Qeditas: A Formal Library as a Bitcoin Spin-Off

Bill White*

Draft of March 29, 2015

Abstract

Formalization of mathematical theories is a time consuming process for which there is currently little reward. We describe how block chain technology can be used to support the formalization of mathematics by encouraging and rewarding useful work while discouraging repeating the work of others. The block chain will contain a record of definitions made and propositions proven. In addition the block chain can be used as a registry to record the first participant to make a definition or prove a theorem as the owner of the object or proposition as intellectual property. Future users may be required to buy rights to make use of the object or proposition. Purchasing of these rights can be avoided by repeating the content, but at the expense of larger documents and increased fees to the creators of blocks.

1 Introduction

Qeditas is a project to apply block chain technology to support the construction of a library of formalized mathematics. Motivations for building such a library were spelled out in an anonymously published document in 1994 with the title The QED Manifesto [3]. In order to construct such a library there must be a clear record of which definitions have been made and which theorems have been proven. In addition the record may include unproven conjectures of interest. Qeditas will use a block chain to secure such a record in a decentralized manner. In addition a block chain can be used to reward those who make useful definitions and prove useful theorems.

Bitcoin [32] introduced the notion of a block chain to have a secure distributed record of currency transactions. The Ethereum project [46] plans to use a block chain to provide a distributed platform for computation. Qeditas will use a block chain to provide a distributed platform for deduction. Qeditas will also include an internal currency whose initial distribution will be based on a snapshot of the Bitcoin block chain. In other words Qeditas will be a Bitcoin spin-off [39].

*Bitmessage: BM-2cWvQoqzQPJS8CvuRi7DZ2iCud4m9FB1HA
Email: billwhite@bitmessage.ch BTC: 12pbhpqEg7cjaCLLcvdhJhBWGUQWkRK3zS

The task of building a formal mathematical library is enormous. Wiedijk estimated the work to formalize the mathematics that mathematicians "take for granted" as requiring 140 man-years [45]. Achieving such a goal will require the work of many independent people. The system in which the formalization takes place must allow people submit definitions, conjectures and theorems (with proofs) independently. Their contributions to the library should be appropriately rewarded and other users should be encouraged to build on previous work.

For the most part the work of formally proving theorems has been limited to graduate students who formalize certain areas of mathematics or computer science as part of obtaining their degrees. This is sometimes done as part of a larger project. One example of such a large project is the recently completed Flyspeck project [23] to formalize Hales' proof of the Kepler conjecture. Gonthier's work formalizing theorems about the classification of finite simple groups [20] provides another prominent example.

In the academic world *wealth* is largely measured in publications, and academics who create formalizations receive their reward in the form of the resulting publications. Outside of academia there is currently little reward for doing such work. To some degree this is surprising since theorem proving can be used to ensure properties of programs and protocols.

In Qeditas there will be two ways to be rewarded.

1. Users will be able to place bounties on conjectures and the bounties can be collected by the publisher of the first document resolving the conjecture.

2. If a user is the first to define an object or prove a proposition, then the user will be able to claim ownership of the object or proposition. In order for future users to make use of the definition or theorem without repeating the content, they may (at the discretion of the owner) be required to purchase corresponding rights.

The second point means that mathematical objects and propositions will become essentially a form of *decentralized intellectual property* (within the system) and the purchasing and usage of rights will be enforced by the network. This is similar to the requirement that academics include appropriate references to previous work in their publications. Note, however, that in the current academic world many publications have copyright restrictions and are often behind paywalls. In contrast all documents published in the Qeditas block chain will be freely available.

In this white paper we will give a high level description of the Qeditas project. The code is still being written and details are subject to change.

In Section 2 we describe the Qeditas currency units including the plan for an initial distribution taken from a snapshot of the Bitcoin block chain. The remaining currency units will be given as block rewards. The current plan is to use a lightweight block chain so that each member of the network need not permanently store, for example, every formal document. The consensus mechanism will likely be proof of stake [28, 14] in some combination with proof of

Full Name	Short Name	Basic Unit Factor	Corresponding Bitcoin Units
Cantor (Cantors)	cant (cants)	1	0.0001 satoshis
Frege (Freges)	freg (fregs)	10^3	0.1 satoshis
Church (Churches)	church (churches)	10^6	1 microbit
Zermelo (Zermelos)	zerm (zerms)	10^9	1 millibit
Fraenkel (Fraenkels)	fraenk (fraenks)	10^{12}	1 bitcoin
Grothendieck (Grothendiecks)	groth (groths)	10^{15}	1000 bitcoins

Table 1: Qeditas currency units

retrievability [31]. These choices are discussed in Section 3. We use a small example to show how formal documents can be specified and published into the block chain in Section 4. In Section 5 we extend the small example to demonstrate how ownership of objects and propositions can be recorded into the block chain. In Section 6 we describe how bounties can be placed on conjectures and later collected by someone who proves the conjecture (or possibly its negation). Currency units, ownership, rights, and bounties can generally be described as *assets* which are held at addresses. This is described in Section 7. All of the concepts above are generic and could be instantiated to many different theorem provers. In Section 8 we list some pros and cons of a few theorem provers which could be used as the underlying engine for Qeditas, leaving the definite choice for later.

