A Formal Model of the Bitcoin Transaction Graph and Block Chain*

Cliff Joslyn, Chase Dowling, Sean Kreyling, Curt West

March, 2015

Abstract

We develop an abstract, formal model of the Bitcoin transaction graph as a vertex-weighted directed, additive hypergraph with frictional loss; and the block chain as a hypervertex partition thereon.

Contents

1 Introduction

We introduce a formal model of Bitcoin operations [8]. We begin with a pseudo-EBNF (extended Baukus-Naur form) specification, with mathematical notation, defining a weighted directed hypergraph model. We illustrate with an example, and go on to discuss implications and observations, including the role of user identification, fungibility of Bitcoin, and statistical network flow models.

*PNNL Technical Report PNNL-24166.

2 Definitions

Address $a := $ **Amount:**`int` Perhaps the most fundamental aspect of Bitcoin is an **address** which is simply an identifier a.[1] These will be nodes in the transaction graph. Every address is also attached to an **amount** $s(a) \in \mathbb{Z}^+$ in units of Satoshis, the smallest subdivisions of Bitcoins. We have:

- 100 Satoshis = 1 microbit = 1 μ bit = 1 bit;
- 1,000 bits = 1 millibit;
- 1,000 millibits = 1 Bitcoin.

Wallet $W = \{$**Address**$\}$ A wallet is just a collection of addresses. Moreover, each address is in a unique wallet $W(a)$. We note that wallets of addresses are not publicly disclosed.

User $U = \{$**Wallet**$\}$ A user is identified by his or her collection of wallets. Moreover, each wallet is owned by a unique user $U(W)$. Users of wallets are also not publicly disclosed.

Trans $t := $ **Address, [LinkTrans:Trans]** Define a **trans** as an object used as a component of a transaction, specifically, an address which has been used as either an input or an output of a transaction. It also includes an optional linking transaction LinkTrans, which is used recursively to define the transaction graph. If LinkTrans is included, we say that trans is linked, otherwise we say that it is unlinked.

Transaction $T := \{$ **ITrans:Trans** $\}, \{$ **OTrans:Trans** $\}$ A full transaction contains one or more inputs transes and one or more output transes. Let

$$I(T) := \{I_i\}_{i=1}^n, \quad n \geq 1; \qquad O(T) := \{O_j\}_{j=1}^m, \quad m \geq 1$$

be the set of addresses of the input and output transes, respectively, of transaction T. Additionally:

- We require that $I(T) \cap O(T) = \emptyset$, that is, an address cannot be used as both an input and an output in any one transaction.
- Linked input transes are outputs of upstream transactions; linked output transes are inputs of downstream transactions.
- Unlinked output transes are called "UTXOs" (Unlinked Transaction Outputs) [2], and are Bitcoins being held for downstream transactions.
- Unlinked input transes represent newly mined Bitcoins.
- Addresses which are used as neither input nor output links are newly mined Bitcoins available for spending.
- Let $S^i(T) := \sum_{i=1}^n s(I_i)$ and $S^o(T) := \sum_{j=1}^m s(O_j)$ be the total input and output amounts of T, respectively. Then we have the subadditivity requirement that

$$S^i(T) \geq S^o(T). \tag{1}$$

  With this inequality, we can identify the **transaction cost** as

$$c(T) := S^i(T) - S^o(T) \geq 0.$$

[1]https://en.bitcoin.it/wiki/Address

	n	m
T_1	2	2
T_2	1	2

		a	$s(a)$	LinkTrans
T_1	I_1^1	g	10	null
	I_1^2	b	30	null
	O_1^1	c	20	I_1^2
	O_2^1	d	19	null
T_2	I_1^2	c	20	O_1^1
	O_1^1	e	10	null
	O_2^1	f	8	null

Table 1: (Left) Example transactions. (Right) Constituent transes.

Transaction Graph: Let $A := \{a\}$ be the set of all $N := |A|$ addresses and $\mathcal{T} := \{T\}$ be the set of all transactions. Then the **transaction graph** is a directed hypergraph [3] $H := \langle A, \mathcal{T} \rangle$. Here we interpret each transaction T as a directed hyperedge over A, with head $I(T)$ and tail $O(T)$. The vertices A of H are partitioned into three groups:

- Let $I(H) \subseteq A$ be the unlinked input transes (UTXIs). These are roots or sources.
- Let $O(H) \subseteq A$ be the UTXOs. These are leaves or sinks.
- Let $L(H) \subseteq A$ be the linked addresses. These are internal vertices.

