Blockchain Cohomology[⋆]

Wyatt Meldman-Floch[1] [0000-0001-6582-5925]

Constellation Labs, San Francisco CA 94108, USA

Abstract. We follow existing distributed systems frameworks employing methods from algebraic topology to formally define primitives of blockchain technology. We define the notion of cross chain liquidity, sharding and probability spaces between and within blockchain protocols. We incorporate recent advancements in synthetic homology to show that this topological framework can be implemented within a type system. We use recursion schemes to define kernels admitting smooth manifolds across protocol complexes, leading to the formal definition a Poincare protocol.

Keywords: Distributed computing · Homology · Algebraic topology

1 Introduction and Literature Overview

The following shows a correspondence between analytical models for verifying distributed systems and functional programming for practical software engineering of blockchain technology. Applications are provided in terms of emerging blockchain technology with specific examples given from the architecture of the Constellation protocol, which was designed using the following models. The goal of this work is to provide tools for verifying blockchain protocols with techniques from functional programming that have been successful in improving statistical modeling and distributed system design.

The goal of constructing these algebraic models is to provide a correspondence to analytical models of distributed systems and functional programming for practical software engineering.

2 Consensus Protocols

Recent advancements in distributed computing adopt methods from algebraic topology for formally defining protocols [1, 2]. We use these methods to model blockchain protocols as well as an internet of blockchains. We first define an execution space as a topological space equipped with a discrete product topology [3]. Defining a distributed process in terms of topology only requires us to care about the structure of the set of possible schedules of a distributed system [4]. We adopt Nowak's algebraic definition of an execution space in terms of the

⋆ Supported by Constellation Labs

homology of protocol complexes [2]. We define a protocol complex $S_k : P_k\Delta^q$ as the q-dimensional standard simplex

$$\Delta^q = \{x \in \mathbb{R} | \Sigma x_j = 1, x_j \geq \forall j\} \tag{1}$$

at morphism k described by the following vertex set

$$S_k = \{v_{i,0} \dots v_{i,q}\} \tag{2}$$

where $P \subset S$ is the set of all admissible configurations and S is the set of all possible configurations.

Define a consensus protocol $P_*^\sigma(S) : \{S_k, \partial_k\}$ as the singular homology of a simplicial chain complex, carried by a group morphism implementing distributed consensus. Let S_k be a simplex configuration at step k and ∂_k be the differential of a distributed consensus morphism:

$$P_*^\sigma(S) : 0 \leftarrow \dots P^\sigma(S_{k-1}) \xleftarrow{\partial_{k-1}} P^\sigma(S_k) \xleftarrow{\partial_k} P^\sigma(S_{k+1}) \dots \tag{3}$$

where $P_k = ker\partial_k / im\partial_{k+1}$ and is also an abelian group. Thus, $P_* = (P_k) \; | k \in \mathbb{Z}$ is a graded abelian group which is referred to as the homology of a protocol complex S. We abuse our notation of P but rectify by noting that an admissible state k is required for anther step $k+1$, thus we define P as the functor carrying our consensus operator defined below.

Define a consensus operator σ as the group morphism on the singular q-simplex $\sigma : \Delta^q \to S$

$$\sigma_k : S_{k-1} \times P_k \to S_k \tag{4}$$

which are continuous on discrete topologies such as Δ^q [1]. Define homology between configurations as a measure of divergence given by the differential

$$\partial_k(\sigma) = \sum_{k=0}^{q} (-i)^{i-1} (\sigma \circ \delta_q^i) \tag{5}$$

for continuous functions $\delta_q^i : \Delta^{q-1} \to \Delta^q | 1 \leq i \leq q+1$ where

$$\delta_q^i(x_1, \dots, x_q) = (x_1, \dots x_{i-1}, 0, x_i, x_{i+1}, \dots, x_{q-1}, \dots, x_q) \tag{6}$$

As the graded abelian group of our protocol complex is the simplicial singular homology group and σ is our homology preserving map, it is trivial to note that homology holds $\forall k \in \mathbb{Z}$, i.e.

$$\partial_k \circ \partial_{k+1} = 0 \tag{7}$$

As a corollary of the fact that the geometric realization of a simplicial complex is dually a topological space, due to the vanishing homology up to k, $P_k\Delta^q$ is k-acyclic, or that there is a consistently forward moving "arrow of time".