2 Currency

One of the oft-repeated complaints about Bitcoin is that the same name was given to both the network and the currency units. This is remedied in print by writing “Bitcoin” for the network and “bitcoin” for the currency unit. Actually, the basic currency unit is called a “satoshi” and a bitcoin consists of 100 million satoshis. It is also possible that subdivisions of satoshis will be included at some point in the future.

As the value of bitcoin has increased, community members have engaged in long debates about the possible names to give to units between satoshis and bitcoins. In an attempt to preempt such debates in Qeditas, we give names for the units here. We do not use the term “qeditas” for any of the units. Instead we derive the names from the names of some mathematicians who have contributed to the understanding of the foundations of mathematics. The names are listed in Table 1 and the pronounciations are intended to be the same as the pronounciations of the names of the corresponding mathematicians. The names are given in full form (both singular and plural) and in a short abbreviated form (both singular and plural). For the remainder of the paper we will use the abbreviated names.

The intention is that one fraenk corresponds directly to one bitcoin (in terms of units, not value), and so the number of fraenks in Qeditas is capped at 21 million. The first (approximately) 14 million ($\frac{2}{3}$) will be distributed by taking a snapshot of the first 350,000 blocks of the Bitcoin block chain.[1] The remaining 7 million will be distributed as block rewards with the same schedule Bitcoin is using (starting from block 350,000). That is, the block reward will begin with 25 fraenks for the first 70,000 blocks and will then halve to 12.5 fraenks. After block 70,000 the block reward will halve each 210,000 blocks. A block time of 10 minutes will be targetted, so that the number of fraenks should always be approximately the same as the number of bitcoins. Note that since there are finer units in Qeditas than Bitcoin, very small Qeditas block rewards will continue after the Bitcoin block rewards have stopped.

Arguments in favor of such a snapshot distribution can be found in the "spin-off" thread begun by Peter R [39].[2] One can view the Bitcoin block chain as recording an efficient distribution of wealth in a voluntary environment. Bitcoin is sufficiently well-known in 2015 that each individual has had a chance to make an independent judgement about the idea. Consequently each person now holds the amount of bitcoins that reflects their own judgment.

3 Lightweight Ledger

Qeditas will use a lightweight ledger in the sense described in [44]. The idea is similar to the mini-blockchain scheme implemented in the cryptocurrency Cryptonite [8] as well as to the intended use of Merkle-Patricia trees in Ethereum [46]. The state of the ledger will be represented by a compact form of a trie combined with a Merkle structure (a Merkle-Patricia tree up to some details). The full trie allows one to look up the assets held by an address. These assets may include currency units but may also include formal documents, ownership information, rights and bounties.

We distinguish between assets *held* by an address and assets being *controlled* by addresses. An asset a is *held* by an address α if a can be found in the trie at address α. The asset itself contains *obligations* (see [44]) indication who can spend the asset. The obligations allow for the possibility of multi-signature requirements. It is possible for an address to hold an asset without being able to spend the asset.

The two most popular kinds of consensus mechanisms used by cryptocurrencies are proof of work and proof of stake. Proof of work (PoW) was first described by Back [4] and is used by Bitcoin. Proof of stake (PoS) was introduced by King and Nadal [28], and first implemented (as a proof of work/proof of

[1] To be precise, the snapshot will include all pay-to-public-key-hash addresses, including those derived from pay-to-public-key outputs. The snapshot will also include native multisig outputs. In addition, all pay-to-script-hash outputs will be included, but we cannot guarantee the Qeditas script interpreter is 100% compatible with the script interpreter of Bitcoin. A preliminary script interpreter has been written and tested on already spent pay-to-script-hash outputs in the Bitcoin block chain and it succeeds in over 99.8% of cases. Also, Bitcoin's genesis block is counted having height 0, and so the snapshot will include the block at height 349,999 but not the block at height 350,000.

[2] While the cited thread contains the primary discussion on the topic, the idea seems to be older. See for example the earlier thread begun by go1111111 [18].

stake hybrid) in Peercoin. Another consensus mechanism, proof of retrievability (PoR), described by Miller, et. al., is designed to support data preservation [31].

Since Qeditas is intended to support a library, some form of PoR seems to be a good choice. On the other hand, the amount of data being secured by Qeditas is likely to be relatively small (especially at first) and it is not clear that PoR is appropriate in this case. Conceivably, PoR could be used to ensure storage of the syntactic terms which hash to give addresses of objects and possibly storage of proofs of theorems. While the consensus mechanism is not yet fixed, it is likely to be some combination of PoS and PoR.

The original PoS mechanism used a notion of coin age. When coin age is coupled with a block reward proportional to the stake, the incentive to constantly stake (and thereby support the security of the network) is decreased. The Qeditas network will not suffer from this problem since it will have a fixed block reward independent of the stake of the forger of the block. This idea has been described earlier as "proof of stake definite" [14]. Qeditas may use a notion of coin age, but since there is a fixed block reward potential stakers maximize their reward by keeping their nodes online as much as possible.