Block $B := \{$ **Transaction** $\}$ A block is simply a group of transactions.

Blockchain $C :=$ `List` $<$ **Block** $>$ The blockchain is then simply an ordered list of such blocks.

3 Example

An example transaction graph is shown in graphical form in Fig. 1, and as a tabular database in Table 1. We have

$$A = \{b, c, d, e, f, g\}, \quad N = 6, \quad I(H) = \{b, g\}, \quad O(H) = \{d, e, f\}, \quad L(H) = \{c\}, \quad \mathcal{T} = \{T_1, T_2\}.$$

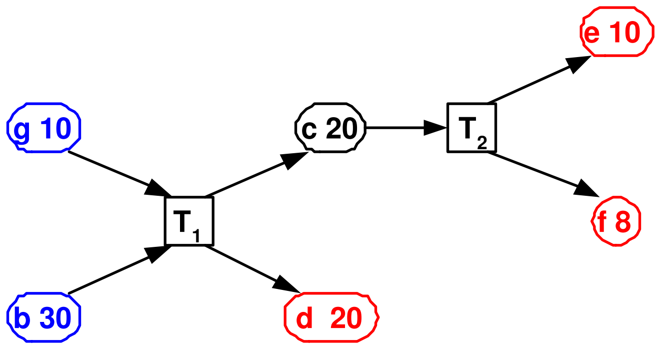

Figure 1: Bipartate graph represenation of a sample transaction hypergraph.

Fig. 1 uses the bipartate graph representation of the directed hypergraph H. Circular nodes are the addresses a shown with their amount $s(a)$. The square nodes show the two transaction hyperedges T_1, T_2. The two UTXI sources are shown in blue as newly mined Bitcoins; the three UTXO sinks in red as available for new downstream transactions. Note that subadditivity (1) is observed for both transactions, with full additivity for T_1. Transaction costs are $c(T_1) = 0, c(T_2) = 2$.

4 Discussion

We make a number of observations based on this formal development.

4.1 Cyclicity

Fig. 2 shows a transaction T_3 being added to the graph. First, some collection of UTXOs (here d, e) are identified to be linked as linked input transes for T_3 (recalling that unlinked input transes can only be newly mined Bitcoins). T_3 then creates a new collection of UTXOs, and *only* a new collection of UTXOs. So with each new transaction, n UTXOs are eliminated, while m UTXOs are added. Here addresses d and e go from red to black, while new addresses g and h are added as new UTXOs in red.

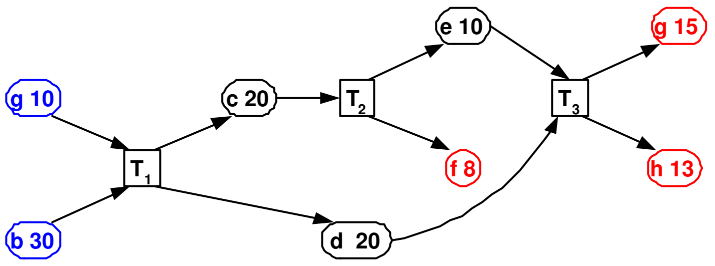

Figure 2: Adding a transaction.

Linked addresses can never be reused, in particular as outputs in new transactions. It follows that H is necessarily acyclic in the sense that the bipartate form is. Since H is a DAG, it determines a partial order, and is thus ammenable to partial order measures [6, 9]. Different definitions of acyclicity for hypergraphs [4], and in particular for directed hypergraphs, need to be addressed in order to understand what statements can be made about the acyclicity of H as a hypergraph.

4.2 Binary Edges and Trees

As a mathematical matter, in the special case when $n = m = 1$ for some transaction T, that represents a situation where a single quantity of Bitcoin s is transferred from a single address I to a single address O. This is a special case where the hyperedge of transaction T can be replaced by a binary graph edge, as shown in Fig. 3: here a small bipartate sub-hypergraph T_1 is replaced by a (trivially) small directed graph consisting of a single binary edge, now with the *label* T_1. Since as noted, the quantity $s(a)$ is fixed for each address a, it follows that $s(I) = s(O)$: here the address $\langle g, 10\rangle$ goes to $\langle c, 10\rangle$.

