Z^{Ac}—FORMALISING ACCOUNTING SYSTEMS IN Z

1. The basic systems

1.1. **Single-entry.** Here a transaction leads to an entry in one single account. And the record for this account is just a sequence of transactions, perhaps written as lines in a ledger.

Typically a transaction is given by a date, a description and an amount, which is either added to the money a company owns (i.e. someone pays it money, say or goods or services) or taken from (it pays someone money, say because they buy some raw materials or for running-costs). So, an entry might be

Date	Description	Amount
1st January 2017	Sold Tissot watch	$250.00
3rd March 2017	Bought can of oil	-$40.00

A slightly more refined version might split the "Amount" column into "Income" and "Expenditure"

Date	Description	Income	Expenditure
1st January 2017	Sold Tissot watch	$250.00	
3rd March 2017	Bought can of oil		$40.00

just to make things rather clearer. But there would still be just one account involved: the money we currently hold in the company. The name single-entry comes from the fact that each transaction talks about (the same) single account. (See Ijiri's point about what the "double" in double-entry refers to [].)

This way of running a company (the experts tell us, and the laws allow us) is OK for small, simple companies that just deal in cash (or at least don't deal in credit for itself or customers) and who want to keep a simple record of where they stand in terms of what money the have at any one moment in time.

Once a company wants to start selling things on credit, for example, it needs, though, to have a way of keeping track not just of the cash is has now, but also what cash is still owed to it. That is not available to use at this moment, but it will eventually appear (in a perfect world). This starts to allow a company to predict its future, for example. This also means that when it sells something, cash might not immediately flow into its account (whereas it does for the first transaction in the example above). But in order to keep track of what it is owed, another account needs to be created, so when a customer buys on credit the transaction does not mean money moves into its cash account, but into another account that records other sorts of asset. So, we now have two accounts, and thus we enter the world of double-entry.

Just to anticipate what is coming, we might amend our transactions above further, as in:

Date	Description	Account	Amount
1st January 2017	Sold Tissot watch	Capital	$250.00
3rd March 2017	Bought can of oil	Capital	-$40.00

and even to:

Date	Description	Account	Income	Expenditure
1st January 2017	Sold Tissot watch	Capital	$250.00	
3rd March 2017	Bought can of oil	Capital		$40.00

1.2. **Double (and why, and its mad (well...conventional) naming for credits and debits etc).** In order to keep track of two or more accounts (which give the benefits of being able to deal with credit, loans, invoices etc.) we have to be more sophisticated with our recording methods. This in turn requires that we have more checks in place so that the more complicated system does not get out of hand. Continuing our example from above shows what is needed.

Our transactions above in the single case would, assuming we now have asset accounts (what the company owns, like stock and cash) and liabilities accounts (like some debt owed to a bank due to a loan from the bank, or an invoice comes in from another company for some asset we bought from them) now become something like:

Date	Description	Account	Income	Expenditure
1st January 2017	Sold Tissot watch	Stock		$250.00
		Bank	$250.00	
3rd March 2017	Bought can of oil	Bank		$40.00
		Equipment	$40.00	

and the transaction now has two lines since information about two accounts needs to be recorded.

All the accounts in this expanded example are *asset* accounts, since they are where we hold things we own. (We squashed these three accounts into the Capital account in the previous section.) The main point of this new idea of several accounts, though, is that we can have other sorts of account, so that we can keep track of invoices, say, where we give, essentially, a promise to pay someone for an asset at a later agreed date. So, we don't use an asset to pay for an asset in that transaction, we use a liability (that is, the promise to pay later).

Continuing to make our example more sophisticated, we can introduce a liability account called *Promise to pay* where we record these promises, i.e. invoices sent with goods we receive, to pay later. So our second transaction above now becomes:

At this point the headings on the last two columns start to be misleading. They made sense when there was just one account that held out assets. But now we have

Date	Description	Account	Income	Expenditure
3rd March 2017	Bought can of oil	Promise to pay		$40.00
		Equipment	$40.00	

accounts that hold liabilities, and the whole point of them is that we have not, at this moment, spent anything, so *Expenditure* as a title needs changing. The promise to pay later is an increase in liability, so by convention this is viewed as a credit to that account (our liabilities got bigger). Similarly, if our liabilities decrease, because we pay off a promise, then we have debited our liability account. This suggests the following improved headings:

Date	Description	Account	Debit	Credit
3rd March 2017	Bought can of oil	Promise to pay		$40.00
		Equipment	$40.00	

Well, improved in the sense that the rightmost heading now makes sense. But the one next to it? Hm!

But note that the second line of the transaction is talking about an asset account, Equipment, whereas the first was talking about a liability account, so perhaps that difference explains the rather counter-intuitive name in relation to the second line and the penultimate column?

