Proofgold: Blockchain for Formal Methods

Chad E. Brown, Thibault Gauthier, Cezary Kaliszyk, Josef Urban

HIM Trimester “Prospects of Formal Math’
July 2, 2024, Bonn
Supported by ERC PoC Project FormalWeb3

Overview

- Proofgold: Cryptocurrency network supporting formal logic and mathematics
- Built on Qeditas codebase, with modifications
- Combines proof-of-stake and proof-of-burn (using Litecoin)
- Allows publishing theories, definitions, conjectures, and proofs
- Includes a bounty system for incentivizing proofs
- A lot of links to the Egal/Megalodon systems (C. Brown)
- However also proofs from HOL4 (Gauthier) and Prover9/Ivy

Pointers

- Initial 2020 announcement: `https://memo.cash/a/b25b6e856f`
- Explorer: `https://formalweb3.uibk.ac.at/pgbce/`
- Clients:
  - Lava (best) `http://proofgold.net/`
  - Core (first): `https://prfgld.github.io/`
  - Lite (lightweight): `https://github.com/dalcoder/proofgoldlite`
  - (Love: Extra features for bitcoin swaps)
- Archived forum: `https://prfgld.github.io/forum`
- Paper: `https://doi.org/10.4230/OASIcs.FMBC.2022.4`
- Formalweb3 ERC PoC poject: `https://formalweb3.uibk.ac.at/`

Brief History and Related

- Bitcoin – 2008/9 (white paper vs launch)
- Litecoin – 2011 (faster, smaller, cheaper)
- 2014 MathGate 14BTC treasure hunt (exact proofs)
- Qeditas – IOHK 2016 (never launched)
- DalilCoin – fork of Qeditas 2017
- Mathcoin – Su 2018 (ideas paper?)
- ProofGold – June 2020, Blake Keiler, fork of Qeditas/DalilCoin
- PG Lava – since 2021/22
- PG Lite – 2022
- PG Explorer – 2023/24
- Megalodon wiki linked to PG Explorer - 2024:
- `https://github.com/mgwiki/mgw_test`
  `https://mgwiki.github.io/mgw_test/`

MGWiki: A Collaborative Platform for Megalodon proofs

- Wiki for formal mathematics using the Megalodon system
- Based on higher-order Tarski-Grothendieck set theory
- Online editing and automatic checking of Megalodon files
- Workflow: Clone/fork repo, edit/add files, commit/push
- Automatic checking and HTML generation
- Pull requests for contributions to main repo
- Integration with Proofgold blockchain for conjectures and bounties
- Automatic generation of Proofgold documents (.pfg) from Megalodon files (.mg)
- Bounties viewable on explorer: `https://formalweb3.uibk.ac.at/pgbce/bounties.php`
- Example MG code: `https://mgwiki.github.io/mgw_test/Part12.mg.html#sqrt_SNo_nonneg_mon_strict`

Example Screenshots - mgwiki - surreal numbers

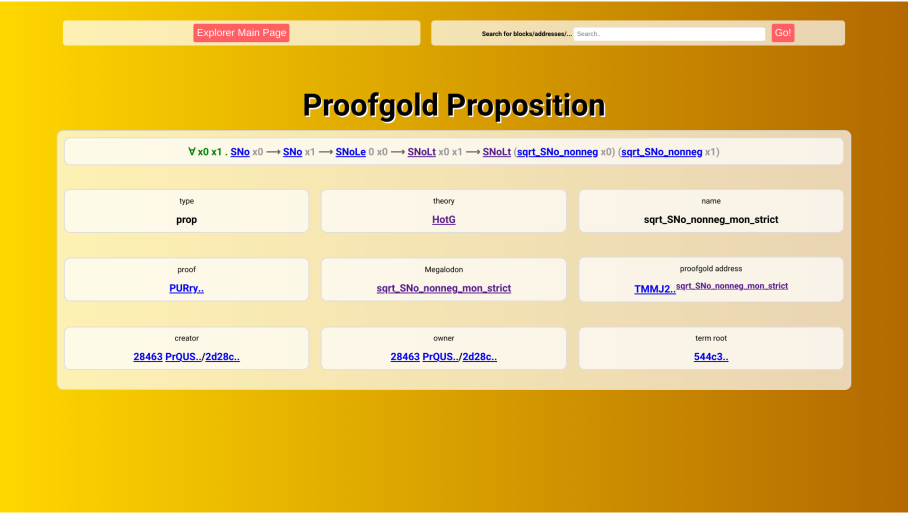
Theorem. (sqrt_SNo_nonneg_mon_strict)

```
∀x y, SNo x → SNo y → 0 ≤ x → x < y → sqrt_SNo_nonneg x < sqrt_SNo_nonneg y
```

