Blockchain Cohomology

Wyatt Meldman-Floch

Constellation Labs
March 16, 2019

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.

1 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 homology of protocol complexes [5]. We define a protocol complex $S_k : P_k\Delta^q$ as the q-dimensional standard simplex

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

[1]T. Nowak, "Topology in Distributed Computing" University of Vienna, https://pdfs.semanticscholar.org/fd74/ed78ccffc6faa708b933fb0bdf7ceb62896d.pdf

[2]M. Herlihy et al. http://www.lix.polytechnique.fr/~goubault/papers/sv.pdf

[3]Alpern, Bowen and Fred B. Schneider. Defining liveness. Technical Report TR85-650, Cornell University, 1985. https://ecommons.cornell.edu/bitstream/handle/1813/6495/85-650.pdf?sequence=1&isAllowed=y

[4]Saks, Michael and Fotios Zaharoglou 2000, impossibility of the wait free k-set agreement http://courses.csail.mit.edu/6.852/08/papers/SaksZaharoglu.pdf

[5]M. Herlihy et al.

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.

We 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[6] such as Δ^q. Define homology between configurations as a measure of divergence given by the differential

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

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

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

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 cohomology up to k, we note that $P_k \Delta^q$ is k-acyclic[7].

[6] Nowak, Lemma 4.5

[7] Nowak, Definition 5.4

2 Protocol Topologies

It's possible that we could 'mix' protocol complexes defined as above. We employ our notion of cohomology to define a 'liquidity' or the ability to exchange configuration states between protocol complexes. We leave applications of this as an exercise for the reader.

We define liquidity as the existence of a functoral vertex map between singular homologies (defined equivalently here as the disjoint subset of protocol complexes) $l : \bigcup_k P_\pi \to \bigcup_k P_{\pi+1}$.

Making use of homotopy type theory allows us to focus on structure by treating topological characteristics called homotopy groups as primitives. 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[8]

Simplicial complexes together with simplicial vertex maps form a category. Let us define a protocol topology $T_P^\Sigma : \Sigma_* P_\pi$ as 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

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

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

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 $l : P_\pi \to P_{\pi+1}$ 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 [9], it follows that a protocol topology as defined above is π-acyclic.

[8] R. Grahm "Synthetic Homology in Homotopy Type Theory" `https://arxiv.org/pdf/1706.01540.pdf`

[9] Nowak, Theorem 5.1

3 Block Sheaves

Designing distributed architectures with topology gives us a lot of power, but in order to use it we need to design our topologies such that they are mathematically tractable for solving a specific problem. In principle, Abstract Differential Geomoetry (ADT) admits any topological space as a base space on which to 'solder sheaves' for carrying out differential geometry[10]. 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.

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, the singular cohomology of a protocol topology is equivalent to an A-module of Z+-graded discrete differential forms. One can, in a natural way, assign a decision tree to any set of executions that captures the decision of choosing a successor[11]. 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. 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

$$S^* : 0 \to \epsilon \to S^0 \xrightarrow{d^0} S^1 \xrightarrow{d^1} \dots \tag{10}$$

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 per (10).

Blockchains are naturally equipped with a sheaf, that of the block. This would allow us to 'unpack' data within a block recursively under the product operation. Every abelian unital ring admits a derivation map [12], thus if we reformulate our definition of a consensus protocol above as a sheaf with semigroup

[10] A. Mallios et al. "Finitary Cech-de Rham Cohomology: much ado without C^∞-smoothness"

[11] Nowak, Section 4.1.2

[12] Mallios, A., Geometry of Vector Sheaves: An Axiomatic Approach to Differential Geometry, vols. 1-2, Kluwer Academic Publishers, Dordrecht (1998)

operations carried by right derived functors with monadic bind, we can form a manifold.

By noting the equivalence of Sorkin's fintoposets[13] as simplicial complexes, Mallios et al. showed that the Gelfand duality[14] implies that a manifold can be constructed out of the incidence Rota algebra of a simplex's corresponding fintoposet [15]. 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

$$\Omega(P) = \bigoplus_{i \in \mathbb{Z}_+} \Omega^i = \Omega^0 \oplus \Omega^0 \cdots := A \oplus R \tag{11}$$

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 $\Gamma_X(\epsilon) \equiv \Gamma(X, \epsilon)$

$$H_n(X, \epsilon) := R^n(\Gamma(C, \epsilon) := H^n[\Gamma(C, S^*)] := ker\Gamma_X(d^n)/im\Gamma_X(d^{n-1}) \tag{12}$$

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[16]. The corresponding abstract $\mathbb{A}$-complex S^* can be directly translated by the functor Γ_x to the 'global section $\mathbb{A}$-complex' $\Gamma_X(S^*)$

$$\Gamma_X(S^*) : 0 \rightarrow \Gamma_X(\epsilon) \xrightarrow{d^0} \Gamma_X(S^0) \xrightarrow{d^1} \ldots \tag{13}$$

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: $\Omega_i \rightarrow \Omega_{i+1}$.