3 Protocol Topologies

It's possible to layer protocol complexes defined as above with guarantees about consistency as long as continuity is preserved. Our definition of homology above is verification criteria for the ability to exchange configuration states between protocol complexes.

Specifically, a valid 'layering' as the existence of a functoral vertex map between singular homologies (defined equivalently here as the disjoint subset of protocol complexes) $l : \bigcup_k P_\pi \rightarrow \bigcup_k P_{\pi+1}$.

Making use of homotopy type theory allows us to focus on structure by treating topological characteristics called homotopy groups as primitives. Noting that simplicial complexes together with simplicial vertex maps form a category, if we redefine our k-acyclic distributed consensus protocol σ categorically as the functoral carrier Σ_* we can form a chain complex that adheres to the homology theory of homotopy types [5].

Let us define a 'layering' as a Protocol Topology $T_P^\Sigma : \Sigma_* P_\pi$, the singular homology of a chain complex of protocol complexes carried by a homotopy preserving functor Σ_*. The protocol topology is given by the following chain complex

$$T_P^\Sigma : 0 \leftarrow \Sigma_* P_\pi \overset{\partial}{\leftarrow} \Sigma P_0 \overset{\partial}{\leftarrow} \ldots \Sigma P_i \; | i \leq \pi \in \mathbb{Z} \tag{8}$$

where $\Sigma_\pi : ker\partial_k^\pi / im\partial_{k+1}^\pi \rightarrow \partial_k^{\pi+1} / im\partial_{k+1}^{\pi+1}$

For protocol complex morphisms $\Sigma_\pi, \Sigma_{\pi+1}$ chain homotopy from Σ_π to $\Sigma_{\pi+1}$ is a homotopy preserving graded abelian group morphism $l : P_\pi \rightarrow P_{\pi+1}$ yielding a vanishing homology, i.e.

$$\begin{aligned} \Sigma_\pi - \Sigma_{\pi+1} &= \partial^\pi \circ l + l \circ \partial^{\pi+1} \\ &= \partial^\pi \circ \partial^{\pi+1} = 0 \end{aligned} \tag{9}$$

Noting that these conditions are met by the definitions of an acyclic carrier as above, it follows that a protocol topology as defined above is π-acyclic.

3.1 Applications: Scale out Networking

Traditional blockchains have come up short in their ability to support real world use cases. Scalability and integration for end to end secure notarization are still open problems, lacking general solutions. In terms of scalability, sharding or partitioned approaches have taken form in improvements to Bitcoin with Lightning [17] as well as base layer protocols like Zilliqa [18]. In these approaches, relays or subnets are deployed to ferry larger amounts of transactions into a fixed mempool size. The key to their success is adding another layer to the network topology to buffer and compress data. If we consider either base layer protocol as a protocol complex, then a partitioned or sharding mechanism is described by our model of protocol topology. A type hierarchy is enough to verify a protocols equivalence to a Protocol Topology due to covariance, which is a valid null differential as detailed by R. Grahm above. Specifically, given a base layer protocol Σ_π with block type π, a type preserving operation $\Sigma_{\pi+1}$ such as a side chain mechanism or sharding scheme then if there exists covariance between their data types $\pi+1 — \partial^\pi \circ \partial^{\pi+1} = 0$ network queries can be implemented with guarantees around correct access governed at the type level, as functoral operators such as .map() and .reduce(). Specifically, consider a protocol topology of an arbitrary one layer side channel mechanism:

$$T_P^\Sigma : 0 \leftarrow \Sigma_* P_\pi \xleftarrow{\partial} \Sigma P_0 \xleftarrow{\partial} \Sigma P_2 \tag{10}$$

A program implementing the above protocol is verifiable by ensuring functoral covariant state transitions which can be enforced by static analysis [10].

4 Blockchain Cohomology

Distributed architectures designed topologically can be verified at the type level. In order to model a differentiable state, we need to design our topologies such that they are mathematically tractable for modeling specific operations. In principle, Abstract Differential Geometry (ADT) admits any topological space as a base space on which to 'solder sheaves' for carrying out differential geometry [6]. We introduce methods from Abstract Differential Geometry, namely finitary cech-deRham cohomology in order to define an orientable manifold from our definition of protocol topology.