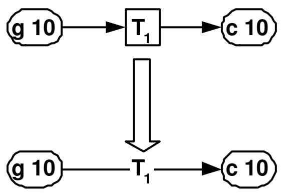

Figure 3: Special case of a minimal directed hyperedge replaced by a labeled directed binary edge.

If this continues, then s is simply moved from one address to another along a chain. This may, in fact, be a significant situation methodologically, as it could be used in laundering schemes to change addresses, wallets, and users. But as a formal matter, it suffices to note this.

Moving on, we consider the somewhat more general case where $n = 1$ for $m > 1$. Now there is only "fan-out" of a single input I being distributed to potentially multiple output addresses O_j, so that $s(I) \geq S^o(T)$.[2] While of course this need not hold in general, it may over portions of H, and under some special interpretations discussed below. In this case it is also possible to replace hyperedges, now not with individual binary edges, but with small sub-trees where each branch is labeled with the hyperedge, as show in Fig. 4. The result is that all of H is a binary (in the sense of binary graph) tree.

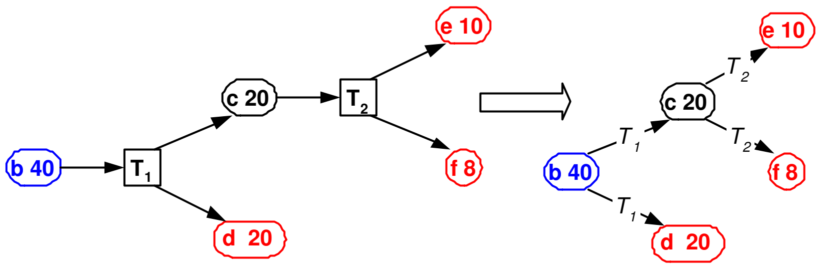

Figure 4: Special case of a tree.

4.3 Time

Since new transactions can only be built from UTXOs, it follows that transactions are ordered in time. So in Fig. 2, we can place $T_1 < T_2 < T_3$ into a linear order. Mathematically, this is actually a "preorder", allowing transactions to be able to occur simultaneously, as long as they're not in a chain of the graph.

4.4 User Graph

Each address has a unique user through its wallet as $U(W(a))$, which we simply call $U(a)$. Thus U partitions H vertex-wise. This is shown on the left side of Fig. 5 for an example assignment of users.[3]

[2]Mathematically, there is a dual case where $m = 1$ for $n > 1$, and there is only "fan-in", but our judgement at this point is that that only adds complexity to the formal development without methodological significance.

[3]Note that for display purposes, the address $\langle e, 10\rangle$ has been moved up and to the left.

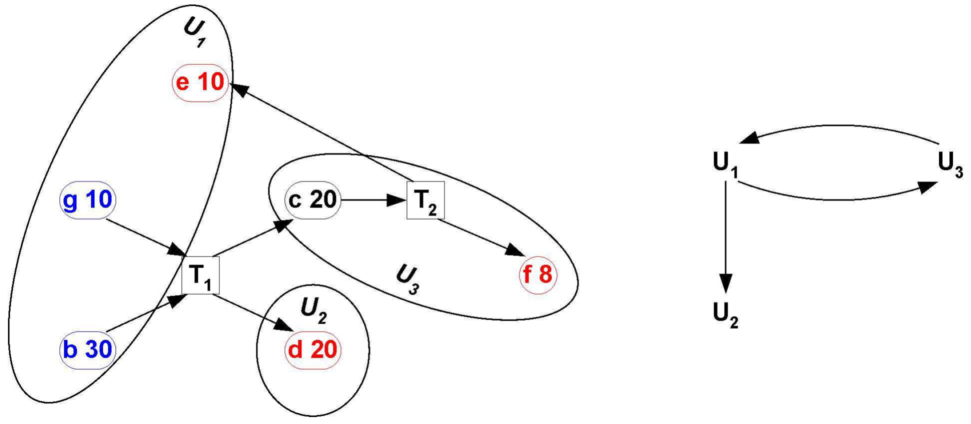

Figure 5: (Left) Transaction graph adorned with some user identities. (Right) User graph contraction $U(H)$.