In Proofgold the corresponding term root is 544c39... and proposition id is 02624d...

```
Proof:
  Let x and y be given.
  Assume Hx Hy Hxnn Hxy.
We prove the intermediate claim LsxS: SNo (sqrt_SNo_nonneg x).
    An exact proof term for the current goal is SNo_sqrt_SNo_nonneg x ?? ??.
We prove the intermediate claim Lsxnn: 0 ≤ sqrt_SNo_nonneg x.
    An exact proof term for the current goal is sqrt_SNo_nonneg_nonneg x ?? ??.
We prove the intermediate claim Lynn: 0 ≤ y.
    Apply SNoLe_tra 0 x y SNo_0 ?? ?? ?? to the current goal.
    We will prove x ≤ y.
    Apply SNoLtLe to the current goal.
    An exact proof term for the current goal is Hxy.
We prove the intermediate claim LsyS: SNo (sqrt_SNo_nonneg y).
    An exact proof term for the current goal is SNo_sqrt_SNo_nonneg y ?? ??.
We prove the intermediate claim Lsynn: 0 ≤ sqrt_SNo_nonneg y.
    An exact proof term for the current goal is sqrt_SNo_nonneg_nonneg y ?? ??.
  Apply SNoLtLe_or (sqrt_SNo_nonneg x) (sqrt_SNo_nonneg y) ?? ?? to the current goal.
    Assume H2.
    An exact proof term for the current goal is H2.
    Assume H2: sqrt_SNo_nonneg y ≤ sqrt_SNo_nonneg x.
    We will prove False.
    Apply SNoLt_irref x to the current goal.
    We will prove x < x.
    Apply SNoLtLe_tra x y x Hx Hy Hx Hxy to the current goal.
    We will prove y ≤ x.
    rewrite the current goal using sqrt_SNo_nonneg_sqr y ?? ?? (from right to left).
    rewrite the current goal using sqrt_SNo_nonneg_sqr x ?? ?? (from right to left).
    We will prove sqrt_SNo_nonneg y * sqrt_SNo_nonneg y ≤ sqrt_SNo_nonneg x * sqrt_SNo_nonneg x.
    Apply nonneg_mul_SNo_Le2 (sqrt_SNo_nonneg y) (sqrt_SNo_nonneg y) (sqrt_SNo_nonneg x) (sqrt_SNo_nonneg x) ?? ?? ?? ?? ?? ?? ?? ?? to the
    current goal.
```

Example Screenshots - PG explorer - surreal numbers



Example Screenshots - PG explorer - largest bounties

Description of bounties

Open bounties

TMHZ9..	FermatsLastTheorem	5,000.00
cc749..	TwoRamseyProp_4_5_25	800.00
7c52e..	MetaCat_struct_b_b_r_e_e_ordered_field_left_adjoint_forgetful	750.00
e16e0..	MetaCat_struct_b_b_e_e_ring_left_adjoint_forgetful	750.00
771a0..	MetaCat_struct_b_b_r_e_e_left_adjoint_forgetful	750.00
c7343..	MetaCat_struct_b_b_e_e_cring_left_adjoint_forgetful	750.00
ea9f0..	MetaCat_struct_b_b_e_crng_left_adjoint_forgetful	750.00
a33bd..	MetaCat_struct_b_b_e_e_left_adjoint_forgetful	750.00
08a75..	MetaCat_struct_b_b_e_rng_left_adjoint_forgetful	750.00
39e48..	MetaCat_struct_b_loop_left_adjoint_forgetful	750.00
3295d..	MetaCat_struct_b_abelian_group_left_adjoint_forgetful	750.00
0887e..	MetaCat_struct_b_semigroup_left_adjoint_forgetful	750.00
b66fd..	MetaCat_struct_c_Hausdorff_topology_left_adjoint_forgetful	750.00
63ebf..	MetaCat_struct_b_quasigroup_left_adjoint_forgetful	750.00
5f551..	MetaCat_struct_b_b_e_left_adjoint_forgetful	750.00
59fb4..	MetaCat_struct_c_topology_left_adjoint_forgetful	750.00
29f18..	MetaCat_struct_b_b_e_e_semiring_left_adjoint_forgetful	750.00
1ef6e..	MetaCat_struct_b_b_e_e_field_left_adjoint_forgetful	750.00
2570d..	MetaCat_struct_b_group_left_adjoint_forgetful	750.00
ed079..	MetaCat_struct_c_T1_topology_left_adjoint_forgetful	750.00

