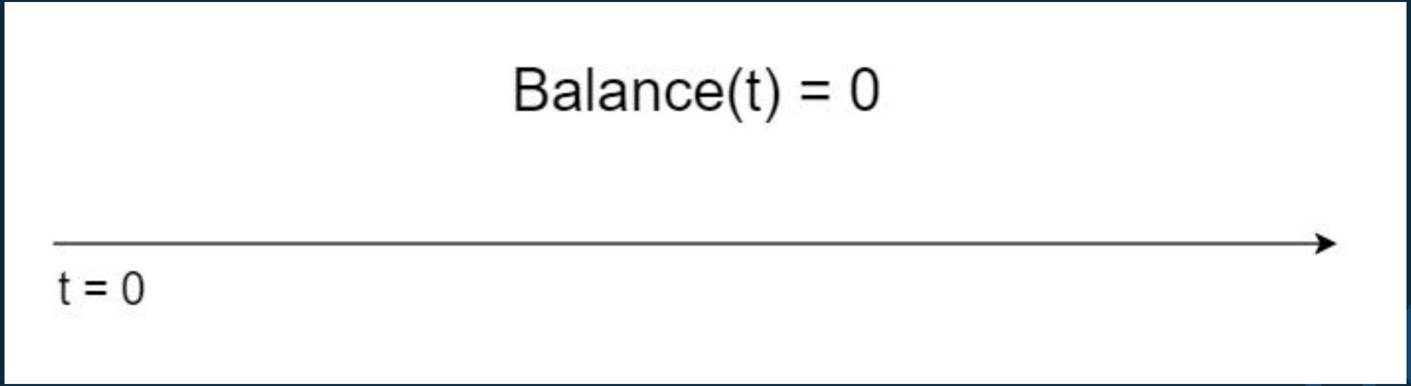
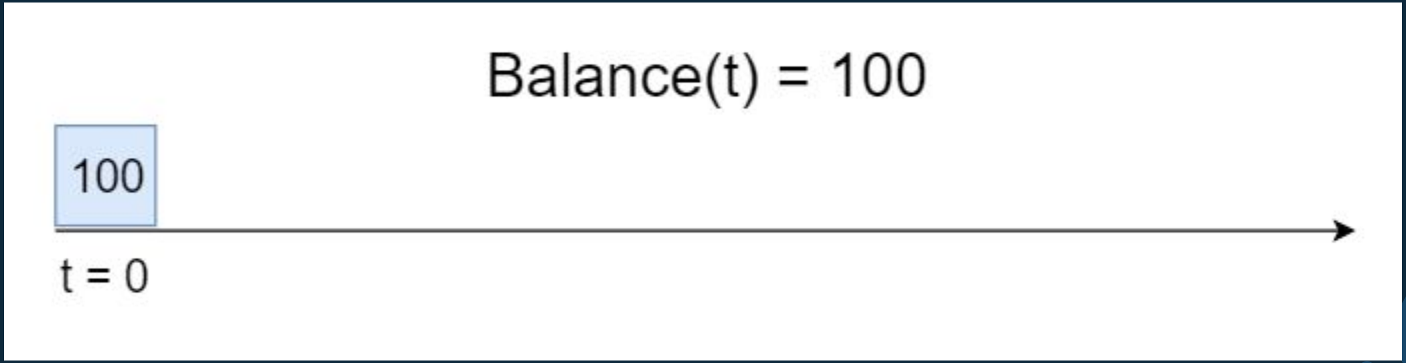
PROJECT AURUM

