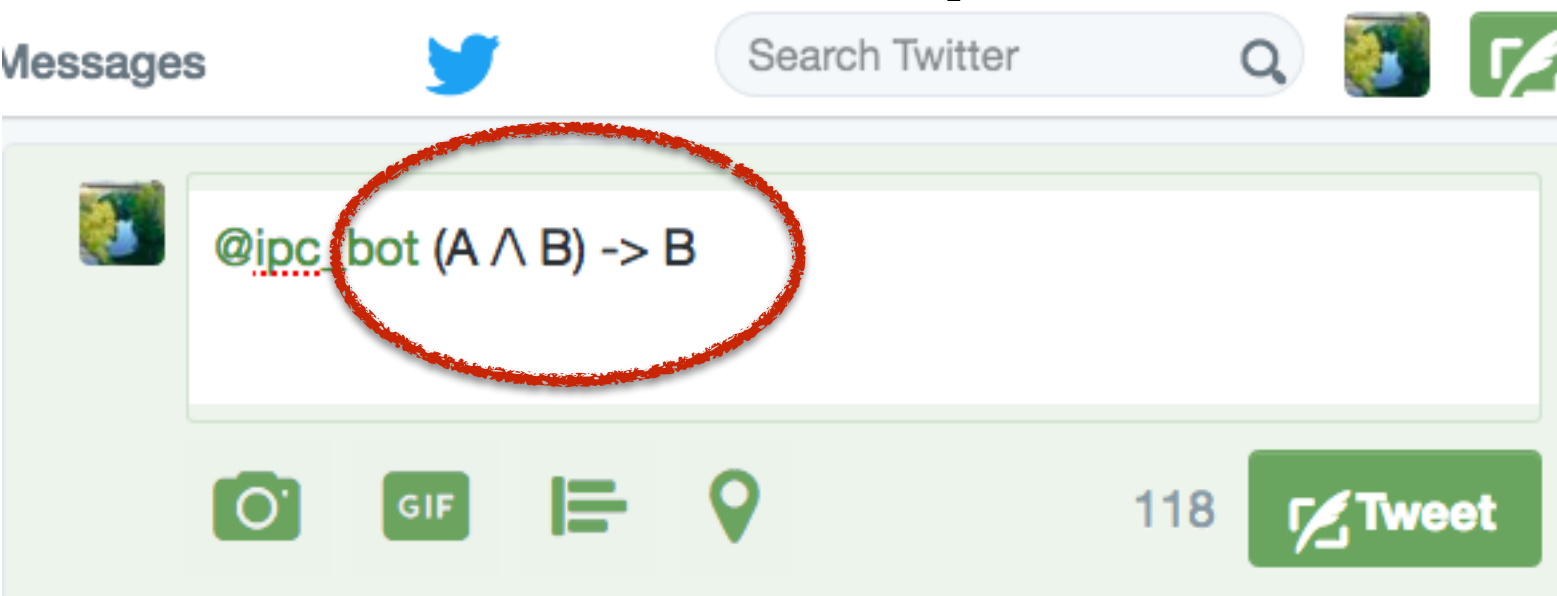


no idea

just
looking at
“form”

correct

formal proofs on twitter



$$\frac{\frac{P \vee \neg P}{\neg P} \quad \overline{P}}{P \vee \neg P} \quad \text{IPC bot}$$
 @ipc_bot


.@pirapira Provable. (pwl)

$A \wedge B \rightarrow B$:
 Provable.
 Proof tree (intuitionistic):

$$\frac{\frac{[A \wedge B]_1}{B} \wedge E_2}{A \wedge B \rightarrow B} \rightarrow I(1)$$

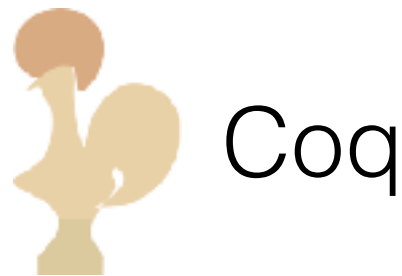
8:00 PM - 31 Oct 2016

Use proof checkers against lots of cases

- Kepler's conjecture “ is the most compact”
- one needs to see all other ways are less efficient
- This involves checking **lots of corner cases**.
Flyspeck project (led by Thomas Hales) used
Isabelle and HOL-light
- **(That sounds useful for software development too.)**

