A Mathematical Model of Financial Bookkeeping

Robert Wright

18th May 2018

Abstract

In this paper we introduce a mathematical model for an entire bookkeeping system. Building on the work of Ellerman, we define objects representing value, accounts, and transactions, and introduce a convenient notation for each of the above. This notation allows you to define transactions locally – without knowing, a priori, all possible value types, and accounts. We then use these mathematical objects to formally prove known accounting results.

We also extend transactions to temporal transactions, add a tree structure to the set of accounts, and create an object representing an entire ledger. These allow you to express as single objects financial events that are conceptually a single event, but must be expressed as multiple objects in standard financial bookkeeping (such as depreciation).

1 Introduction

To run a successful business, you need to keep track of your transactions[1], so as to know how much capital, and goods, you have available to you. You do this, as you cannot sell any goods, if you don't know that you have any to sell in the first place.

Even if you didn't need to do this, you'd still need to keep track of your transactions, to know that your business is, in fact, successful.

Double entry bookkeeping, as formalized by Luca Pacioli, in 1494, [6], is one method of doing this. Many businesses have used this method of bookkeeping, largely unchanged, from ancient times [2], 'till the modern day.

Double entry bookkeeping provides you with a collection of disconnected mathematical objects, and rules to manipulate these objects. The aim of this paper is to build a mathematical bookkeeping system from the ground up, to include these objects, and encode these rules, so as to unify the system, and allow mathematical analysis of the whole.

2 Value

2.1 Money Space

To start talk of bookkeeping, we fist need a thing to keep track of – some concept of money. What "money" actually means to us depends on our specific circumstances, but let's call the set of all possible monies $\mathcal{M}$.

Example 1. *For example, if the only money we accept is the GBP, then our money space would be*

$$\mathcal{M} = \{£0.00, £0.01, £0.02, ...\}. \tag{1}$$

The smallest amount of money we can have is nothing, and the primitive operation we can do with our money is to add it together. If we have one pile of money, and another pile of money, then we can put them together into a bigger pile of money.

So we want some operation $+$, that allows us to add values together in the way that we expect, and some value 0, which represents the value of nothing. Mathematically,

Definition 1. *A* value space $\mathcal{M}$ *is a commutative monoid.*

I'm glossing over the precise definition of a commutative monoid here, as, from an accounting perspective, everything works as you expect from the analogy of piles of money on the floor. More details on monoids may be found in any introductory work on algebraic structures, such as [1].

[1]Of course, as I write this, I wonder if it's possible to set up a business with certain local rules, as to make a global set of accounts unnecessary, but I digress.

2.2 Value Space

But companies do more than just take money[2]. They also need to purchase raw materials, and pay employees. So we need some way to represent not only the money, but also whether it's a gain, or a loss.

Naively, we want to add a $-$ operation to $\mathcal{M}$, so that we can take some money from another amount. But in order for this to make sense, we need to add negative numbers to our space as well (so we can do things like $£0 - £10$).

Historically, western mathematicians were uncomfortable with the concept of negative numbers [3], and so instead the concepts of debits and credits were used. These two methods are, in fact equivalent, as shown by the Pacioli group, as defined in Ellerman, *On Double Entry Bookkeeping* [5].

Definition 2. *The* Pacioli group*, P, of a commutative monoid M is the set of ordered pairs $[\,a\,/\!/\,b\,]$ with both elements in M, under the equivalence relation $[\,a\,/\!/\,b\,]\,R\,[\,c\,/\!/\,d\,]$ when $a + d = b + c$.*

Further, we define $+$ pointwise on P, define unary $-$ by $-[\,a\,/\!/\,b\,] = [\,b\,/\!/\,a\,]$, and binary $-$ by $u - v = u + (-v)$, for all $a, b \in M$, and $u, v \in P$.

Definition 3. *A* value space *$\mathcal{V}$, is the Paciloi group of a money space $\mathcal{M}$.*

In a pair $[\,a\,/\!/\,b\,]$ in a value space $\mathcal{V}$, we think of a as a debit, and b a credit.