4.1 Block Sheaves

First we need to introduce the dual of homology as described above, namely cohomology. In describing our protocol complex it only makes sense to have an arrow moving 'forward in time' as consensus itself is acyclic, with each iteration pointing 'backwards in time' to its previous state. In this sense our evolution was the compounding dimensionality of the space of all configurations, as implied by the discrete product topology of a protocol complex. In defining an orientable manifold, we need to move 'backwards' through our space, i.e. from higher to lower dimension. This is shown as the differential on an arrow going right instead of left.

By constructing the protocol topology within a monoidal category, A. Malios et al. showed that the singular cohomology of a protocol topology is equivalent to an A-module of Z+-graded discrete differential forms [6], otherwise known as discrete differential manifolds. This forms an execution tree, a sequence of configurations a sequence of configurations in a poset (directed acyclic graph) topology. A decision tree can be assigned to any set of executions that captures the decision of choosing a successive configuration. A blockchain can be defined as an extension of an execution tree, where each block is formulated as a sheaf with a well defined tensor operation and each successive block verified by a decision tree. We define a sheaf ϵ as the 'enrichment' of any cochain $\mathbb{A}$-complex of positive degree/grade, corresponding to the $\mathbb{A}$-resolution of an abstract $\mathbb{A}$-module

$$S^* : 0 \to \epsilon \to S^0 \xrightarrow{d^0} S^1 \xrightarrow{d^1} \dots \tag{11}$$

and homomorphism given by Cartan-Kahler-type of nilpotent differential operator d. We will make use of the fact that an $\mathbb{A}$-module sheaf ϵ on any arbitrary topological space (shown above with an arbitrary simplicial cochain-complex) admits an injective resolution.

Blockchains are naturally equipped with a sheaf, known as a block hash. Every abelian unital ring admits a derivation map [7], allowing us to 'unpack' data within a block recursively under the product operation. By noting the equivalence of Sorkin's fintoposets to simplicial complexes, A. Mallios et al. showed that the Gelfand duality implies that a manifold can be constructed from simplicial complexes [6]. Thus if we reformulate our definition of a consensus protocol above as a sheaf with semigroup operations carried by right derived functors with monadic bind, we can form a manifold.

For a fintoposet (the topological equivalent of a directed acyclyc graph), it's incidence algebra can be broken down into a direct sum of vector subspaces

$$\Omega(P) = \bigoplus_{i \in \mathbb{Z}_+} \Omega^i = \Omega^0 \oplus \Omega^0 \dots := A \oplus R \tag{12}$$

where $\Omega(P)$s are $\mathbb{Z}_+$ graded linear spaces, A is a commutative sub algebra of Ω and $R := \bigoplus_{i \geq 1} \Omega^i$ is a linear (ringed) subspace. It is trivial to notice that $\Omega(P)$ is an A-module of a Z+-graded discrete differential form.

A manifold can be constructed by organizing the incidence algebras of our protocol complexes into algebra sheaves. The n-th (singular) cohomolgy group $H_n(X, \epsilon)$ of an A-module sheaf $\epsilon(X)$ over topological space X, can be described by global sections $\Gamma_X(\epsilon) \equiv \Gamma(X, \epsilon)$

$$H_n(X, \epsilon) := R^n(\Gamma(C, \epsilon) := H^n[\Gamma(C, S^*)] := ker\Gamma_X(d^n)/im\Gamma_X(d^{n-1}) \tag{13}$$

where $R^n\Gamma$ is the right derived functor of the global section functor $\Gamma_x(.) \equiv \Gamma(X, .)$. Note that R^n is equivalent to the i^{th} linear ringed subspace above. These dual definitions of gamma correspond to out definitions of σ and Σ_* with respect to our functoral vertex map l in our definition of a protocol topology.

The sheaf cohomology of a topological space is the cohomology of any Γ_X-acyclic resolution of ϵ [11]. The corresponding abstract $\mathbb{A}$-complex S^* can be directly translated by the functor Γ_x to the 'global section $\mathbb{A}$-complex' $\Gamma_X(S^*)$

$$\Gamma_X(S^*) : 0 \to \Gamma_X(\epsilon) \xrightarrow{d^0} \Gamma_X(S^0) \xrightarrow{d^1} \dots \tag{14}$$

which is the abstract de Rham complex of a discrete manifold X. The action of d is to effect transitions between the linear subspaces Ω_i of $\Omega(P)$ in (11), as follows: d: $\Omega_i \to \Omega_{i+1}$.