The currency units held by an address α are part of the stake of that address. Accordingly, if an address holds some currency units, then the private key for α will be able to forge new blocks in the block chain. In order to check α holds the alleged stake it is sufficient to look at an approximation of the current state showing the stake is among the assets held by α. The approximation need not include information about assets held by any other address. More details can be found in [44].

Note that if β controls some currency units, then β can allow the asset to be held by α so that α can use them to forge blocks. This would allow a staker to keep the keys for spending offline while having the staking keys online. It would also allow for a non-staker (controlling β) to "loan" their currency units to a staker (controlling α). In return α might pay some of the block rewards back to β. Since β still controls the asset, the owner of the private key for β can spend the asset even though it is held by α. (Though the obligation may contain a block height before which the asset cannot be spent.)

There are strong arguments that PoS cannot provide the same level of distributed consensus as PoW [38]. One reason is that if someone had stake in the currency at one time, then this previously owned stake could, in principle, be used to create a fork of the block chain starting from that earlier time. In practice a PoS cryptocurrency can avoid a long range attack of this kind by preventing long reorganizations. For example, the cryptocurrency Nxt [13] disallows reorganization beyond 720 blocks. The cost of this solution is that if someone new enters the network they may need a way, outside the network itself, to determine the "correct" chain. Buterin calls such a criteria "weakly subjective" in contrast to the "objective" criteria used by Bitcoin [9]. Short range attacks are still possible, though simulations by Consensus Research show them to be unlikely even in the face of participants signing competing chains [10, 2, 40].

The initial distribution may provide a protection against certain kinds of attacks. In the first place, it would be very difficult for an individual or group to obtain 50% of the currency units since this would require obtaining the private keys corresponding to at least 3.5 million bitcoins as of block height 350,000. On the other hand, a 51% attack on a proof of stake coin only requires having more than 50% of the *actively staking* coins. There is a threat that if a single wealthy bitcoin holder decided to attack the Qeditas network before many others are participating, the network could be strangled in its cradle. The economic rationality of such an attack is questionable, since they would be doing work to destroy a block chain that would add to their wealth if left alone. Nevertheless, the threat exists. The hope is that the wide initial distribution will make it likely that for each participant with a certain amount of the distribution there will be others with a similar amount.

4 Documents

Some examples of formal mathematical libraries include the Mizar Mathematical Library [41], the Archive of Formal Proofs supported by Isabelle-HOL [33] and the Mathematical Components library of ssreflect [21]. In each case the library is made up of documents. Each of these documents extends the mathematical content of the library by including mathematical items, mainly definitions and theorems. The organization of these libraries tends to be handled by experts who have a familiarity with the current contents of the library. These experts can determine if the content is new and whether the document conforms to the standards of the library.

Since Qeditas will be decentralized, there will be no "expert" to filter out submissions. The protocol itself must ensure the submission meets the necessary criteria. In this section we walk through a small example of how a document might be created and published.

Suppose a user creates a document `Relns` which defines the converse of a relation, defines when a binary relation is symmetric and proves a theorem that the converse of a relation is symmetric if the relation is symmetric. The particular syntax and format of such a document depends on the system used to check the proofs. For now, we can remain system-independent by using informal mathematical language. The mathematical content of `Relns` is shown in Figure 1.

Note that the document contains a proof of the theorem. There are a variety of representations for proofs, but a fundamental requirement is that it must be easy for a proof checker to determine whether or not the proof is correct. We return to this issue in Section 8.

The first criteria for the document to be accepted by the network for publication is that it is formally correct. This ensures that only correct definitions and theorems will be included in the library. There are a number of other criteria which will be discussed in the remainder of the paper. For this section, let us consider how plagiarism is avoided.

Suppose the author of `Relns`, Alice, controls an address α. Suppose `Relns` is signed with the address α and submitted to the Qeditas network for publication.

Definition 1. Let R be a binary relation. The binary relation R^{-1} is defined such that $R^{-1}(x, y)$ holds if $R(y, x)$ holds.

Definition 2. A binary relation R is *symmetric* if $R(y, x)$ holds whenever $R(x, y)$ holds.

Theorem 1. Let R be a binary relation. If R is symmetric, then R^{-1} is also symmetric.

Proof. Assume R is symmetric and $R^{-1}(x, y)$ holds. By Definition 1, $R(y, x)$ holds. By symmetry of R and Definition 2, $R(x, y)$ holds. By Definition 1 again, $R^{-1}(y, x)$ holds. □

Figure 1: Mathematical Content of `Relns`

Another network participant, Bob, controlling an address β, could easily remove Alice's signature and replace it with his own. If Bob's version were confirmed before Alice's, then Alice would lose credit for having done the work.

A solution to this problem is to enforce the following protocol.