Notation 1. *We identify a money a with the pair $[\,a\,/\!/\,0\,]$, so that we can write*

$$[\,a\,/\!/\,b\,] = a - b.$$

This is equivalent to identifying the Pacioli group with its debit isomorphism, as defined in Ellerman, Economics, Accounting, and Property Theory *[4].*

Note that the choice of identifying a money a with the debit component of the pair is completely arbitrary, and we could instead use the credit component $a \mapsto [\,0\,/\!/\,a\,]$. All that would do would flip all the $+$ and $-$ signs.

Example 2. *Using* (1) *for $\mathcal{M}$, our value space is*

$$\mathcal{V} = \{..., -£0.02, -£0.01, £0.00, £0.01, £0.02, ...\}$$

where a positive amount represents a debit, and a negative amount a credit.

Example 3. *If you have a $£10$ note, and owe a friend $£5$, then your net worth is*

$$£10 - £5 = £5$$

leaving you with $£5$.

[2]We hope

If instead you owed your friend £20, then your net worth is

$$£10 - £20 = -£10$$

leaving you £10 in debt.

3 Accounts

3.1 Individual bookkeeping

If there's only one entity in all of existence, then transactions are very boring, as all that can happen is that entity giving things to themselves. As such, we need to consider more than one entity, but how many?

We generally do bookkeeping to keep track of our transactions with the outside world. So we could just consider two entities, Me and Not Me.

This allows us to record what transactions happen, but usually we also want to record more details than that. We care for what purpose we're making each transaction, and we care who we're making the transaction with. This can be represented by splitting Me and Not Me, respectively, into smaller parts. How small those parts are depends on how much detail you want to record.

Thus, we have a finite collection of entities involved in our accounting. Call each of these entities an "account", and let $\mathcal{A}$ be the set of all accounts. From a mathematical perspective, accounts are just labels, and what they represent doesn't matter. From an accounting perspective, accounts allow you to keep track of where your money is coming from, and going to.

3.2 Distributed bookkeeping

As a side note, nothing is requiring us to view our accounting from the point of view of any particular entity. A collection of individuals could theoretically come to a common agreement on what an account is, and maintain their books together. In practice, this would probably require some sort of distributed bookkeeping software (eg. in the style of Bitcoin).

4 Transactions

Now that we've got our entities and values, the natural question to ask is how much each person has. The state of a financial system, as it were.

Definition 4. *A* financial state *f is a vector, with values in $\mathcal{V}$, and indexed by $\mathcal{A}$.*

A financial state represents the state of an entire financial system at a point in time. It includes the total value owned by each of the entities involved in the system.

But a financial state can also represent a change in the state of a financial system. Adding our vectors together then represents applying that change to the system.

For a transaction, the same amount of "stuff" exists before as after. This motivates the following definition.

Definition 5. *A* transaction t *is a financial state, with*

$$\sum_{A \in \mathcal{A}} t_A = 0.$$

Let $\mathcal{T}$ be the set of transactions.

As "positives" (debits) and "negatives" (credits) cancel each other out, this is saying that the total of the debits is the same as the total of the credits, a notion familiar from double-entry bookkeeping.

Example 4. *Suppose we have three accounts, A, B, and C, and index our vectors respectively. Then the financial state where A has £10, B has £5, and C has nothing is represented by*

$$(£10, £5, £0), \tag{2}$$

and the transaction of £5 from A to C is represented by

$$(-£5, £0, £5). \tag{3}$$

So if we were to start at financial state (2), and then apply transaction (3), we would be in the financial state represented by

$$\begin{aligned}(£10, £5, £0) + (-£5, £0, £5) &= (£10 - £5, £5 + £0, £0 + £5), \\ &= (£5, £5, £5).\end{aligned} \tag{4}$$

That is, all accounts have £5.

This notation is a little clunky, so let's rephrase to make things easier.

Notation 2. *For a value $a \in \mathcal{V}$, and accounts $A, B \in \mathcal{A}$, I define a_A to be the financial state with amount a for account A, and 0 elsewhere. Further, I define $a_{A\to B} = a_B - a_A$.*

Note that $a_{A\to B}$ represents a transaction transferring amount a from account A, to account B.