The finitary de Rham theorem defines a finitary equivalent of the typical c^∞ smooth manifold. Noting $\varGamma_m^{P_m}$ is fine by construction, Mallios et al. show that finsheaf-cohomology differential tetrads

$$\tau := (P_m, \varOmega_M, d, \varOmega_{deR}^M) \tag{15}$$

is equivalent to the c^∞-smooth Cech-de Rham complex. In our definition of τ, $\varOmega_M$ is the categorically dual finsheaf (finitary sheaf) of Sorkin's fintoposets P_m, d is effectively an exterior product, and $\varOmega_{deR}^M$ is the abstract de Rahm complex.

4.2 Protocol Manifold

We've shown how to create a manifold from the cohomology of a discrete topological space. We now show how to construct a manifold from of a protocol topology. Define a cochain-complex within the cohomology theory of homotopy types under the cup product. Making note of the existence of a tensor product in the cohomology theory of homotopy types by E. Cavallo [9] define the protocol manifold as

$$\varGamma_\varSigma^\epsilon = \bigoplus_{0 \leq i \leq \pi} \varSigma_* \epsilon_i \tag{16}$$

4.3 Applications: Data Locality and MapReduce

The design of tools in the data engineering and data science space has been often employ principles from functional programming and type theory due to the benefits they have at verifying code at compile time. This is largely a source of the origins of scalable data processing tools for analysis and modeling like Hadoop and Spark. Distributed data stores and microservice architectures employ monadic design patterns for improvements on concurrency, static type checking, testing, design patterns, cluster management, training models etc. One benefit is the development conveniant API's with Map/Reduce operations simplifying integration by abstracting low level data locality management [12], a key example would be Apache Sparks RDD [14]. Monadic execution models allow complex behavior to be implemented and governed declaratively, which allows distributed data stores to be constructed with high level API's. The models above correspond to network topology, configuration state and dynamic rebalancing respectively. As such, if we follow their construction at the type level in architecture and code design we can develop increasingly complex distributed systems with greater guarantees.

An open problem concerning blockchains integrating into open networks end to end security for smart contracts. Advancements have been made in end to end security has been made by DAG technology, and state channel mechanisms through direct deployment of nodes on physical hardware, improving visibility of chain of custody pipelines. These types of systems typically rely sessionization, typically a streaming topology mixed with batch processing to merge heterogeneous event data by keying by time or application specific criteria. The

Protocol Manifold describes a distributed data store of topological data formed of state data across blocks and architecturally equivalent to a state matrix for a sessionization or streaming joins. Indexing all event data within a given state transition window or timestep is fundamental for basic analytics and a requirement for developing online and downstream models. Consider an enrichment of a type hierarchy $\epsilon \ldots \epsilon_n$, if all ϵ commute it implies that there exists a valid state transition. Such an ϵ implemented as a type with a product operation [15] would allow us to define state data as an 'unpacking' operation across blocks under the product operation, allowing for state channels to be defined by 'chaining' validation criteria for each type within a block and verifying at the type level. The end result is a decentralized orchestration of validation and composition of state channels required for interoperable protocols (cross chain liquidity) and secure state data for consumers.

5 Typesafe Poincare Duality

Up until now we have not explicitly defined functoral group homomorphisms that can construct the complexes described above. We show that the dual nature of the hylomorphic and metamorphic recursion schemes maintain vanishing differentials and thus poincare duality for all π.

If we define a catamorphism and anamorphism with the same f-algebra and f-coalgebra, we can show by construction that the resulting co/chain-complexes are valid definitions of protocol topologies/manifolds and that poincarre duality of the protocol manifold is maintained up to π isomorphism. We define in terms of Σ and ϵ, noting that our functor Σ is a valid f-algebra and sheaf ϵ a co-algebra.