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

$$\tau := (P_m, \Omega_M, d, \Omega_{deR}^M) \tag{14}$$

[13] Section 3.2, Mallios et al.

[14] Section 3.3, Mallios et al.

[15] eq 9, Mallios et al.

[16] Mallios, A., "On an Axiomatic Treatment of Differential Geometry via Vector Sheaves." Applications, Mathematica Japonica

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

4 Blockchain Cohomology

We've shown how to create a manifold from the cohomology of a discrete topological space. We can define a synthetic manifold out of a protocol topology[17]. 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 [18] we define the protocol manifold as

$$\Gamma^\epsilon_\Sigma = \bigoplus_{0 \le i \le \pi} \Sigma_* \epsilon_i \tag{15}$$

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

$$\epsilon \leftarrow P \times \Sigma : \Omega^T(\epsilon, P) \tag{16}$$

and metamorphism

$$\Omega_\Gamma(P, \epsilon) : \Gamma_\Sigma \times \epsilon \to P \tag{17}$$

[17] J. Gallier et al. Definition 3.3: "A Gentle Introduction to Homology, Cohomology, and Sheaf Cohomology" https://www.seas.upenn.edu/ jean/sheaves-cohomology.pdf

[18] E. Cavallo, "Synthetic Cohomology in Homotopy Type Theory", http://www.cs.cmu.edu/~ecavallo/works/thesis15.pdf

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

$$\Omega_{\Gamma}^{T} : 0 \overset{\partial}{\longleftrightarrow} \Omega_{\Gamma^*}^{T^*}(\epsilon) \overset{\partial}{\longleftrightarrow} \Omega_{\Gamma}^{T}(\epsilon(P_0)) \ldots \Omega_{\Gamma}^{T}(\epsilon(P_\pi)) \tag{18}$$

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

6 Applications

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. The design of tools in the data engineering and data science space has been influenced by the correspondence of functional programing and category theory. 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, a key example would be Sparks RDD and extending interfaces. Monadic execution models allow complex behavior to be implemented and governed declaratively, which allows distributed data stores to be constructed as 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 more complex distributed systems with greater guarantees at the granularity of differential models. An example of a distributed system governed by differential model and 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.

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 and Ethereum with Raiden as well as newer protocols like Zilliqa. 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 data model following a type hierarchy satisfies the null differential requirement because of covariance [19] each layer in the network topology is governed by the configuration complex of each rank in the cw-complex. 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 $\pi + 1 \mid \partial^\pi \circ \partial^{\pi+1} = 0$ network queries can be implemented query with guarantees around correct access governed at the type level, as functoral operators such as .map() and .reduce().

An open problem concerning blockchains integrating into open networks is the security of 'relays',[20] or mechanisms which interact with a smart contract. Advancements have been made in end to end security has been made by DAG technology, and state channels by direct deployment of nodes on physical hardware, giving direct visibility on the chain of custody data. Advancements in application integration around state channels correspond to our construction of a Protocol Manifold. Consider an enrichment of a type hierarchy $\epsilon \ldots \epsilon_n$, if all ϵ commute it implies that there exists a state transition with $d = 0(12)$. Such an ϵ implemented as a typeclass with a product operation[21] would allow us to 'unpack' data within a block recursively under the product operation[22], 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 and cross chain liquidity.

A blockchain protocol that can natively integrate with data engineering tools would require the guarantees given by a Protocol Topology and Protocol Manifold, or equivalently, implement a Poincare Protocol. 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.

6.1 Constellation

The Constellation Protocol was designed specifically to provide scalable application integration in an open network setting with reputation based consensus. The problem can be phrased as follows: given a collection nodes in a configura-

[19]

[20]

[21] See product operator in validation pipeline, link to examples here `http://eed3si9n.com/herding-cats/Cartesian.html`

[22] see figure on Constellation's validation pipeline

tion space, and a topologically invariant distance metric (ala XOR) does there exist a protocol that forms a vector space of verifiable state transitions of rank up to some π? The Poincare protocol is a solution by construction and the space is formed by the outer product of wedge space of the sheaves in the protocol manifold, $\epsilon \wedge \cdots \wedge \epsilon_\pi$.