Example 5. *Rephrasing the transaction application (4) with our new notation, we have*

$$\begin{aligned}(£10_A + £5_B) + (£5_{A\to C}) &= (£10 - £5)_A + £5_B + £5_C, \\ &= £5_A + £5_B + £5_C.\end{aligned}$$

This notation also allows us to define things locally, in the sense that we only need to know about the accounts involved in the transaction.

5 What's "double" about double-entry?

The name of double-entry bookkeeping comes from the recognition that transactions have both a source, and a destination. Therefore, a standard transaction, with one source, and one destination, must be written down twice, once for each account. In a very literal sense, the entry is doubled.

Similarly, we could recognize that, in a transaction, the value has to come from somewhere, and it has to go somewhere. A kind of "Conservation of Value", if you will (to take inspiration from physics naming conventions), or "Fundamental Theorem of Accounting" (to take inspiration from mathematics naming conventions).[3]

Theorem 1. *The total value in a financial system is the same before and after a transaction, where the total value of a system, with a financial state f and set of accounts $\mathcal{A}$, is given by*

$$\sum_{A \in \mathcal{A}} f_A.$$

Proof. Let f be a financial state, t a transaction, and $\mathcal{A}$ the set of accounts.

Then the total value of the financial system after the transaction is $\sum_{A \in \mathcal{A}} (f+t)_A$. This gives

$$\begin{aligned}
\sum_{A \in \mathcal{A}} (f+t)_A &= \sum_{A \in \mathcal{A}} f_A + \sum_{A \in \mathcal{A}} t_A, && \text{rearranging the sum,} \\
&= \sum_{A \in \mathcal{A}} f_A + 0, && \text{by definition of a transaction,} \\
&= \sum_{A \in \mathcal{A}} f_A.
\end{aligned}$$

□

Note that we could equally define transactions as value preserving changes in financial states, and then derive the current definition from that.

For more complicated transactions, we can have more than two non-zero entries, so perhaps the name "double" is a bit of a misnomer. On the other hand, we still require a balance between the "debits" and "credits", which is a sort of doubling up of values.

6 Time

So we now have transactions, and a way of combining them (addition). But the order in which these transactions occur is still important, as (for example) you can't sell goods before you own them. We could just take a list of the transactions, but then we forget when exactly something happened, which may be important for reporting purposes.

[3] Though with the Fundamental Theorems of Algebra and Arithmetic, I think mathematics has enough FTAs.

So we probably want to record our transactions with some sort of timestamp. So we need some concept of time.

Definition 6. *Let our set of* times $\mathbb{T}$ *be a total order.*

But if we want to add together multiple transactions with different timestamps, then we need to extend our space of transactions to allow transactions that take different values at different times.

Definition 7. *Let a* temporal transaction *be a function* $\mathbb{T} \to \mathcal{T}$ *from time to transactions.*

Define the various transaction operations on temporal transactions pointwise. We say a temporal transaction t involves *an account* A*, if there is a time* τ*, with* $t(\tau)_A \neq 0$*.*

The most natural class of temporal transactions are changes that takes place instantaneously, at a certain point in time. Let's call these step transactions.

Definition 8. *For a transaction* t*, and a time* τ*, define the* step transaction t^τ *to be the temporal transaction given by*

$$t^\tau(\tau') = \begin{cases} 0 & \tau' < \tau \\ t & \tau' \geq \tau \end{cases}$$

for all times τ'*.*

Note that classic double-entry bookkeeping uses only step transactions, as transactions are applied instantaneously. More complicated transactions are then represented by multiple step transactions. Temporal transactions, on the other hand, allow a much richer description of how transactions take place. We will consider some more complicated examples of temporal transactions later.

7 Ledgers

Now let's put it all together. A financial system consists of multiple transactions, so we need some sort of collection to hold them all, which we'll call our ledger.

We could use a list to represent our ledger, but we've already captured our concept of order in temporal transactions. As we don't need to consider order again, we could use a set, but we could, in theory, have two of the exact same transaction at the exact same time. So we want a set with appropriate counting of duplicates.

Definition 9. *A* ledger *is a finite*[4] *multi-set of temporal transactions.*

This ledger represents all the transactions in a financial system. We can now speak of things like the balance of an account, and interaction of accounts.