1. The author chooses a salt and includes it in the document.

2. The author computes a hash of the signed salted document and publishes it as an *intention*.

3. After the intention is sufficiently confirmed, the author releases the signed salted document to be published.

The network will only allow a document to be confirmed if it has a sufficiently confirmed intention. Now for an attacker to take credit for another document, the attacker would need to publish a new intention, wait for enough confirmations, and then attempt to publish the plagiarized document all before the original author's document is confirmed.

Publishing an intention and a document will likely require fees, and the fees will likely depend on the size of the document. This gives the first in-system use of the currency. We will see other in-system uses of the currency in the next sections.

5 Ownership and Rights

In this section we consider the notions of ownership of objects and propositions and rights of use. We use the example document `Relns` with the contents described in Figure 1 to guide the discussion.

In a document a formal definition declares a certain name to be an abbreviation for a certain syntactic term. Likewise, a formal theorem declares a name to be a reference to the fact that a certain syntactic term (representing a proposition) is provable (with the proof following the declaration). These syntactic terms can be hashed in order to assign a unique address corresponding for each term.

Consider the first definition in `Relns`. This defines an operation $-^{-1}$ on relations. As a λ-term [12] the definition can be written as

$$\lambda R \lambda x \lambda y . R(y, x).$$

This definition can then be converted to a nameless (de Bruijn) representation [16]

$$\lambda \_ \lambda \_ \lambda \_ . 2(0, 1).$$

This nameless version can be serialized and hashed to give an address. Let δ_1 be the address corresponding to the first definition. Likewise, we can compute an address δ_2 corresponding to the second definition and δ_3 corresponding to the proposition of the theorem.

Suppose Alice (the author of `Relns`) is the first to make both of the definitions, the first to mention the proposition of the theorem and the first to prove the theorem. In this case α (the address of Alice) will be marked as the owner of the objects δ_1 and δ_2 and the owner of both the object and proposition δ_3 when the document is published (and confirmed). This ownership information is held as a kind of asset at the addresses δ_1, δ_2 and δ_3 and will include a *royalty* requirement. The royalty requirement is used to determine under what conditions others may use the object or proposition. The owner may either allow everyone to freely use the item, may allow no one to use the item, or may allow others to purchase rights for each use. Ownership can also be transfered to a different address using the private key for α. The royalty information can be changed by the current owner.

Let us suppose that Alice publishes the document `Relns` and assigns ownership of δ_1, δ_2 and δ_3 to her address α. Suppose Alice allows δ_1 to be freely used, does not allow the use of δ_2 at all, and requires a payment of 3 zerms for each right to use δ_3.

Now suppose Bob, the controller of address β, wants to author a document extending Alice's work by proving that $(R^{-1})^{-1}$ is symmetric of R is symmetric. Bob has a number of options for doing this. We will discuss three options as documents `Relns2v1`, `Relns2v2` and `Relns2v3`.

In `Relns2v1` (see Figure 2) Bob copies Alice's work and adds one new theorem. The new theorem (Theorem 2) is easily proven using Alice's theorem twice. Theorem 2 is new and will have a corresponding address δ_4 which currently has no owner.

Bob can publish `Relns2v1` signed using the private key for his address β. The first three items already have an owner, Alice, and she will remain the owner of these three items. The address β for Bob will be assigned the owner of the new item δ_4.

Simply repeating Alice's work is not the best way to import previous work. In this case as in many others it would be a superior choice to only record the parts of the work that are needed. For this reason suppose Bob is unsatisfied and does not publish `Relns2v1`. Bob could examine his proof of Theorem 2 and recognize that he did not need the actual definitions of $-^{-1}$ or symmetric, but only needed the fact that the proposition of Theorem 1 has been proven.

Definition 1. Let R be a binary relation. The binary relation R^{-1} is defined such that $R^{-1}(x, y)$ holds if $R(y, x)$ holds.

Definition 2. A binary relation R is *symmetric* if $R(y, x)$ holds whenever $R(x, y)$ holds.

Theorem 1. Let R be a binary relation. If R is symmetric, then R^{-1} is also symmetric.

Proof. Assume R is symmetric and $R^{-1}(x, y)$ holds. By Definition 1, $R(y, x)$ holds. By symmetry of R and Definition 2, $R(x, y)$ holds. By Definition 1 again, $R^{-1}(y, x)$ holds. □

Theorem 2. Let R be a binary relation. If R is symmetric, then $(R^{-1})^{-1}$ is also symmetric.

Proof. Assume R is symmetric. Applying Theorem 1 with R we know R^{-1} is symmetric. Applying Theorem 1 again, this time with R^{-1}, we know $(R^{-1})^{-1}$ is symmetric. □

Figure 2: Mathematical Content of `Relns2v1`

Object 1. $(-)^{-1}$ is the object with address δ_1.

Object 2. *symmetric* is the object with address δ_2.

Known 1. Let R be a binary relation. If R is symmetric, then R^{-1} is also symmetric.

Theorem 2. Let R be a binary relation. If R is symmetric, then $(R^{-1})^{-1}$ is also symmetric.