Some other issues of interest here include the following:

- Since U partitions H, we can construct a **user graph contraction**, denoted $U(H)$, by replacing sets of address vertices from the same user with user supernodes. Supernodes are then connected with an edge if there is any edge from any address in one supernode to any address in another. The results are shown on the right side of Fig. 5. Note that unlike H, $U(H)$ is a regular binary graph, not a hypergraph. Additionally, it is not a DAG, as seen in the example.

- In the transaction T_2, 10 Satoshis from user U_3 are sent to address e, user U_1, while 8 are sent *back* to user U_3 at address f (note a transaction cost of 2). Formally, we have $\exists I_i, \exists O_j, U(I_i) = U(O_j)$. This is a **change making** pattern, and is quite prominent.

- In the transaction T_1, the two input addresses $\langle b, 30\rangle, \langle g, 10\rangle$ are both from U_1. We say that U_1 is **combining** multiple quantities into a single transaction, formally $\forall 1 \leq i, i' \leq n, U(I_i) = U(I_{i'})$.

- Continuing, while in T_1, the 40 Satoshis are then sent to two different users U_2, U_3, it is common in some transactions for them to go back to the original user U_1. Formally, this condition:
$$m = 1; \qquad \forall 1 \leq i, i' \leq n, \quad U(I_i) = U(I_{i'}) = U(O)$$
we can identify as **consolidation**, and is another prominent pattern.

- We noted above that discovering the wallets and users of addresses can be difficult. The truth is, it is widely believed that multiple inputs ($n > 1$) from *different* users (that is, *without* combining from a single user) is very *uncommon*. On that basis, a popular analytical approach [11] is to impute a single user identity to all the inputs of each transaction, at least to a first approximation.

  Fig. 6 shows this in our example, with imputed users V_1 for b and g and V_2 for c. d, e and f are UTXOs, and so cannot have imputed users, and so are included as themselves, forming all (and only) the leaves.

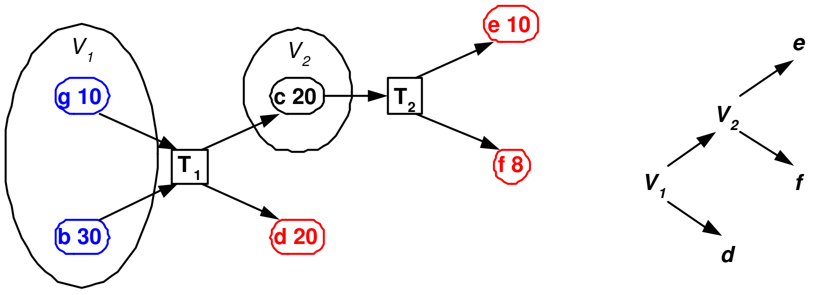

Figure 6: (Left) Imputed users. (Right) User contraction graph $U(H)$ (including UTXOs) is a tree.

Note that the resulting user contraction graph $U(H)$ here is necessarily a tree, whereas in Fig. 5 it is not. That's because while we can impute that all the inputs of a transaction are from the same user, we can't impute that some of the imputed users might themselves be identical. Rather it's as if this imputation process merely collapses all the n inputs of each transaction, actually recovering the situation described in Sec. 4.2 above. In this case, we happen to know that $U(e) = U(g) = U_1$, so that in Fig. 6 e should actually be included in V_1, so that there's an edge from V_2 to V_1, making $U(H)$ not a tree.

4.5 Block Chain

In the mining process, transactions are combined into blocks. So where user identity partitions *addresses*, block identity partitions *transactions*. This is shown on the left of Fig. 7. Note that we consider a transaction T to include all its inputs $I(T)$ and outputs $O(T)$, so that linked addresses are considered as members of both their upstream and downstream transactions, resulting in blocks overlapping.

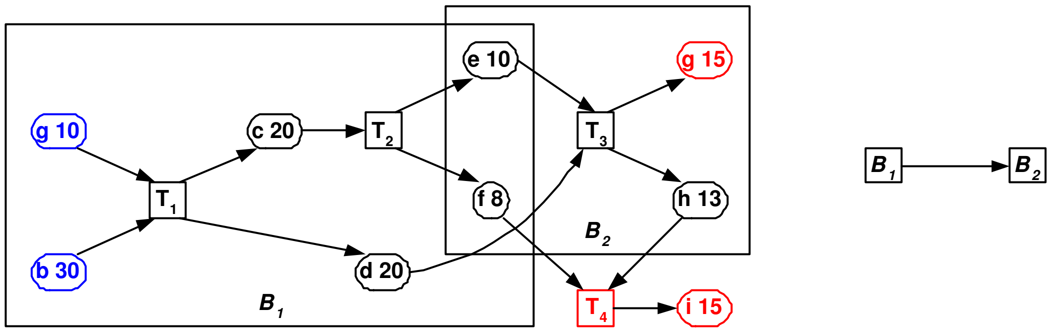

Figure 7: (Left) An extended example transaction hypergraph with blocks identified. (Right) Resulting block chain.