[4] We could probably be less restrictive than this, but all real-world examples are finite

Definition 10. *The* balance b *of an account* A*, in a ledger* L*, is a function* $\mathbb{T} \to \mathcal{V}$*, where the balance at time* τ *is given by*

$$b(\tau) = \sum_{t \in L} t(\tau)_A.$$

Definition 11. *We say two accounts* A *and* B *have* interacted *in a ledger* L*, if there is a temporal transaction* $t \in L$*, which involves both* A *and* B*.*

8 Multi-dimensional value space

8.1 Product money space

Unfortunately, there is more than one type of money, and we can also trade in goods not generally considered as money. These different commodities are not directly compatible, in an "apples and oranges" manner.

Example 6. *If we consider the money spaces*

$$\pounds = \{\pounds 0.00, \pounds 0.01, \pounds 0.02, ...\}, \textit{ and}$$

$$\$ = \{\$0.00, \$0.01, \$0.02, ...\},$$

then $\pounds 10$ *is not interchangeable with any amounts of* $\$$.[5]

If we have two different money spaces, $\mathcal{M}_1$, and $\mathcal{M}_2$, then we might want to define the money space consisting of *both* the monies $\mathcal{M}_1$ and $\mathcal{M}_2$. But to make this new space a money space, we'd also have to include all possible sums of values from both these spaces. Mathematically, this is known as the product space (product as in multiplication, rather than something you sell).

Definition 12. *Given two money spaces* $\mathcal{M}_1$*, and* $\mathcal{M}_2$*, we define the* product money space $\mathcal{M}_1 \times \mathcal{M}_2$ *to be the set of pairs* (m_1, m_2)*, with* $m_1 \in \mathcal{M}_1$*, and* $m_2 \in \mathcal{M}_2$*.*

Further, we make the product money space into a money space by defining addition component-wise.

$$(m_1, m_2) + (n_1, n_2) = (m_1 + n_1, m_2 + n_2).$$

Example 7. *For the above money spaces of* $\pounds$ *and* $\$$*, the product money space is*

$$\begin{aligned}\pounds \times \$ = \{ & \\ & \begin{matrix} (\pounds 0.00, \$0.00), & (\pounds 0.00, \$0.01), & (\pounds 0.00, \$0.02), & \dots \\ (\pounds 0.01, \$0.00), & (\pounds 0.01, \$0.01), & (\pounds 0.01, \$0.02), & \dots \\ (\pounds 0.02, \$0.00), & (\pounds 0.02, \$0.01), & (\pounds 0.02, \$0.02), & \dots \\ \vdots & \vdots & \vdots & \ddots \end{matrix} \\ \}. & \end{aligned}$$

[5] If your first reaction involves the phrase "exchange rate", then note that you're exchanging the £10 *with someone.* So what you're actually doing is a transaction where you're purchasing some amount of \$ for £10. You can't change £10 to some amount of \$ without getting someone else involved.

By repeating this process, we can make money spaces out of arbitrarily many smaller money spaces.

Similarly to transactions, product money spaces require you to know the shape of the whole space to be able to write down an element of it. So we introduce analogous notation to allow you to express monies in product money spaces locally, without needing to know all components.

Notation 3. *We identify the monies of the two spaces with the appropriate subspaces of the product money space*

$$m_1 \mapsto (m_1, 0), m_2 \mapsto (0, m_2)$$

for all $m_1 \in \mathcal{M}_1$, and $m_2 \in \mathcal{M}_2$.

Example 8. *Then, in the space £ × \$, we can write things like £10 + \$10 to mean (£10, \$10) as*

$$\begin{aligned} £10 + \$10 &= (£10, \$0) + (£0, \$10) \\ &= (£10, \$10) \end{aligned}$$

8.2 Product value space

We can similarly define the combination of two value spaces, and show that the value space of a product space is essentially the same as the product space of a value space.