More open bounties

Collected bounties

not_TwoRamseyProp_3_6_17	not_TwoRamseyProp_3_6_17	800.00
not_TwoRamseyProp_3_5_13	not_TwoRamseyProp_3_5_13	800.00
TwoRamseyProp_3_6_18	TwoRamseyProp_3_6_18	800.00
not_TwoRamseyProp_4_5_24	not_TwoRamseyProp_4_5_24	800.00
TwoRamseyProp_3_5_14	TwoRamseyProp_3_5_14	800.00
7f3dc..	MetaCat_struct_b_monoid_left_adjoint_forgetful	750.00
5da2f..	MetaCat_struct_c_left_adjoint_forgetful	750.00
1e88d..	MetaCat_struct_r_graph_left_adjoint_forgetful	750.00
8dcfe..	MetaCat_struct_r_per_left_adjoint_forgetful	750.00
82000..	MetaCat_struct_r_ord_left_adjoint_forgetful	750.00
a69df..	MetaCat_struct_u_bij_left_adjoint_forgetful	750.00
123cf..	MetaCat_struct_r_partialord_left_adjoint_forgetful	750.00
49304..	MetaCat_struct_u_idem_left_adjoint_forgetful	750.00
538f6..	MetaCat_struct_r_wellord_left_adjoint_forgetful	750.00
8822b..	MetaCat_struct_u_inj_left_adjoint_forgetful	750.00
ed3d2..	MetaCat_struct_u_left_adjoint_forgetful	750.00
57ed9..	MetaCat_struct_p_nonempty_left_adjoint_forgetful	750.00
80d3d..	MetaCat_struct_r_equivreln_left_adjoint_forgetful	750.00
c3ca2..	MetaCat_struct_p_left_adjoint_forgetful	750.00
301a5..	MetaCat_struct_r_left_adjoint_forgetful	750.00

More closed bounties

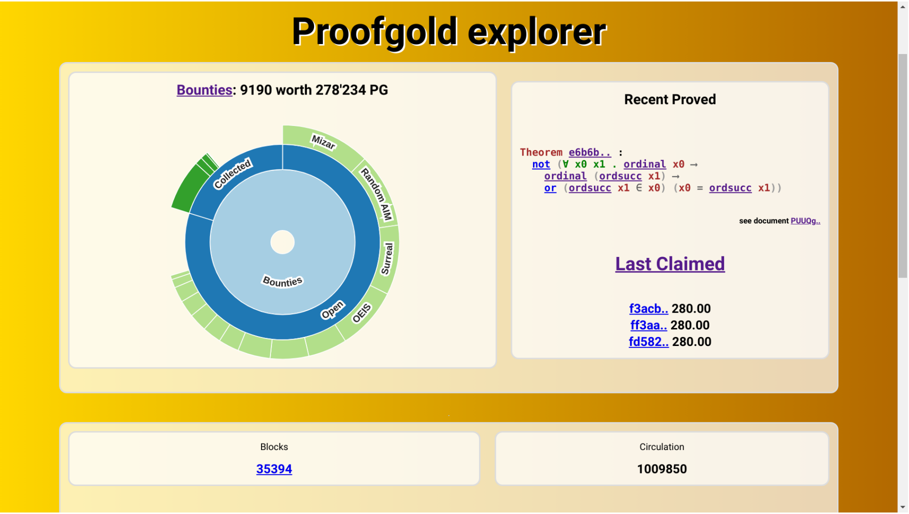
Open sum	Collected sum
304,109.67	206,580.88