At any time, some transactions are unblocked, as shown for a new transaction T_4 in red. These are avalable for competion by miners for future blocks.

New blocks are added to the list of existing blocks by simply appending them in a list. The resulting block chain (as shown on the right of Fig. 7) is simply a linearly (more or less temporally) ordered list. However, based on our note about overlapping blocks above, it may be possible to reconstruct the blockchain as a more complex object in virtue of shared addresses among blocks.

4.6 Fungibility

The quantities $s(a)$ attached to each address are imutable: once created, they cannot be changed. It follows that in a transaction, the *entire* quantity of Satoshis at an input address I_i must be transferred into the transaction, rather than a portion. Furthermore, that entirety of $s(I_i)$ is then transferred to the potentially *multiple* output addresses O_j. Two related observations emerge:

- When two inputs I_1, I_2 are combined and sent to two outputs O_1, O_2, it is not possible to establish the amount or proprtion of Satoshis from either I_1 or I_2 which go to either O_1 or O_2.
- Within any one address I_1, it is not possible to distinguish, label, or identify any one Satoshi from another. It is thus not possible to say that "this" one Satoshi from the $s(I_1)$ at address I_1 went to "that" Satoshi amongst the $s(O_1)$ now at address O_1.

In this sense, Satoshis are fungible, like real money. In our example, in T_1, when the 10 Satoshis from g are combined with the 30 from b, they are pooled, and then 20 sent to c and 20 to d. It is not possible to assign or apportion some from g to c or d, but not others, nor to track which go where. In Fig. 7, of the 15 Satoshis ending up at address i, only weak bounds are available to express how many came from T_3 (between 7 and 13) or from T_2 (between 2 and 8). And even if we were to know that, say, 2 of the Satoshis at i came from f, it still wouldn't be meaningful to ask *which two* of the 8 available there made the journey, and which 6 went to the transaction cost.

In this way, Bitcoins are like real money. However, treasury notes have serial numbers, and bank transfers can be tracked. Similarly, the pseudonymous nature of the transaction graph is entirely public. Thus, while tracking individual Satoshis through individual transaction isn't possible, interrogating the entire transaction graph yields all the information that would otherwise be needed to do so. In this sense, while Bitcoins are fungible, this fungibility becomes meaningless and inoperable *given knowledge of the whole transaction graph.*

4.7 Flow Models

The subadditivity requirement of (1) provides some important structure on H. If equality, rather than inequality, held in (1), then, to a first approximation, H would implement a **flow network** [1]. Network flows are defined on binary graphs, not hypergraphs, but map to hypergraph models because of conservation over transitive edges. Flow conservation would then imply that for any fixed H, the total input flow (the sum of all UTXI amounts) equals the output flow (the sum of all UTXI amounts):

$$\sum_{a \in I(H)} s(a) = \sum_{a \in O(H)} s(a).$$

The total discrepancy between these sums, the total transaction cost of the network, represents a level of "friction" in the flow network.

While imutability of amounts and fungibility of Bitcoin implies that downstream amounts cannot be assigned to particular upstream sources, it is possible to construct *models* of flows as if you could.

In particular, one could assume uniform distributions and proportional assignment over a Markov process over H as a "mixing" model to *apportion* fractions of Satoshis to downstream transactions. This is shown in Fig. 8 for our example. The 40 input Satoshis to T_1 are assigned proportionally to c and d, as noted. These then flow down to T_2.[4] It is noteworthy, however, that if done so, such Markov processes are likely "ergodic", resulting, in a large graph, in effectively arbitrary or smooth mixing: downstream addresses will contain uniformly "smeared" contributions from *all* upstream transactions.