Definition 13. *Given two value spaces $\mathcal{V}_1$, and $\mathcal{V}_2$, we define the* product value space $\mathcal{V}_1 \times \mathcal{V}_2$ *to be the set of pairs (v_1, v_2), with $v_1 \in \mathcal{V}_1$, and $v_2 \in \mathcal{V}_2$.*

Further, we define value space operations on $\mathcal{V}_1 \times \mathcal{V}_2$ componentwise.

Theorem 2. *If we have two money spaces $\mathcal{M}_1$, and $\mathcal{M}_2$, with corresponding value spaces $\mathcal{V}_1$, and $\mathcal{V}_2$, then the value space $\mathcal{V}$ of the product money space $\mathcal{M}_1 \times \mathcal{M}_2$ is isomorphic to $\mathcal{V}_1 \times \mathcal{V}_2$.*

Proof. The isomorphism is given by

$$\left[\, (m_1, m_2) \mathbin{//} (m_1', m_2') \,\right] \mapsto \left(\left[\, m_1 \mathbin{//} m_1' \,\right], \left[\, m_2 \mathbin{//} m_2' \,\right] \right).$$

Proof that this is an isomorphism is omitted, as it is long winded, but unenlightening. □

This shows that we can take products of spaces and value spaces of money spaces in any order, without risking confusion.

8.3 Non-standard money types

As well as freely adding money types, we can consider types that we do not normally think of as monies.

In classical double entry bookkeeping, the purchase of an asset decreases the balance of our cash account, and increases the balance of our assets account by the same amount. But this really represents a transfer of money to some external entity, in exchange for that asset. So we can instead encode the asset as a new money type, and have the transaction as a literal exchange of cash for asset.

Example 9. *For example, a transaction where I buy a sheep from Fred for* £100 *could be represented by*

$$\mathrm{Sheep}_{\mathrm{Fred}\to\mathrm{Me}} + £100_{\mathrm{Me}\to\mathrm{Fred}}.$$

9 Account Personification

The use of a sheep as a money type doesn't seem to make sense from the classical accounting point of view, as a sheep is not a persistent store of value. If I have a sheep, and the sheep dies, then there is no longer a sheep. So how can we represent it in our bookkeeping system, if one day it will disappear?

This seeming contradiction can be resolved by considering the death of the sheep as another sort of transaction. If we personify of the process of death, say by creating an account called Death, then we can represent the death of the sheep by transferring the sheep from ourselves, to Death.

Example 10. *The death of a sheep could be represented by*

$$\mathrm{Sheep}_{\mathrm{Me}\to\textsc{Death}}.$$

We can also use this idea for other sorts of processes.

Example 11. *As an example where we are receiving something from a process, consider lambing season.*

$$10\,\mathrm{Sheep}_{\mathrm{Mother\,Nature}\to\mathrm{Me}}$$

We can even have a transaction where we exchange some things for others.

Example 12. *For an example transaction with a process with movement in both directions, consider the construction of a chair.*

$$(3\,\mathrm{Planks\,of\,wood} + 4\,\mathrm{Nails} + 5\,\mathrm{Man\,Hours})_{\mathrm{Me}\to\mathrm{Carpentry}} + 1\,\mathrm{Chair}_{\mathrm{Carpentry}\to\mathrm{Me}}$$

This allows you to encode your stock control into your bookkeeping system, by setting up a money type for each type of stock. By increasing the level of detail you use for your money types, you can record more information about your stock. For example, for the sheep above, you could encode them as Animals, as Sheep, or you could even set up a money type for every individual sheep, allowing you to track each one precisely.

10 More Transaction Types

Temporal transactions can represent in a single object things that would be represented by multiple transactions classically. For example, consider straight-line appreciation, where we appreciate with transaction t at equally spaced times τ_1, τ_2, ..., τ_n. This could be represented by a collection of step transactions $\{t^{\tau_i}\}_{i=1}^{n}$, or it could also be represented by a single temporal transaction t', given by

$$t'(\tau) = \sum_{\tau_i} t^{\tau_i}(\tau).$$

If we increase the frequency of the step transactions, while keeping the final change the same, we can approximate the appreciation occurring continuously with time.