Proving software correct

- using interactive proof assistants

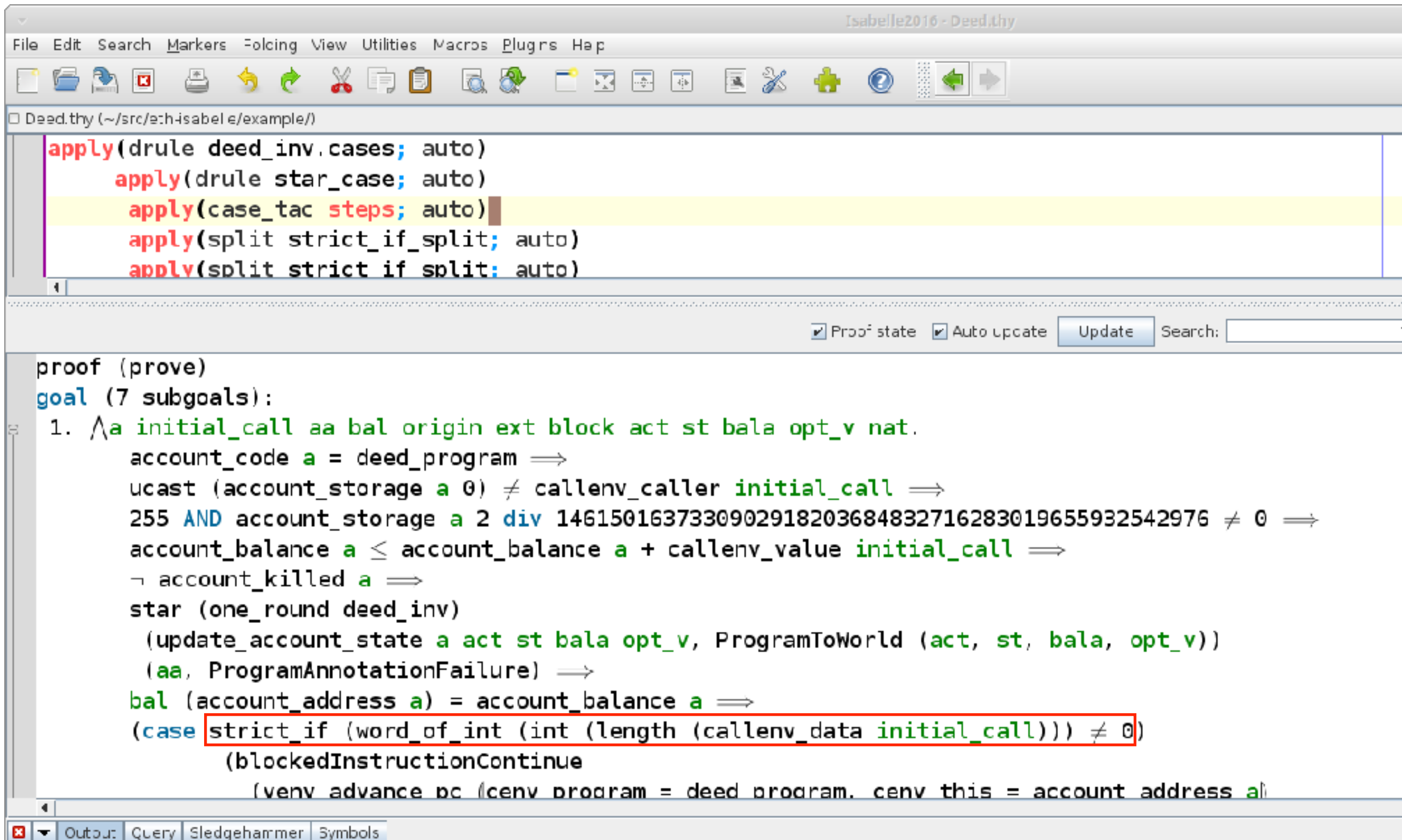


- they use only ~ 20 inference rules to derive the whole math
- ... and that code matches specification

Steps for Proving Ethereum contracts correct

- Ethereum Virtual Machine for theorem provers
- Test the EVM in the provers against other implementations
- Use the EVM for proving byte code correct against specifications

Proving smart contracts correct!



The screenshot shows the Isabelle2016 IDE interface. The title bar indicates the file is 'Isabelle2016 - Deed.thy'. The menu bar includes File, Edit, Search, Markers, Folding, View, Utilities, Macros, Plugins, and Help. The toolbar contains various icons for file operations, editing, and navigation. The main editor window displays the file 'Deed.thy' at the path '~/src/eth-isabelle/example/'.