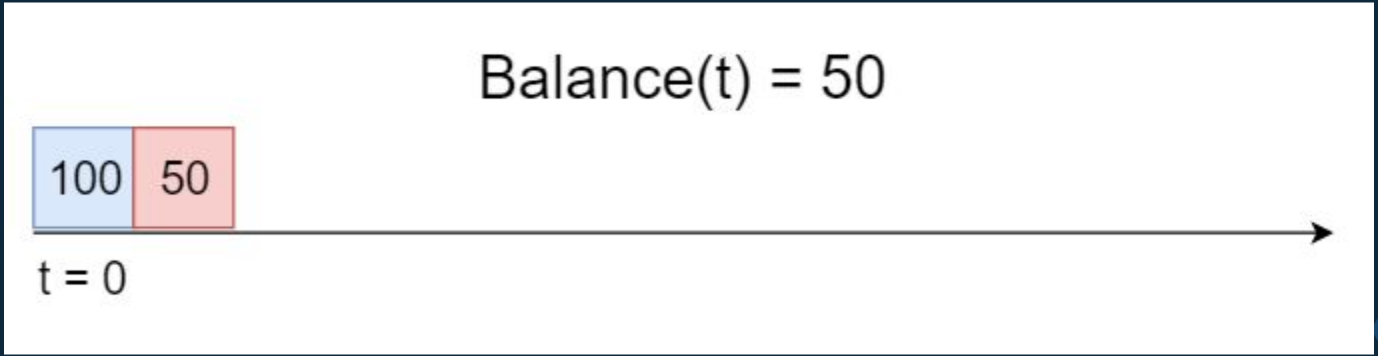
Contracts



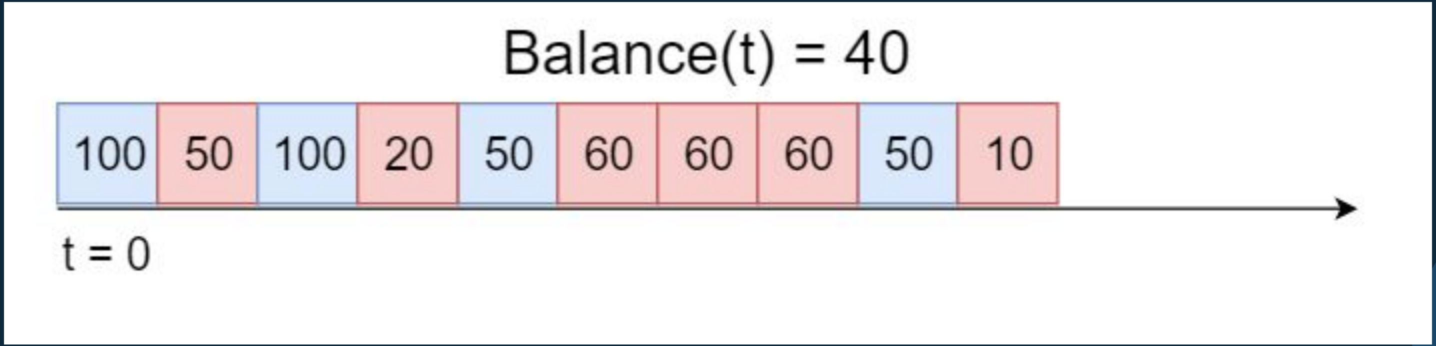
Balance(t) = total balance at time t.
Here, the total balance at time 0 is 0.



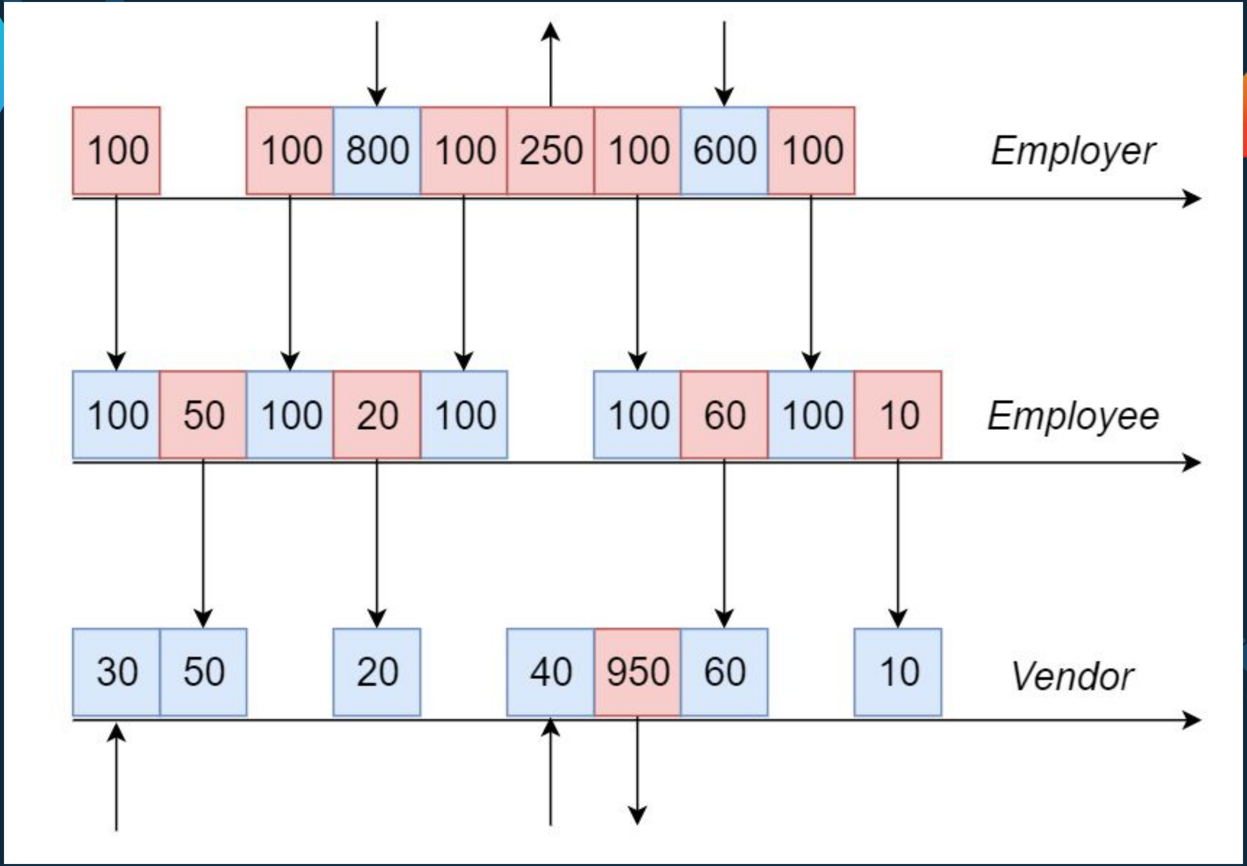
A deposit into the bank account is a positive change to the balance.