Indeed, given appropriate operations on times, we can define a continuous straight-line appreciation l of value a, from time τ_1 to τ_2, as a function $\mathbb{T} \to \mathcal{V}$, such that l at time τ is given by

$$l(\tau) = \begin{cases} 0 & \tau < \tau_1 \\ \frac{\tau-\tau_1}{\tau_2-\tau_1} a & \tau_1 \leq \tau < \tau_2 \\ a & \tau \geq \tau_2 \end{cases}.$$

Other forms of appreciation, depreciation, and amortization may be represented classically as a collection of step transactions, and so may also be represented as a single temporal transaction by adding together the component step transactions.

By taking smaller and smaller time steps, we may then approximate the continuous time equivalents. These continuous time equivalents may usually be represented by simple functions $\mathbb{T} \to \mathcal{V}$. For example, straight-line is a linear function on it's period of influence, double-declining is piecewise exponential and linear, sum-of-years-digits is quadratic, and so on.

Notably this method doesn't work for units-of-production depreciation, as the amount to depreciate by is not known in advance.

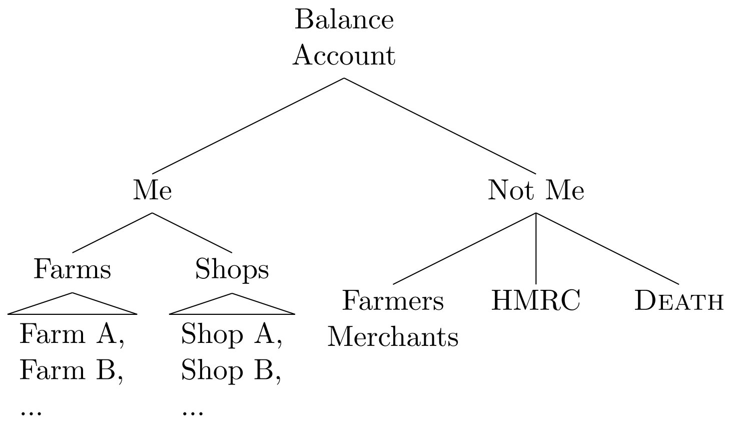
11 Tree of Accounts

Previously, we split our Me account into multiple smaller accounts, to keep track of why we're making each transaction. But we may still want to ask what the balance of the Me account is, and consider the smaller accounts as sub-accounts of Me. In turn, we may want those sub-accounts to have sub-sub-accounts, and so on, so that we may consider our accounts at varying levels of detail. This motivates the following definition.

Definition 14. *A* tree of accounts $\mathcal{A}$ *is a rooted tree. The account that is the root of* $\mathcal{A}$ *is called the* balance account*.*

For accounts A and B, we say $A \leq B$ if B is a descendant of A (that is, if B is A, is a sub-account of A, a sub-sub-account of A, or so on).

Example 13. *For example, a farmer who sells his own produce at self-run farm shops may have a tree of accounts similar to*



Then Farm A *is a sub-account of* Farms*, which is in turn a sub-account of* Me*. So* Farm A *is a descendant of both* Farms *and* Me*, written* Farms $\leq$ Farm A*, and* Me $\leq$ Farm A*, respectively.*

Now we can add up all the balances of the descendant accounts of an account, to find out what its balance would be, if we didn't have all of those sub-accounts.

Definition 15. *The* total balance b *of an account* A*, in a ledger* L*, is a function* $\mathbb{T} \to \mathcal{V}$*, where the total balance at time* τ *is given by*

$$b(\tau) = \sum_{t \in L, A \leq B} t(\tau)_B.$$

Theorem 3. *The total balance of the balance account is* 0 *at all times.*

Proof. As the balance account R is the root of the tree of accounts $\mathcal{A}$, then we have $R \leq A$ for all accounts $A \in \mathcal{A}$. So the total balance b of R at time τ is

$$\begin{aligned}
b(\tau) &= \sum_{t \in L, R \leq A} t(\tau)_A, && \text{by definition of total balance,} \\
&= \sum_{t \in L, A \in \mathcal{A}} t(\tau)_A, && \text{as } R \text{ is the root account,} \\
&= \sum_{t \in L} \left(\sum_{A \in \mathcal{A}} t(\tau)_A \right), && \text{rephrasing the above,} \\
&= \sum_{t \in L} 0, && \text{as each } t(\tau)_A \text{ is a transaction,} \\
&= 0.
\end{aligned}$$

□

So the balance of the balance account can be used as a checksum, to ensure that the balances of all other accounts have been calculated correctly.