Proof. Assume R is symmetric. Applying Known 1 with R we know R^{-1} is symmetric. Applying Known 1 again, this time with R^{-1}, we know $(R^{-1})^{-1}$ is symmetric. □

Figure 3: Mathematical Content of `Relns2v2`

Armed with this information Bob the definitions and first proof can be replaced with references in a new version called `Relns2v2` (see Figure 3).

The correctness of the proof of Theorem 2 in `Relns2v2` can still be checked. More information is needed to ensure that there really are objects with addresses δ_1 and δ_2. This can be verified by looking up δ_1 and δ_2 in the trie. Assuming they are previously defined objects, the information will be there. Furthermore, the system must verify that the proposition in Known 1 has been previously proven. This can be verified by computing the address corresponding to the proposition, δ_3, and then looking up the relevant information in the trie at this address.

The document `Relns2v2` is shorter than `Relns2v1`. Consequently it should be less expensive (in terms of fees) to publish it. In general fees are expected to encourage succinctness.

However, the network will not allow Bob to publish `Relns2v2`. In addition to checking δ_1 and δ_2 are objects and δ_3 is a known proposition, the permission to make use of them must be determined. Alice has allowed free use of δ_1, but Alice has not allowed use of δ_2 at all. In addition, Alice requires the purchase of rights to use δ_3. At the moment, Bob has no such rights.

Object 1. $(-)^{-1}$ is the object with address δ_1.

Definition 2. A binary relation R is *symmetric* if $R(y, x)$ holds whenever $R(x, y)$ holds.

Known 1. Let R be a binary relation. If R is symmetric, then R^{-1} is also symmetric.

Theorem 2. Let R be a binary relation. If R is symmetric, then $(R^{-1})^{-1}$ is also symmetric.

Proof. Assume R is symmetric. Applying Known 1 with R we know R^{-1} is symmetric. Applying Known 1 again, this time with R^{-1}, we know $(R^{-1})^{-1}$ is symmetric. □

Figure 4: Mathematical Content of `Relns2v3`

Armed with this further information, Bob purchases the rights to use δ_3 twice. He can do this by creating a transaction with outputs sending 6 zerms to α (the owner of δ_3) and sending 2 δ_3-*rights* to himself. Note that Alice need not be involved in this transaction. In the future Alice may change the royalty requirements for δ_3, but this will not affect Bob's right to use δ_3 twice. Purchasing such rights are the second in-system use of the currency.

Bob then creates a third document `Relns2v3` (see Figure 4). In this case Bob avoids the fact that Alice does not allow use of δ_2 by simply repeating the work of defining symmetry. Bob can then publish an intention to publish `Relns2v3` and later publish `Relns2v3`. The transaction publishing `Relns2v3` will spend the two rights purchased above, consuming them.

In the end the previous work that is repeated in vs. imported into a document will be (at least partially) economically determined based on fees (relative to the size of documents) and royalty requirements. If the owners of items determine new documents are opting to repeat work and pay higher fees, then these owners are likely to reduce the royalty requirements. However, as developments become increasingly complicated, repeating work becomes less feasible. The reason is simple: to repeat a definition or a proof, one will generally need to include other dependencies. In the most extreme case, to avoid all dependencies a document can include every definition and proof all the way down to the foundation.

6 Bounties

We next consider bounties on conjectures. This gives a third in-system use of the currency and allows users to guide the development the library. A document may include conjectures (unproven propositions) and include a bounty in the form of currency units. The bounty will be automatically paid out to the publisher of a document resolving the conjecture. By *resolving* we mean either proving the conjecture or proving its negation.

A similar notion of bounties for bitcoin is available at Sakaguchi's website `proofmarket.org` [42]. At `proofmarket.org` there is a list of Coq [29] and

Agda [34] propositions with bitcoin bounties. These can be collected by submitting formal proofs to the website. If the proofs are checked by the corresponding system to be correct, then the bounty is paid out. The bounty mechanism at `proofmarket.org` is centralized and carries counterparty risk.

Qeditas will handle bounties in a decentralized manner as follows. Suppose δ is the address of the syntactic term specifying a conjecture. All bounties will be held (as bounties, not currency) at address δ. Suppose Alice, with address α, publishes a document which contains a proof of the proposition. As a result of this publication, α will be entered as the owner of δ (as a proposition). The owner of a proposition will always be allowed to spend bounties held at the address back into currency units (to any address). To handle the case when Alice proves the negation of the conjecture, we say the owners of propositions will also be allowed to spend bounties held at the *negation* of propositions back into currency units.

Note that it is possible for a proposition to be *independent* – meaning neither the proposition nor its negation is provable. Using currency units to place bounties on independent propositions essentially *burns* the currency units.

The collection of bounties seems to have a "winner takes all" quality. However, as the conjectures become increasingly complicated, it seems likely that the document which resolves the conjecture is built from previous work. In practice, we assume the creators of this previous work required royalties, and they will be rewarded by the purchase of rights instead of the bounty. This means bounties can indirectly encourage users to publish intermediate results intended to build towards a solution to a conjecture with a bounty.