Example Screenshots - PG explorer - theories

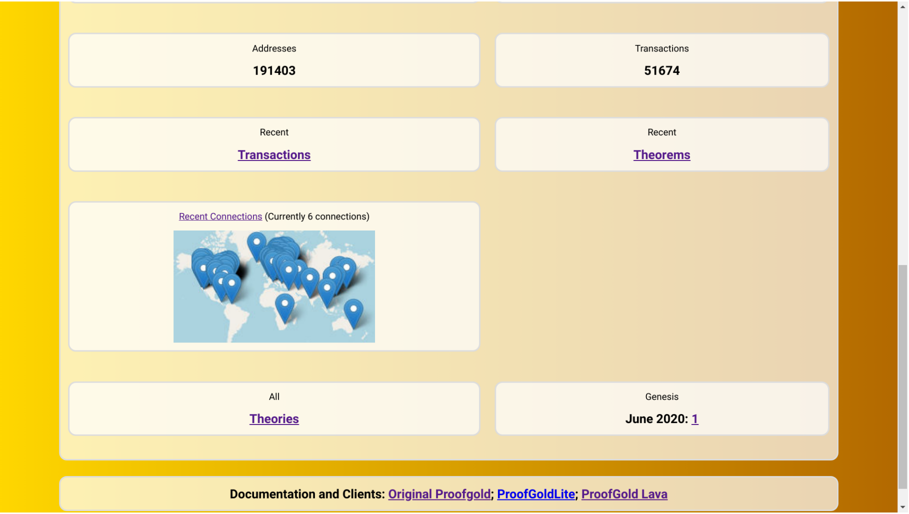
Proofgold Theories

Theory	Asset	Address	Publisher
HOAS	a0162..	PUPsB..	PrGxv..
HotG	205bb..	PUQdE..	Pr6Pc..
IHOLne	76a90..	PUTFi..	Pr8qe..
HoTg	5082b..	PUgom..	PrGxv..
HF	-	-	-

Example Screenshots - PG explorer - statistics



Example Screenshots - PG explorer - statistics



Core Logic

- Based on intuitionistic higher-order logic (IHOL)
- Simple types with base types o (propositions) and i
- Implication and universal quantifier as primitives
- Other logical constructs defined impredicatively
- Natural deduction proof system
- Proof terms for Curry-Howard correspondence

Proofgold Theories

- HF Theory: Built-in theory of hereditarily finite sets
- HOTG Theories: Two axiomatizations of higher-order Tarski Grothendieck set theory
- HOAS Theory: For reasoning about syntax with binding
- Theories are isolated from each other
- Users can publish new theories and develop them

Bounty System

- Users can attach bounties to propositions
- Initial automated bounties on pseudorandom propositions
- Later bounties on meaningful mathematical problems
- Incentivizes proof development and formalization
- Potential for supporting large formalization projects

Types of Bounties

- Ramsey Graphs: e.g. $R(4,5) = 25$ (recent formal proof by Gauthier)
- Mizar: 1400 *hard* Mizar ATP problems motivated by AITP research
- OEIS: 1k problems on equivalence of OEIS programs (Alien Coding)
- Surreal numbers: related to surreal numbers devel
- Category theory: Properties of specific categories
- AIM Conjecture: Related to loops and inner mappings
- Quantified Boolean Formulas (QBF)
- Set Constraints: Challenges involving set variable instantiation
- Higher-Order Unification: unification problems in HOL
- Untyped Combinator Unification
- Abstract HF: Problems about hereditarily finite sets
- Diophantine Modulo: Polynomial equations with modular arithmetic
- Diophantine: Equations or inequalities with polynomials over HF sets
- Random: General propositions with controlled generation

Problems with the Proofgold Core Client

- Proofgold Core: released client for participating in the Proofgold network
- Core has several problems:
  - It's slow (inefficient).
  - It crashes (unstable).
  - Sometimes the database gets corrupted and one has to resync from scratch (or use a recent backup).
- Proofgold Lava: our improved client.

Proofgold Lava Improvements

- Database: Use GDBM instead of Core's file based approach.
- Cryptography: Instead of Core's OCaml implementations use:
  - Elliptic curve: Use Harrison's verified code or Bitcoin's crypto implementation.
  - SHA256: Use Bitcoin's implementation
- Plus other changes to networking and proof checking
- These changes already make Lava orders of magnitude faster than Core, and make Lava more stable than Core.
- Proofgold Lava has been quite stable (running for months/year).
- An alpha release is available at proofgold.net .
- Lava and the related Proofgold blockchain explorer developed at `https://github.com/cezaryka/proofgold-lava`