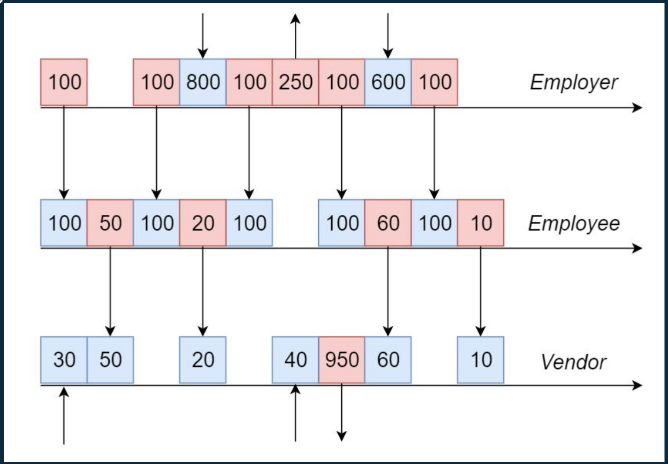
A future withdrawal is a negative change on the balance.

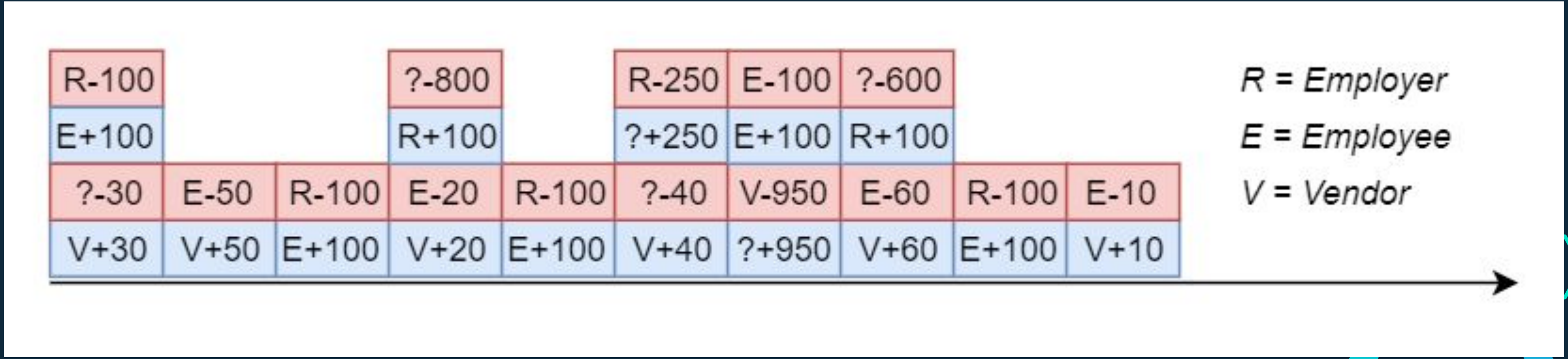


A ledger is chronologically ordered.
Starting t = 0 to t = x, the sequential summation of all transactions is the balance at t =x.
If you think about it, this bank account example is just a collection of withdrawals and deposits.