The code in the editor is as follows:

```
apply(drule deed_inv.cases; auto)
  apply(drule star_case; auto)
  apply(case_tac steps; auto)
  apply(split strict_if_split; auto)
  apply(split strict_if_split; auto)
```

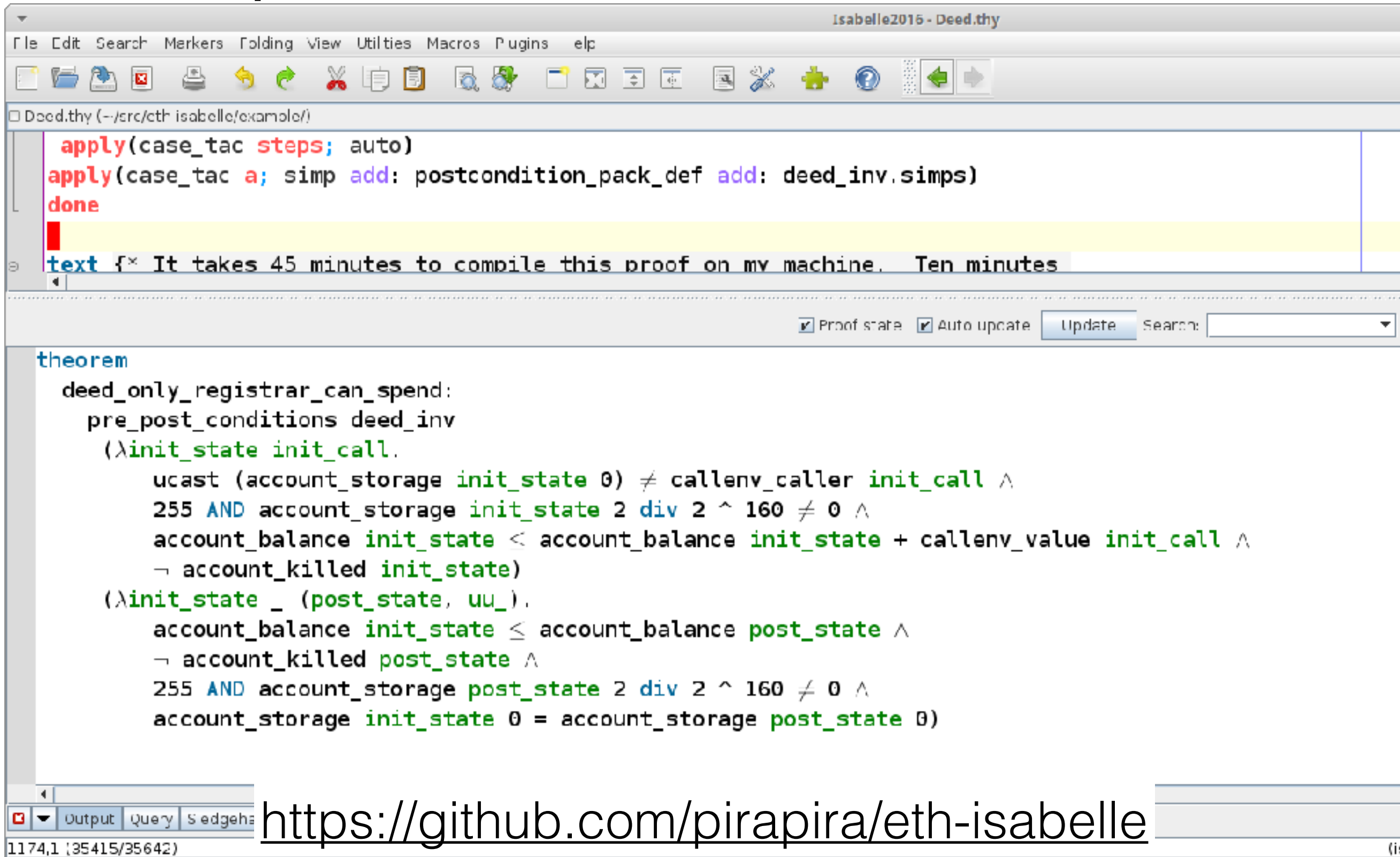
Below the code editor, there are checkboxes for 'Proof state' and 'Auto Update', an 'Update' button, and a search field.