Theorem 4. *For an account B, of balance b, that has interacted with only the descendants of an account A, the contribution $c(\tau)$ of the total balance of A, at time τ, by transactions involving B is $-b(\tau)$.*

Proof. Consider a transaction t that involves B. As B interacts only with descendants of A, then $t_C = 0$ when $C \neq B$ and $A \leq C$.

So

$$\begin{aligned} 0 &= \sum_{C \in \mathcal{A}} t_C, && \text{as } t \text{ a transaction,} \\ &= \sum_{A \leq C} t_C + \sum_{A \nleq C, C \neq B} t_C + t_B, && \text{splitting apart the sum,} \\ &= \sum_{A \leq C} t_C + t_B, && \text{as each } t_C \text{ is 0 for } A \nleq C,\ C \neq B, \end{aligned} \tag{5}$$

giving $\sum_{A \leq C} t_C = -t_B$.

$$\begin{aligned} c(\tau) &= \sum_{t \in L, A \leq C, t \text{ involves } B} t(\tau)_C, && \text{by definition of } c \\ &= \sum_{t \in L, t \text{ involves } B} \left(\sum_{A \leq C} t(\tau)_C \right), && \text{rearranging the sum,} \\ &= \sum_{t \in L, t \text{ involves } B} -t(\tau)_B, && \text{by (5),} \\ &= - \sum_{t \in L, t \text{ involves } B} t(\tau)_B, && \text{moving the } - \text{ sign outside the sum,} \\ &= - \sum_{t \in L} t(\tau)_B, && \text{as } t(\tau)_B = 0 \text{ if } t \text{ doesn't involve } B, \\ &= -b(\tau), && \text{by definition of the balance of } B. \end{aligned}$$

□

12 Conclusion

This provides the beginnings of a formalized treatment of mathematical bookkeeping, which allow us to describe a bookkeeping system, and prove common accounting results.

We should note that this formalization is far from complete, though. For example, we have made no formal definition of debits and credits. Indeed, everything we've done in this paper can be phrased without mentioning either of those two words. We could introduce this concept by adding a lattice ordering to value spaces, and then prove results about debits and credits.

Similarly, we have made no distinction between different sorts of accounts, so we cannot even state, let alone prove, the accounting equation. Though once we have, it follows easily from the total balance of the balance account being 0.

We have barely touched on the formal definition of time, defining it only as a total order, and glossing over the operations required for continuous time appreciations. What continuity restrictions do we want on transactions, and do we want any completeness restrictions on time?

If we go so far as to encode our stock control in our bookkeeping system, then we can no longer trivially do a business valuation. We could instead introduce valuation functions, to measure the value of our business, as linear maps between value spaces.

Finally, for real world applications, we need to consider how these objects would be represented in computer systems, and what algorithms we would use to manipulate them.

References

[1] Pete L. Clark. Introduction to semigroups and monoids. `http://math.uga.edu/~pete/semigroup.pdf`, 2013.

[2] Many contributors. *Encyclopædia Britannica.* Encyclopædia Britannica, Inc., 1911. `https://en.wikisource.org/wiki/1911_Encyclop%C3%A6dia_Britannica/Book-Keeping`.

[3] Diophantus. *Arithmetica.* Original Publisher Unknown, c. CE 250.

[4] David Ellerman. *Economics, Accounting, and Property Theory.* Lexington Books, 1982. `http://www.ellerman.org/wp-content/uploads/2012/12/EAPT.pdf`.

[5] David Ellerman. On double-entry bookkeeping: The mathematical treatment. 2014. `https://arxiv.org/abs/1407.1898`.

[6] Luca Pacioli. *Summa de arithmetica, geometria, proportioni et proportionalita.* Paganino Paganini, 1494. Original `https://books.google.com/books?id=iqgPe49fhrsC`, English translation (1994) `http://jeremycripps.com/docs/Summa.pdf`.