7 Assets

In previous sections we have described currency units, intentions, documents, ownership, rights and bounties. We combine these under the general notion of a *preasset.* A *preasset* can be combined with other information to give an *asset.* Here we follow the presentation in [44].

A *preasset* is one of the following:

- a currency unit (with a 64-bit number giving the number of cants),
- an initially distributed pay-to-script-hash item (with a script hash and a 64-bit number giving the number of cants),
- an intention (with a hash value),
- a published document (arbitrary, but checked to ensure correctness, intention and appropriate rights),
- a deed for an object (with the address of the owner and an optional 64-bit number indicating the cost of rights to use it as an object),
- a deed for a proposition (with the address of the owner and an optional 64-bit number indicating the cost of rights to use it as a known proposition),

- a right to use an object (with the address of the object and the number of times it may be used as an object),
- a right to use a proposition (with the address of the proposition and the number of times it may be used as a known proposition) or
- a bounty on a conjecture (with a 64-bit number giving the number of currency units).

An *obligation* ω is a triple $(\overline{\alpha}, m, n)$ where $\overline{\alpha}$ is a finite set of addresses and m and n are natural numbers. The number m is the number of signatures required from $\overline{\alpha}$ in order to spend an asset. The number n indicates the minimum block height at which an asset can be spent.

An *asset* is a triple (h, ω, u) where h is a unique identifier of the asset (in practice, a hash value), ω is an obligation, and u is a preasset.

The state of the system can be representated as a function taking addresses to lists of assets. Such a state can be represented using tries (similar to Merkle Patricia trees) as described in [44]. One can use the same representation to only keep up with the portions of the state relevant to the node in question.

8 Theorem Provers and Proof Checkers

In this section we discuss what kind of theorem prover or proof checker is appropriate for Qeditas. We begin by distinguishing between theorem provers and proof checkers. A *theorem prover* is a system in which one constructs proofs, often with varying degrees of automated assistance. (Given this role, such a system is sometimes called a *proof assistant.*) A *proof checker* is a system which takes a preexisting proof and simply checks that it is correct. For Qeditas the vital ingredient is a proof checker. Each time a document is published each node will need to use the proof checker to ensure its correctness. For the project to be successful, there should also be (at least one) theorem prover in which users can create documents and construct proofs.

For concrete examples, we can contrast two early groundbreaking systems: AUTOMATH [15, 17] and Mizar [41].

AUTOMATH was the first proof checker. The user would give definitions in full detail. Each of the definitions would have a declared type and the system would check that the given term has the given type. Some of the types would correspond to propositions and the terms would correspond to proofs, giving a first implementation of what is now known as the Curry-Howard-de Bruijn correspondence [15, 17, 25]. The AUTOMATH project is no longer active. A modern proof checker is Twelf [36].

Mizar was an early example of a theorem prover (dating back to the 1970s) and is still in use today.[3] Mizar is the only tool in wide use today based on

[3] The Mizar project was not known outside the Soviet block until the Iron Curtain fell. One can find excited discussions on the old QED mailing list prompted by the discovery that a small Polish group led by Andrzej Trybulec had already implemented a system doing many of the things under discussion.

set theory and provides support for some set theoretic notation similar to that used by traditional mathematicians. The proofs are given in a declarative and (relatively) readable style. The foundation of Mizar is essentially first-order Tarski-Grothendieck which is known to be consistent assuming the existence of certain large cardinals. Mizar has been used to build the Mizar Mathematical Library which comprises an impressive collection of mathematics.

What we require for Qeditas is a proof checker (like AUTOMATH or Twelf) with a reasonably small "trusted" code base, and a theorem prover which can be used to construct the proofs to be checked. As long as the underlying logic of a theorem prover is clearly defined, it should be easy enough to write a small proof checker independent of the theorem prover. If the theorem prover follows the Curry-Howard-de Bruijn correspondence, then it should be possible to "compile" proofs constructed with the system's assistance into proof terms to be independently checked.

Many popular theorem provers today follow a different scheme: the LCF approach. In this alternative, abstraction in the meta-language is used to guarantee correctness. While it seems to be feasible to produce different proof representations (e.g., proof terms) by translating from LCF style provers, it is not as simple as one might hope in practice [26, 27]. Theorem provers in the LCF style include Isabelle-HOL [33] and those in the HOL family [22] such as HOL-light [24].

A well-known theorem prover based on Curry-Howard-de Bruijn is Coq [29, 5, 11, 37] and Coq's ssreflect variant [21]. Coq is a widely used and well-developed system, even winning the 2013 ACM Software System award. Ssreflect was used to formally prove both the four color theorem [19] and later the Odd Order Theorem [20]. Coq has also been applied to theories directly related to cryptocurrencies [30, 43, 44, 1, 2]. There is clear evidence that it is possible to formalize serious mathematics in Coq and ssreflect. Coq supports the construction of proofs with some automation, but always compiles to a proof term checkable by a kernel. This would seem to make Coq (or ssreflect) the clear choice to use with Qeditas.