The proof script is shown in the lower pane:

```
proof (prove)
goal (7 subgoals):
  1.  $\bigwedge a$  initial_call aa bal origin ext block act st bala opt_v nat.
    account_code a = deed_program  $\implies$ 
    ucast (account_storage a 0)  $\neq$  callenv_caller initial_call  $\implies$ 
    255 AND account_storage a 2 div 1461501637330902918203684832716283019655932542976  $\neq$  0  $\implies$ 
    account_balance a  $\leq$  account_balance a + callenv_value initial_call  $\implies$ 
     $\neg$  account_killed a  $\implies$ 
    star (one_round deed_inv)
      (update_account_state a act st bala opt_v, ProgramToWorld (act, st, bala, opt_v))
      (aa, ProgramAnnotationFailure)  $\implies$ 
    bal (account_address a) = account_balance a  $\implies$ 
    (case strict_if (word_of_int (int (length (callenv_data initial_call))))  $\neq$  0)
      (blockedInstructionContinue
        (venv advance pc (cenv program = deed program. cenv this = account address a)
```

The bottom status bar shows tabs for 'Output', 'Query', 'Sledgehammer', and 'Symbols'.

The proof finishes somehow



The screenshot shows the Isabelle2015 IDE interface. The title bar reads "Isabelle2015 - Deed.thy". The menu bar includes "File", "Edit", "Search", "Markers", "Folding", "View", "Utilities", "Macros", "Plugins", and "elp". The toolbar contains various icons for file operations, editing, and proof navigation. The main editor window displays the file "Deed.thy" with the following content:

```
apply(case_tac steps; auto)
apply(case_tac a; simp add: postcondition_pack_def add: deed_inv.simps)
done
text {* It takes 45 minutes to compile this proof on my machine. Ten minutes
```

Below the editor, there is a status bar with checkboxes for "Proof state" and "Auto update", an "Update" button, and a search field. The bottom panel shows a "theorem" section with the following text:

```
theorem
  deed_only_registrar_can_spend:
    pre_post_conditions deed_inv
      (λinit_state init_call.
        ucast (account_storage init_state 0) ≠ callenv_caller init_call ∧
        255 AND account_storage init_state 2 div 2 ^ 160 ≠ 0 ∧
        account_balance init_state ≤ account_balance init_state + callenv_value init_call ∧
        ¬ account_killed init_state)
      (λinit_state _ (post_state, uu).
        account_balance init_state ≤ account_balance post_state ∧
        ¬ account_killed post_state ∧
        255 AND account_storage post_state 2 div 2 ^ 160 ≠ 0 ∧
        account_storage init_state 0 = account_storage post_state 0)
```

At the bottom left, there is a status bar showing "1174,1 (35415/35642)".

<https://github.com/pirapira/eth-isabelle>

Did you prove the right thing?

- The account should not do anything wrong.

The balance should not decrease unless an authorised account tells so.

Did you prove the right thing?

- The account should not do anything wrong.

The balance should not decrease unless an authorised account tells so.

A non-authorised account cannot authorise any account.

It's not just about one Ethereum contract...

Verifying Ethereum as a Whole

- Theorem (Sami Mäkelä):
No Ethereum transaction can increase the total amount of Ether.
- Q. How can the total amount of Ether decrease?

Casper

What is Casper

- Ethereum's coming consensus mechanism.
- Several different Casper protocols
<https://github.com/ethereum/research/tree/master/casper>
<https://github.com/ethereum/research/tree/master/casper3>
<https://github.com/ethereum/research/tree/master/casper4>
Vlad's Casper
Meredith's Casper(s)
- Not easy to comprehend everything

Consensus

- The whole thing is for avoiding forks (or double-spends)
- PBFT (practical byzantine fault tolerance) has “2/3 honest implies no fork”
- To make it cryptoeconomic, we need:
“If a fork happens, 1/3 of the deposits can be forfeited”