Underlying the double-entry method is the so-called *accounting equation*, which simply says that[1]

$$assets = liabilities$$

being true defines "the company is keeping its accounts correctly". We have already seen that it makes intuitive sense to say that an increase in liability is a credit in that account. To allow the accounting equation to hold, then, the assets must also increase in this case, but to make the "balancing" in that transaction apparent (when we see a credit we expect to see a matching debit) then we are forced (to make all this work) to call an increase in an asset a debit. Once this is accepted then we can summarise: increases in liabilities are called credits, and decreases in liabilities are called debits; increases in assets are called debits, and decreases in assets are called credits.

The accounting equation holds for all the accounts if we require, for every transaction, that

$$debit = credit$$

So, a global property "the accounting equation holds" is reduced to a local one "debit = credit for this transaction". (Room for some Z-style promotion here, surely!)

This reduction of a global property to a local one (or at least one we can keep track of just by checking each transaction as it happens) probably makes the strange naming convention (which is all it is, we could use any words here) in terms of credit and debit worthwhile.

[1] in fact it's a bit more complicated since the right-hand side also mentions "ownership interest", but here we will just talk about liabilities.

Finally, here are both our transactions with the new naming:

Date	Description	Account	Debit	Credit
1st January 2017	Sold Tissot watch	Stock		$250.00
		Bank	$250.00	
3rd March 2017	Bought can of oil	Promise to pay		$40.00
		Equipment	$40.00	

Note that the first transaction all takes place within asset accounts, but the rules still work given the naming conventions.

1.3. **Triple (useful tool for a business planner).**

1.4. **UTXO and Bitcoin.**

1.5. **Ethereum and beyond.**

2. Building a Z model

2.1. **Single-entry accounting.** We have given sets

$[DATE, TEXT, MONEY]$

and the useful schema

*Transaction*0

$$\begin{array}{l} date : DATE \\ description : TEXT \\ amount : MONEY \end{array}$$

This exactly mirrors a line in the table in Section 1.1.

Note the schema has no predicate part, so we impose no conditions on the transactions. (Should there be any??)

Now the state space for the account is just a sequence of transactions (with an initial state with no transactions):

SingleState

$$account : \text{seq } Transaction0$$

Init

$$\begin{array}{l} SingleState \\ \hline account = \langle\rangle \end{array}$$

Again, this exactly mirrors the table in Section 1.1.

Note this schema also has no predicate part, so we impose no conditions on the account. (Should there be any??)

Then we finally have a single state-changing operation, which itself uses an operation that just sets-up a new transaction (this follows a, simplified, version of the Z idiom of

promotion...done here just because this might be a useful bit of structuring when things get more complicated later..)

*NewTransaction*0

$$
\begin{array}{l}
Transaction0' \\
date? : DATE \\
description? : TEXT \\
amount? : MONEY \\
\hline
date' = date? \\
description' = description? \\
amount' = amount?
\end{array}
$$

$$
\begin{array}{l}
\Delta SingleState \\
Transaction0' \\
\hline
account' = account \frown \langle \theta Transaction0' \rangle
\end{array}
$$

We then promote this local operation (which works on a single transaction) to the whole system ("the accounts") using the following[2]

$$DoSomeBusiness0 \mathrel{\widehat=} \exists\, Transaction0' \bullet\!\wedge\, NewTransaction0$$

We can generalise a bit (as we did in the previous section) to a transaction that tells us what account name the transaction is dealing with. Clearly in the single-entry case, since there is one, single account under consideration, this is always the same (which is why it isn't mentioned in real systems since it is pointless and wasteful to record this redundant piece of information).

So we amend *Transaction*0 to

*Transaction*1

$$
\begin{array}{l}
date : DATE \\
description : TEXT \\
account : ACCOUNT \\
amount : MONEY
\end{array}
$$

and also add the new given type

$$[ACCOUNT]$$

which causes obvious changes to the above Z as follows:

[2]We have used the convention established by some authors make decades ago of using an initial Φ to name the *framing schema* for the promotion (Φ for F in "Framing", eye-catchingly rendered).

*NewTransaction*1

$$
\begin{array}{l}
Transaction1' \\
date? : DATE \\
description? : TEXT \\
account? : ACCOUNT \\
amount? : MONEY \\
\hline
date' = date? \\
description' = description? \\
account' = account? \\
amount' = amount?
\end{array}
$$

$$DoSomeBusiness1 \mathrel{\widehat=} \exists\, Transaction1' \bullet \Phi DoSomeBusiness \land NewTransaction1$$

Finally, again as in the previous section, we introduce the separation of *amount* to give

Transaction

$$
\begin{array}{l}
date : DATE \\
description : TEXT \\
account : ACCOUNT \\
income : MONEY \\
expenditure : MONEY
\end{array}
$$

which flows on to give

NewTransaction

$$
\begin{array}{l}
Transaction' \\
date? : DATE \\
description? : TEXT \\
account? : ACCOUNT \\
income? : MONEY \\
expenditure? : MONEY \\
\hline
date' = date? \\
description' = description? \\
account' = account? \\
income' = income? \\
expenditure' = expenditure?
\end{array}
$$

$$DoSomeBusiness \mathrel{\widehat=} \exists\, Transaction' \bullet \Phi DoSomeBusiness \land NewTransaction$$

(Predicates we might want...keep track of totals amount in and out to make sure it's never below zero? Constrain *account* to always be the same (our only) account name?