With this model, application integration is implemented by deploying data models and application specific validation criteria which are chained on to a monadic validation pipeline (EnrichedFuture) via a product operator

```
package org.constellation.util
import cats.data.{Validated, ValidatedNel}

import scala.concurrent.{ExecutionContext, Future}

object Validation {
 implicit class EnrichedFuture[A](future: Future[A]) {
  def toValidatedNel(implicit ec: ExecutionContext):
      Future[ValidatedNel[Throwable, A]] = {
   future.map(Validated.valid).recover {
    case e =>
     Validated.invalidNel(e)
   }
  }
 }
```

```
package org.constellation.util
import cats.data.{Validated, ValidatedNel}

import scala.concurrent.{ExecutionContext, Future}

sealed trait CheckpointBlockValidatorNel {
 type ValidationResult[A] =
     ValidatedNel[CheckpointBlockValidation, A]
 ...

  def validateCheckpointBlock(
  cb: CheckpointBlock
 )(implicit dao: DAO): ValidationResult[CheckpointBlock] = {
  val preTreeResult =
   validateEmptySignatures(cb.signatures)
    .product(validateSignatures(cb.signatures, cb.baseHash))
    .product(validateTransactions(cb.transactions))
    .product(validateDuplicatedTransactions(cb.transactions))
    .product(validateSourceAddressBalances(cb.transactions))

  val postTreeIgnoreEmptySnapshot =
   if (dao.threadSafeTipService.lastSnapshotHeight == 0)
```

```
        preTreeResult
      else preTreeResult.product(validateCheckpointBlockTree(cb))

    postTreeIgnoreEmptySnapshot.map(_ => cb)
  }
}
object CheckpointBlockValidatorNel extends
    CheckpointBlockValidatorNel
```

6.2 Map Reduce Interface

Algebraic models for APIs are not a new thing [23]. By defining a state channel interms of the algebra of it's api and the coalgebra definining how to mix it with other API's and itself, we can construct nonlinear callback trees that define the network topology of a state channel. This is a Gather Apply Scatter approach to stream processing [24] which merges batch API calls streams with a map/reduce interface (map can be implemented with liftF, reduce with ioF) and static analysis .

```
trait Cell[A] extends EnrichedFuture[A]

/**
  * Functor of cellular execution context, forms cellular
      complex with poset topology.
  * The configuration space, i.e. poincare complex is carried
      functorally by the cellular complex.
  */
object Cell {
  implicit val cellFunctor: Functor[Cell] {def map[A, B](fa:
      Cell[A])(f: A => B): Cell[B]} = new Functor[Cell] {
    override def map[A, B](fa: Cell[A])(f: A => B): Cell[B] =
        fa match {
      case SingularHomology(sheaf) => SingularHomology(sheaf)
      case Homology(a, next) => Homology(a, f(next))
    }
  }

  /**
    * Buildup
    */
  val coAlgebra: Sheaf => Cell[Sheaf] = {
    case sheaf: Sheaf =>
      SingularHomology(sheaf)
  }
```

[23] runar in fp for scala, algebra of an API
[24]

```
  /**
   * Teardown
   */
  val algebra: Cell[Sheaf] => Sheaf = {
    case SingularHomology(sheaf) => sheaf
    case hom@Homology(kernal, image) => kernal.combine(merge)
  }

  def hylo[F[_] : Functor, A, B](f: F[B] => B)(g: A => F[A]): A
      => B = a => f(g(a) map hylo(f)(g))

  /**
   * basically just a lift, see Streaming:
       https://patternsinfp.wordpress.com/2017/10/04/metamorphisms/
   */
  def meta[A, B](g: Cell[B] => B)(f: A => Cell[A]): Cell[A] =>
      Cell[B] = a => a map hylo(g)(f)

  def ioF(sheaf: Sheaf): Sheaf =
      hylo(algebra)(coAlgebra).apply(sheaf)

  def liftF(cell: Cell[Sheaf]): Cell[Sheaf] =
      meta(algebra)(coAlgebra)(cell)
}

case class SingularHomology[A](sheaf: Sheaf) extends Cell[A]

case class Homology[A](sheaf: Sheaf, bundle: A) extends Cell[A]

}

case class SingularHomology[A](sheaf: Sheaf) extends Cell[A]

case class Homology[A](sheaf: Sheaf, bundle: A) extends Cell[A]
```

Thus composite state channels can be composed and verified via declarative validation and orchestrated by the internals of a distributed data store. This allows for higher level API's that follow the MapReduce model for distributed data storage, which is an industry standard for large scale data processing.

7 Remarks

It's worth noting that the isomorphism between simplectic and poset topology shown by Sorkin's fintoposets implies that when applied to blockchains, the existence of cycles in a cohomological or homological cw-complex imples the existence of forks. In future work we will show that finite autamata with monoidal state transitions (semiautomation) admit a Poincare protocol with enrichment isomorphic to the semigroup operation of state transitions. Extending this, we'll make use of a Poincare protocol's manifold to define monoidal state transitions that prevent divergences, and transitively forks.