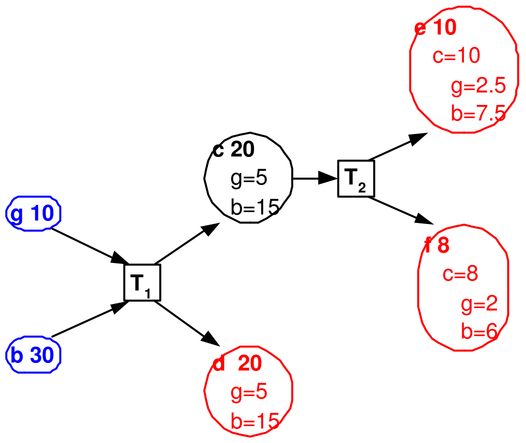

Figure 8: Proportional assignments in a Markov model.

Finally here, we note potential relationships to other modeling and analytical methods:

- (1) can be interpreted as a conservation principle in a network physics system, although with potential frictional losses due to it being a subadditive inequality, that is, that there are costs $c(T)$ associated with transactions. In flow models, this invokes Kirchoff's laws in electrical circuits [7] and similar results in analogous systems.
- Recently, directed hypergraphs have been advanced as relatively simple models of chemical organization and evolution [10].
- Finally, we simply note in passing that we [5] have advanced methods for working with distributions of labels on directed mlutigraphs. In particular, our measures of dispersion κ as a log "compression" ratio, and smoothness G as a normalized entropy, are proving promising as measures on integer partitions to measure the distributions of IPs and ports in cyber flow analytics. These map directly to these kinds of hypergraph models, and are available to us as well, as introduced below.

5 Analytical Quantities

We conclude by simply listing some obvious candidates of things we'd like to calculate on real Bitcoin transaction graphs H. We intend that future drafts will show these quantities over more interesting example graphs.

[4] A more complicated example as in Fig. 7 would be more illustrative and satisfying, but that awaits a future draft.

Transaction Level: Items for any individual transaction.

- Number of inputs and outputs: n, m.
- Total input and output amount: $S^i(T), S^o(T)$.
- Cost: $c(T)$.
- Smoothness G and dispersion κ of the input I_i and output O_j amounts as count distributions.

Graph Overall: Items over an entire graph, showing some from our example from Fig. 1.

- Averages and variances of all of the above.
- Graph size: $N = 6, |\mathcal{T}| = 2$.
- Proportion of UTXIs, UTXOs, and linked addresses:

$$\frac{|I(H)|}{N} = 1/3, \quad \frac{|O(H)|}{N} = 1/2, \quad \frac{|L(H)|}{N} = 1/6.$$

- Overall input, output, and cost amounts:

$$\sum_{a \in I(H)} s(a) = 40, \quad \sum_{a \in O(H)} s(a) = 38, \quad \sum_{a \in I(H)} s(a) - \sum_{a \in O(H)} s(a) = 2.$$

- Smoothness G and dispersion κ of the UTXI and OTXO amounts.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms and Applications.* Prentice Hall, 1993.

[2] Andreas M. Antonopoulos. *Mastering Bitcoin: Unlocking Digital Cryptocurrencies.* O'Reilly Media, 2014.

[3] Giorgio Ausiello, Paolo G Fanciosa, and Daniele Frigioni. Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach. In *ICTCS 2001, LNCS*, volume 2202, pages 312–328. 2001.

[4] C Berge. *Graphs and Hypergraphs.* North-Holland, London, 1973.

[5] Cliff Joslyn, Wendy Cowley, Emilie Hogan, and Bryan Olsen. Discrete mathematical approaches to graph-based traffic analysis. In *2014 Int. Wshop. on Engineering Cyber Security and Resilience (ECSaR14)*, 2014.

[6] Cliff Joslyn, Patrick Paulson, and Amanda White. Measuring the structural preservation of semantic hierarchy alignments. In *Proc. 4th Int. Wshop. on Ontology Matching (OM-2009), CEUR*, volume 551, 2009.

[7] Don Mikulecky. *Application of Network Thermodynamics to Problems in Biomedical Engineering.* NYU Press, 1993.

[8] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. https://bitcoin.org/bitcoin.pdf

[9] Chris Orum and Cliff A Joslyn. Valuations and metrics on partially ordered sets, 2009. submitted.

[10] Stephan Peter and Peter Dittrich. On the relation between organizations and limit sets in chemical reaction systems. *Advances in Complex Systems*, 14(1):77–96, 2011.

[11] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In *Financial Cryptography and Data Secutiry, LNCS*, volume 7859, pages 6–24. 2013.