On the other hand, there are aspects of Coq which are experimental and this makes it somewhat dangerous to use in a context in which value depends on its stability. It is reasonable to be skeptical of the consistency of Coq's fairly sophisticated logic. For example, `proofmarket.org` placed a bounty on the proposition `False`. In other words, there was a bitcoin bounty placed on proving Coq inconsistent. While this should not have been provable in principle, in practice it was proven twice (for two different reasons). In addition, the foundational logic of Coq is not quite fixed and may change in subtle ways with each new version. Even with Coq's very attractive properties, it is also clear that it was not designed for such a purpose.

If there is no clear, safe way of building Qeditas on Coq or ssreflect, an alternative to using Coq would be to use Egal [7]. Like Coq, Egal constructs proof terms. The subset of the Egal code used for proof checking could be easily extracted to be part to be part of the Qeditas kernel. Egal also already has a method of obtaining addresses from syntactic terms as part of the implementation. (This fundamental feature was included in Egal to support the bitcoin theorem proving treasure hunt.) The foundation of Egal is a higher-order set theory (higher-order Tarksi-Grothendieck). Like Mizar's foundation, this is a theory which is known to be consistent assuming the existence of certain large cardinals. That is, if someone were to prove a contradiction in Egal, then either there is an implementation bug or there is a proof of a surprising mathematical result (that certain large cardinals cannot exist). In addition the code base of Egal is smaller than Coq. Egal's set theoretic foundation and notation may be easier for those with traditional mathematical training to use. Another advantage of using Egal is that a few people from the cryptocurrency community gained some experience using Egal in the bitcoin theorem proving treasure hunt at `mathgate.info` in 2014. It is likely that during this process a few bitcoin enthusiasts got over the critical hump required to learn to use such a prover.

The drawbacks of choosing to use Egal are obvious: Egal does not provide the rich environment of other provers, either in terms of the system or in terms of a community. Moreover, the proof tactics in Egal are of only modest power, leaving the user to do most of the work. Finally, the development of the system seems to have ended.

In addition to the systems above, there are two other systems that do not yet exist. A peer-to-peer system called ProofPeer [35] supporting formalization is in development. It is unclear at this early stage how similar ProofPeer and Qeditas will be. Another similar project is BitFuncTor [6]. BitFuncTor is intended to target functional programming instead of mathematics.

9 Conclusion

We have described Qeditas, a project to support distributed formalization of mathematics using block chain technology. The underlying currency will be similar to bitcoin in that there will be a similar 21 million unit cap. Two thirds of these units will be part of an initial distribution based on a snapshot of the Bitcoin block chain. Qeditas will support the publication of formal documents, the ownership of mathematical objects and propositions as intellectual property, the purchasing of rights to use such property and bounties on unproven conjectures. The hope is that this will be sufficient to motivate and reward participants to do the time consuming work of formalizing mathematics.

References

[1] andruiman. PoS forging algorithms: formal approach and multibranch forging. `github.com/ConsensusResearch/articles-papers/blob/master/multibranch/multibranch.pdf`, 2014.

[2] andruiman. PoS forging algorithms: multi-strategy forging and related security issues. `github.com/ConsensusResearch/articles-papers/blob/master/multistrategy/multistrategy.pdf`, 2014.

[3] Anonymous. The QED Manifesto. In Alan Bundy, editor, *CADE*, volume 814 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 1994.

[4] Adam Back. Hashcase – a denial of service countermeasure. `hashcash.org/papers/hashcash.pdf`, 2002.

[5] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[6] bitFuncTor. `bitfunctor.net`.

[7] Chad E. Brown. The Egal Manual. `mathgate.info/egalmanual.pdf`, September 2014.

[8] J. D. Bruce. The Mini-Blockchain Scheme, July 2014.

[9] Vitalik Buterin. Proof of Stake: How I Learned to Love Weak Subjectivity. `blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity`, November 2014.

[10] Alexander Chepurnoy. Inside a Proof-of-Stake Cryptocurrency Part 4: The Executable Forging Simulation. `chepurnoy.org/blog/2014/12/inside-a-proof-of-stake-cryptocurrency-part-4`, 2014.

[11] Adam Chlipala. *Certified Programming with Dependent Types.* MIT Press, 2011.

[12] Alonzo Church. A formulation of the simple theory of types. *J. Symb. Log*, 5(2):56–68, 1940.

[13] The Nxt community, July 2014.

[14] Mike Croteau and Emir Litranab. Proof of stake: Definite. an implementation of constant staking rewards to promote increased network activity, August 2014.

[15] N.G. de Bruijn. The mathematical language AUTOMATH, its usage, and some of its extensions. In M. Laudet, editor, *Proceedings of the Symposium on Automatic Demonstration*, pages 29–61, Versailles, France, December 1968. Springer-Verlag LNM 125.

[16] N.G. de Bruijn. Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indag. Math.*, 34(5):381–392, 1972.