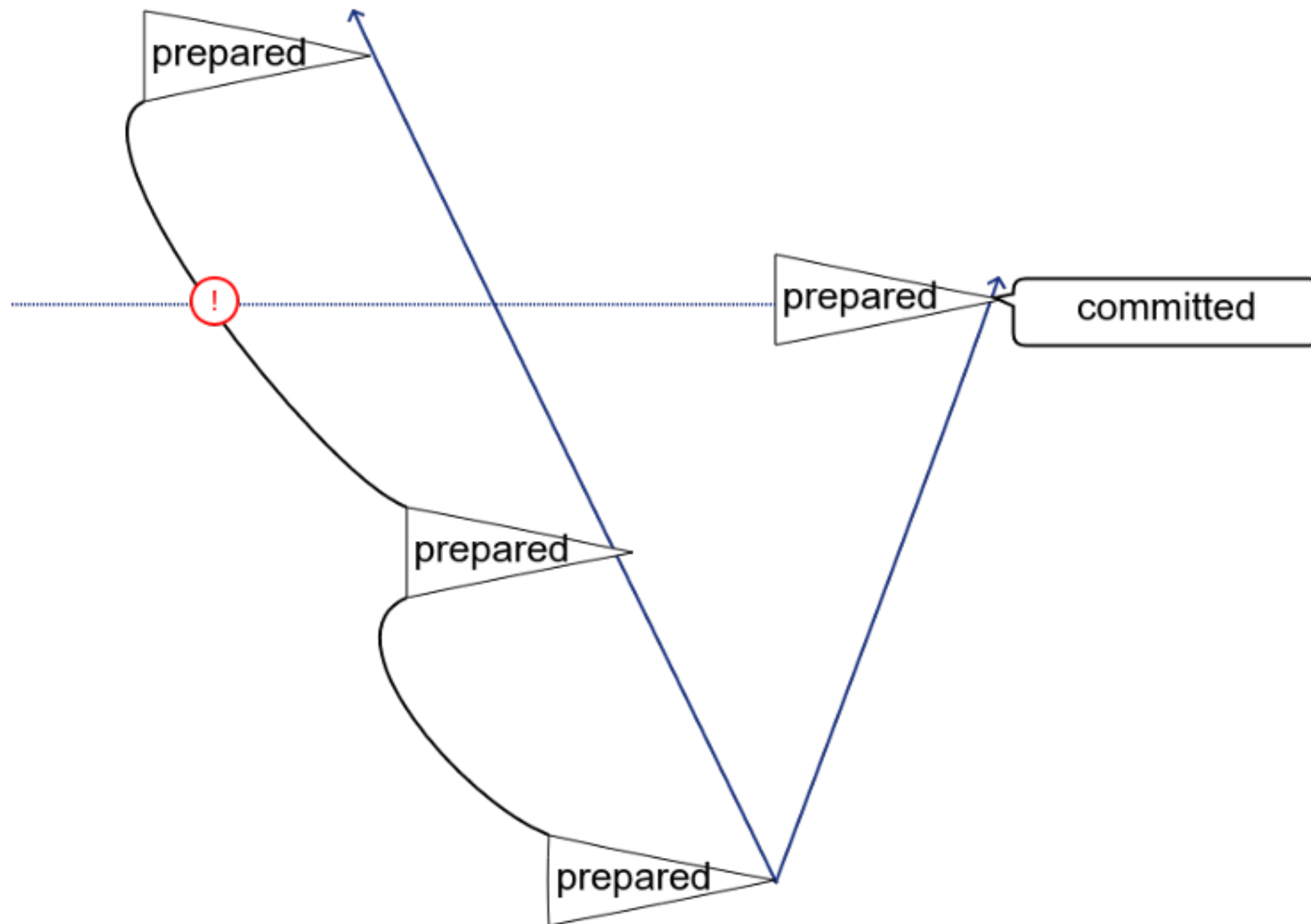
Proof-of-stake requires blaming bad behaviours

- “If 2/3 are honest, everything stays good” is not enough
- “if something goes bad, some participants can be penalised” is better
- Alice: “I sent it”
Bob: “I didn’t receive it”
- Blaming a single party is much better.

Slashing conditions

- if a fork happens, some $1/3$ should be blamed for violating slashing conditions
(signing contradicting “commit” messages /
signing “commit” messages without evidence /
signing “prepare” messages without evidence /
signing “commit” message between two “prepare”)
- many modes of failures because everyone can do whatever
- theorem prover to check all failure modes

Whenever there is a fork,
some slashing condition is violated



but the pictures help only as much.

theorem

lemma *accountable-safety* :

validator-sets-finite $s \implies$

$v \geq 0 \implies$

fork-with-commits $s \ (h, v) \ (h1, v1) \ (h2, v2) \implies$

$\exists \ h' \ v'.$

ancestor-descendant-with-chosen-validators $s \ (h, v) \ (h', v') \wedge$

one-third-of-fwd-or-rear-slashed $s \ h'$

$\langle proof \rangle$

Links

- @pirapira on Twitter
- pirapira on GitHub
- github.com/pirapira/eth-isabelle
Smart contract verification
- github.com/pirapira/pos
Casper verification
- yoichi@ethereum.org