Let us define a hylomorphism

$$\epsilon \leftarrow P \times \Sigma : \Omega^T(\epsilon, P) \tag{17}$$

and metamorphism

$$\Omega_\Gamma(P, \epsilon) : \Gamma_\Sigma \times \epsilon \rightarrow P \tag{18}$$

we formally verify by the construction of the following geometric cw-complex

$$\Omega^T_\Gamma : 0 \overset{\partial}{\leftrightarrow} \Omega^{T^*}_{\Gamma^*}(\epsilon) \overset{\partial}{\leftrightarrow} \Omega^T_\Gamma(\epsilon(P_0)) \ldots \Omega^T_\Gamma(\epsilon(P_\pi)) \tag{19}$$

that T and Γ form a poincare complex, clearly satisfying the poincare duality as ∂ vanishes in our construction of T and Γ^ϵ_Σ. The fundamental class of our corresponding space is $\Omega^{T^*}_{\Gamma^*}$ which carries the type signatures of our hylo and metamorphisms. Formally define Ω^T_Γ as a Poincare protocol.

5.1 Applications: Elastic Infrastructure

A poincare protocol was constructed as a model for designing distributed systems with complex dynamics that can be verified at the type level. Microservice architectures and distributed compute clusters have countless API's with performance considerations. Despite the inherent complexity in the components themselves, constructing a distributed dataset across such a network is tractable when implemented as a poincare protocol, as demonstrated by the Constellation Protocol.

An example of a blockchain system corresponding to a Poincare complex is the Constellation protocol, as is its origin. It's use in defining a feature space representing the protocol's state at the type level is explored below, as well as examples of how these algebraic models correspond to new developments in blockchain technology and can be used in protocol design.

6 Conclusion

References

1. Nowak T., Schmid U.: Topology in Distributed Computing, Master's Thesis, Vienna University of Technology, (2010)
2. Herlihy M., Rajsbaum S.: Algebraic topology and distributed computing a primer. In: van Leeuwen J. (eds) Computer Science Today. Lecture Notes in Computer Science, vol 1000. Springer, Berlin, Heidelberg. (1995) https://doi.org/10.1007/BFb0015245
3. Alpern, B., Schneider F.: Defining liveness. Information Processing Letters 21, 4 (October 1985), 181–185. Cornell University, (1985)
4. Saks, M., Zaharoglou, F.: Wait-free k-set agreement is impossible: the topology of public knowledge. SIAM J. Comput. 29(5), 1449–1483 (2000)
5. Grahm R.: Synthetic Homology in Homotopy Type Theory Robert Graham. arXiv:1706.01540 (2017)
6. Mallios A., Raptis I.: Finitary Cech-de Rham Cohomology: much ado without smoothness. Int.J.Theor.Phys. 41 1857-1902 (2002)
7. Mallios, A.: Geometry of Vector Sheaves: An Axiomatic Approach to Differential Geometry, vols. 1-2, Kluwer Academic Publishers, Dordrecht (1998)
8. Mallios, A.: On an Axiomatic Treatment of Differential Geometry via Vector Sheaves. Applications, Mathematica Japonica(International Plaza), 48, 93. (1988)
9. Cavallo, E.: Synthetic cohomology in Homotopy Type Theory. Master's thesis, Carnegie-Mellon University (2015)
10. Nielson F. Nielson H. R.: Type and Effect Systems. In: Olderog ER., Steffen B. (eds) Correct System Design. Lecture Notes in Computer Science, vol 1710. Springer, Berlin, Heidelberg 114-136, (1999)
11. Gallier J., Quaintance J.: A Gentle Introduction to Homology, Cohomology, and Sheaf Cohomology. Preprint (2016)
12. Wu D., Sakr S., Zhu L.: Big Data Programming Models https://doi.org/10.1007/978-3-319-49340-4_2
13. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
14. Apache Spark RDD documentation, https://spark.apache.org/docs/latest/rdd-programming-guide.html#resilient-distributed-datasets-rdds. Last accessed 15 Oct 2019
15. Product operator from functional programming library Cats, http://eed3si9n.com/herding-cats/Cartesian.html. Last accessed 15 Oct 2019

16. Constellation Protocol repository, https://github.com/Constellation-Labs/constellation/blob/7aeaa786d42e0194e31cd8fd0c6b99462cb63f33/src/main/scala/org/constellation/Cell.scala. Last accessed 15 Oct 2019
17. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. Joseph Poon, Thaddeus Dryja https://lightning.network/lightning-network-paper.pdf
18. The ZILLIQA Technical Whitepaper, The Zilliqa Team. https://whitepaper.io/document/16/zilliqa-whitepaper