Proving a Bounty in HOL4: Import

```
Bounty in Proofgold:
Ex X0 set Ap Ap and Ap Ap Subq X0 Ap Power Ap Power Ap Power
Ap Power Empty All X1 set Imp All X2 set Imp Ap Ap Subq X2
X1 All X3 set All X4 set Imp Ap Ap and Ap not Ap Ap tuple_p
X3 X4 Ap Ap and Ap exactly5 X0 Ap not Ap atleast2 X2 Ap
Ap SNo_ Ap Sing Ap SNoLev X0 X4 Eq set X1 X1
```

Bounty in HOL4 : $\exists X0.\ \mathit{Subq}\ X0\ (\mathit{Power}\ (\mathit{Power}\ (\mathit{Power}\ (\mathit{Power}\ \mathit{Empty})))) \wedge$
$\forall X1.\ (\forall X2. \mathit{Subq}\ X2\ X1 \Rightarrow \forall X3\ X4.\ \neg \mathit{tuple\_p}\ X3\ X4 \wedge \mathit{exactly}5\ X0 \wedge \neg \mathit{atleast}2\ X2 \Rightarrow$
$\mathit{SNo\_}\ (\mathit{Sing}\ (\mathit{SNoLev}\ X0))\ X4) \Rightarrow X1 = X1$

Import requires creating a copy of the HF theory in HOL4:

- mapping logical constants ($\wedge$, $\forall$, ...),
- new HOL4 definitions correspond to HF definitions (*Subq*, *Power*, ...),
- new HOL4 axioms in corresponding to HF axioms.

Proving a Bounty in HOL4: Automation

Manually proving bounties using HOL4 kernel rules and tactics is possible but time-consuming, that is why we have implemented the following automated process:

- Hol(y)Hammer: translate to external provers (Vampire,Eprover,Z3) and returns lemmas used in the proof (if found).
- Metis: minimize the number of necessary lemmas by trying to prove the bounty with subsets of the returned lemmas.
- Custom internal resolution prover that creates a small proof: tracking dependencies, making local definitions for large terms, carefully handling of CNF normalization steps and resolutions steps.

Proving a Bounty in HOL4: Export

- Record HOL4 kernel steps provided by a manual proof or the custom internal prover.
- Simulate each HOL4 kernel step by possibly multiple Proofgold kernel steps.
- Export the proof: create a Proofgold document and submit it to the blockchain for further verification.

The HOL4 interface can also be used to:

- create and prove Proofgold conjectures whether or not they are bounties,
- export the HOL4 standard library to Proofgold. (in practice, large HOL4 proofs are an issue).

Large Formalization Projects

- Potential use of bounty system for projects like Flyspeck
- Split formalization into parts with individual bounties
- Allow wider community participation
- Increase rewards for harder parts as needed
- Possible applications: Fermat's Last Theorem, Classification of Finite Simple Groups

Blockchain Features

- Combines proof-of-stake and proof-of-burn (using Litecoin)
- Theories and proofs recorded on the blockchain
- Ownership of propositions tracked
- Bounties can only be claimed by owners of proven propositions
- Commitment system to prevent frontrunning of proofs

Practical Considerations

- Block size limit of 500KB restricts proof size
- Large proofs must be split into lemmas across multiple blocks
- Proof checking has computational limits to prevent "poison proofs"
- Theories are isolated to contain potential inconsistencies

Current State and Future Work

- Active network with theories, proofs, and bounties
- Improved client implementation (Proofgold Lava)
- HOL4 interface for automated bounty mining
- Potential for supporting large collaborative formalizations
- Ongoing development and community engagement

Conclusion

- Proofgold combines blockchain technology with formal methods
- Provides incentives for proof development and formalization
- Offers potential for collaborative large-scale projects
- Challenges remain in scalability and wider adoption
- Innovative approach to advancing formal mathematics and verification

FormalWeb3 ERC PoC Project – C. Kaliszyk

WebFormal3 HOME SUMMARY TEAM PUBLICATIONS

ERC Project FormalWeb3

Web3 Platform for Formal Mathematics

The ERC PoC Project "FormalWeb3" Grant no. 101156734 is realized at the CL group at the University of Innsbruck.

The project runs from February 2024 to July 2025 and its principal investigator (PI) is Cezary Kaliszyk.

The project builds on the success of our ERC project SMART and aims to build a holistic Web3 platform that will serve as a collaborative hub, enabling diverse stakeholders to collaboratively address practical verification and formalization challenges while rewarding users for their contributions.

universität innsbruck Computational Logic erc