[17] N.G. de Bruijn. A survey of the project AUTOMATH. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, pages 579–606. Academic Press, 1980.

[18] go1111111. Any coin that replaces Bitcoin will use the Bitcoin blockchain. `bitcointalk.org/index.php?topic=367885.0`, 2013.

[19] Georges Gonthier. The four colour theorem: Engineering of a formal proof. In Deepak Kapur, editor, *Computer Mathematics, 8th Asian Symposium, ASCM 2007, Singapore, December 15-17, 2007. Revised and Invited Papers*, volume 5081 of *Lecture Notes in Computer Science*, page 333. Springer, 2007.

[20] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A machine-checked proof of the odd order theorem. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, volume 7998 of *Lecture Notes in Computer Science*, pages 163–179. Springer, 2013.

[21] Georges Gonthier and Assia Mahboubi. An introduction to small scale reflection in Coq. *Journal of Formalized Reasoning*, 3(2):95–152, 2010.

[22] Michael J. C. Gordon. Introduction to the HOL system. In Myla Archer, Jeffrey J. Joyce, Karl N. Levitt, and Phillip J. Windley, editors, *Proceedings of the 1991 International Workshop on the HOL Theorem Proving System and its Applications, August 1991, Davis, California, USA*, pages 2–3. IEEE Computer Society, 1991.

[23] Thomas Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Hoai Thi Ta, Trung Nam Tran, Diep Thi Trieu, Josef Urban, Ky Khac Vu, and Roland Zumkeller. A formal proof of the Kepler conjecture, 2015.

[24] John Harrison. HOL Light: An overview. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics, TPHOLs 2009*, volume 5674 of *Lecture Notes in Computer Science*, pages 60–66, Munich, Germany, 2009. Springer-Verlag.

[25] William A. Howard. The formulas-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, 1980. Reprint of 1969 article.

[26] Cezary Kaliszyk and Alexander Krauss. Scalable LCF-style proof translation. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Proc. of the 4th International Conference on Interactive Theorem*

Proving (ITP'13), volume 7998 of *LNCS*, pages 51–66. Springer Verlag, 2013.

[27] Cezary Kaliszyk and Josef Urban. PRocH: Proof reconstruction for HOL Light. In Maria Paola Bonacina, editor, *Proc. of the 24th International Conference on Automated Deduction*, volume 7898 of *LNCS*, pages 267–274. Springer Verlag, 2013.

[28] Sunny King and Scott Nadal. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake, 2012.

[29] The Coq development team. *The Coq proof assistant reference manual.* LogiCal Project, 2012. Version 8.4.

[30] Andrew Miller, Michael Hicks, Jonathan Katz, and Elaine Shi. Authenticated data structures, generically. In *Proceedings of the ACM Conference on Principles of Programming Languages (POPL)*, January 2014.

[31] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno, and Jonathan Katz. Permacoin: Repurposing bitcoin work for data preservation. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 475–490. IEEE Computer Society, 2014.

[32] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.

[33] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[34] Ulf Norell. Dependently typed programming in agda. In *In Lecture Notes from the Summer School in Advanced Functional Programming*, 2008.

[35] Steven Obua, Jacques Fleuriot, Phil Scott, and David Aspinall. ProofPeer: Collaborative Theorem Proving. `proofpeer.net`, 2014.

[36] Frank Pfenning and Carsten Schürmann. System description: Twelf — a meta-logical framework for deductive systems. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, pages 202–206, Trento, Italy, July 1999. Springer-Verlag LNAI 1632.

[37] Benjamin C. Pierce, Chris Casinghino, Marco Gaboardi, Michael Greenberg, Cătălin Hriţcu, Vilhelm Sjoberg, and Brent Yorgey. *Software Foundations.* Electronic textbook, 2014.

[38] Andrew Poelstra. Distributed consensus from proof of stake is impossible, May 2014. download.wpsoftware.net/bitcoin/pos.pdf.

[39] Peter R. Spin-offs: bootstrap your alt-coin with a bitcoin-blockchain-based initial coin distribution. `bitcointalk.org/index.php?topic=563972.0`, 2014.

[40] Consensus Research. Github repo. github.com/ConsensusResearch, 2014.

[41] Piotr Rudnicki and Andrzej Trybulec. Mathematical knowledge management in mizar. In *Proc. of MKM 2001*, 2001.

[42] Kazuhiko Sakaguchi. proofmarket.org.

[43] Bill White. Formal Idealizations of Cryptographic Hashing. github.com/billlwhite/cryptohash, 2015.

[44] Bill White. A Theory for Lightweight Cryptocurrency Ledgers. github.com/billlwhite/ledgertheory, 2015.

[45] Freek Wiedijk. Estimating the Cost of a Standard Library for a Mathematical Proof Checker. www.cs.ru.nl/~freek/notes/mathstdlib2.pdf.

[46] Gavin Wood. Ethereum: a secure decentralised generalised transaction ledger. gavwood.com/Paper.pdf. Final draft – under review.