Also operations like "what is my total balance?" and "compile a report for the month's income and expenses, in summary".)

2.2. **Double-entry accounting.** Double—some schema calculus with the single version
But also, some refinement.

It looks to me like single can be refined to double.

Very clear from the tables in the first section. The first transaction in the single system

Date	Description	Account	Income	Expenditure
1st January 2017	Sold Tissot watch	Capital	$250.00	

can first be rewritten as

Date	Description	Account	Income	Expenditure
1st January 2017	Sold Tissot watch	Capital		$250.00
		Capital	$250.00	

which is simply related to

Date	Description	Account	Debit	Credit
1st January 2017	Sold Tissot watch	Capital		$250.00
		Capital	$250.00	

So every single transaction and be made more concrete (refined to) a double one. The operations will follow the same pattern and the usual Z refinement properties can be proved (I am sure!)

A

$$a : \text{seq}\, T$$

B

$$b : \text{seq}\, S$$

T

$$a1 : \mathbb{N}$$
$$a2 : \mathbb{N}$$

S

$$b1 : \mathbb{N}$$
$$b2 : \mathbb{N}$$

RTS

$$T$$
$$S$$

$$a1 = b1 + 2$$
$$a2 = b2 + 3$$

RTShideA

$$S$$

$$\exists\, T \bullet RTS$$

RTShideB

$$T$$

$$\exists\, S \bullet RTS$$

R

$$A$$
$$B$$
$$RTShideA$$
$$RTShideB$$

$$\#a = \#b$$
$$\forall\, i : \mathbb{N} \mid i \leq \#a \bullet$$
$$\theta RTShideA = b(i) \land$$
$$\theta RTShideB = a(i)$$

2.3. **Triple-entry accounting (aka Momentum accounting).** Triple—and again?

2.4. **UTXO accounting.** UTXO???????

2.5. **Ethereum-style contracts and accounting????**

3. Rough notes as a plan, some of which is done

Formalising accounts ??????????

?Information used for control settles contracts and commitments and must therefore reside as data shared among the relevant parties? Demski p.160

?Accounting?a science that studies information in an economic context? p.164 Demski 2002.

Transactions are the basis of everything. See Ijiri for example. So that?s where we start.

They have three things observable (at least?might be info about where to find the details in a journal somewhere???):

Type says whether it?s to do with Assets, Liabilities or Owner Inputs

Then there must be exactly two other things observable: amount and where the detail is, for debits; amount and where the detail is, for credits. These two things form the foundation of double-entry book keeping (DEBK) (there are two entries). One question is?.while transitions are the basis of all accounting, clearly not all of them are double-entry?so is this already too specialised? And if so, what is a more general form of transaction? Ijiri would be useful here too.

In fact Ijiri says ?not just that each transaction is being entered in two different accounts, but that transactions are accounted for by two different systems of accounts. ?Double? refers not to the entries, but to the systems of account used to record transactions? (p.28) in Yuji Ijiri, "Momentum Accounting and Triple-Entry Bookkeeping: Exploring the Dynamic Structure of Accounting Measurements", Studies in Accounting Research, volume 31, American Accounting Association, 1989.

Double-Entry forces the identification of causal connections.

The two amounts in these observations MUST be the same and MUST ?cancel out?. This revolves around the table that is derived to make the accounting equation ?work??i.e. there?s a strict convention at the heart of DEBK.

a b

If an Asset Increases Decreases

then a Liability must Decrease Increase

xor

a OI must Decrease Increase

where we have to stick with either column a or column b.

BUT THIS DOES NOT FIT WITH THIS??.!!!!!!!!??.

If a company increases its assets (by buying some property for example) then there has to be the same amount in Liability as an increase too. The Asset amount would be tagged with, say, Property, and the Liability with, say, Loan (to indicate where the money to pay for the property came from). (Or, if an Asset increases then an asset has to decrease?..)

This table follows from the fact that we always want The Accounting Equation to be true. This says

Assets = Liabilites + Owner Input

So if we increase an asset (buy property) then we must balance this (make the equation hold) by increasing either a Liability or have some Owner Input (i.e. they give the company cash, for example).

Refs

?Some thoughts on the Intellectual Foundations of Accounting?, Demski et al., 2002

Acknowledgement

Thanks to Bill Yan for suggesting the name $\mathrm{Z^{Ac}}$.