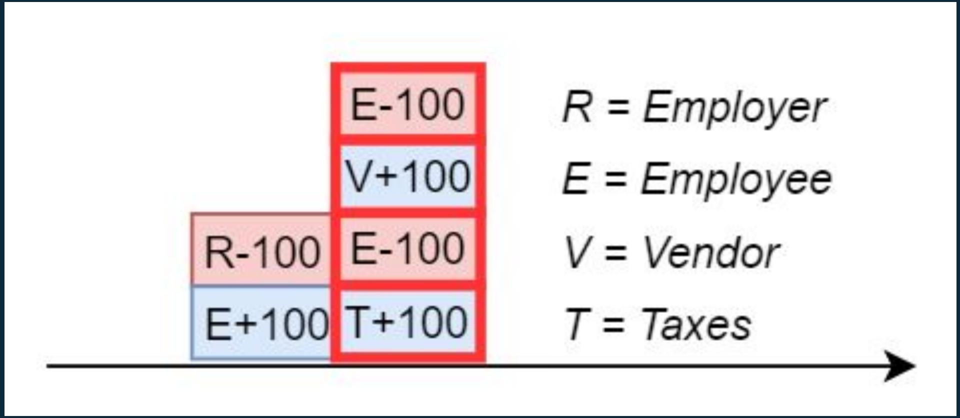


Banks keep track of multiple accounts

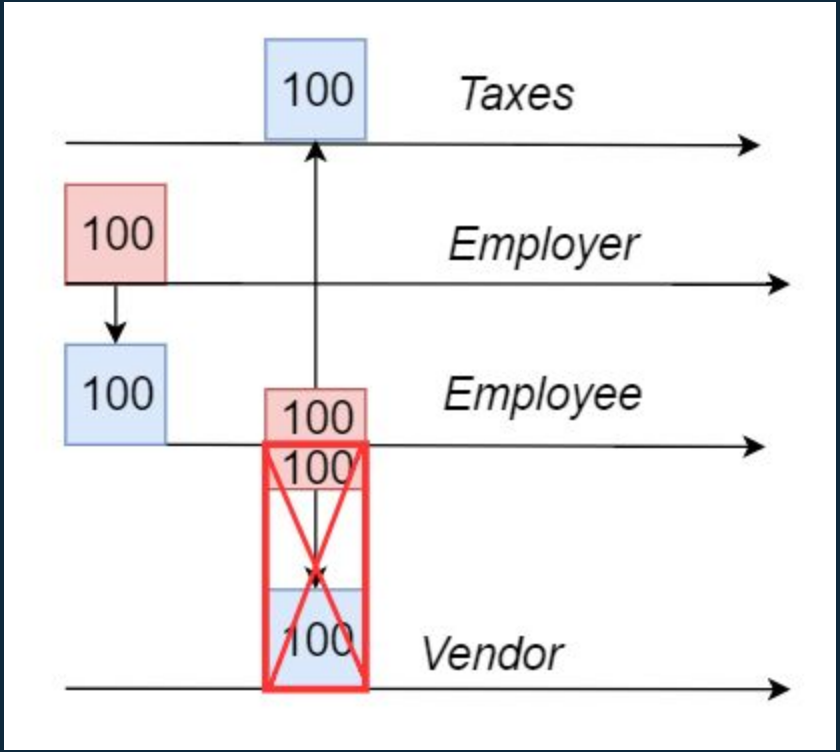

100 100 800 100 250 100 600 100 Employer
100 50 100 20 100 100 60 100 10 Employee
30 50 20 40 950 60 10 Vendor


R-100 ?-800 R-250 E-100 ?-600
E+100 R+100 ?+250 E+100 R+100
?-30 E-50 R-100 E-20 R-100 ?-40 V-950 E-60 R-100 E-10
V+30 V+50 E+100 V+20 E+100 V+40 ?+950 V+60 E+100 V+10
R = Employer
E = Employee
V = Vendor

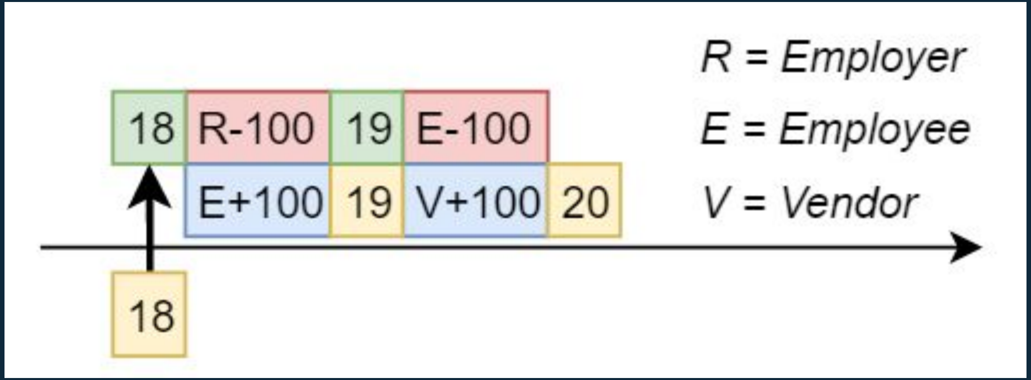
Consolidation into one ledger.



Double Spend problem: *E* makes two withdrawals simultaneously. So which one is the valid one? Technically the system we have so far allows E to double their spending power here.

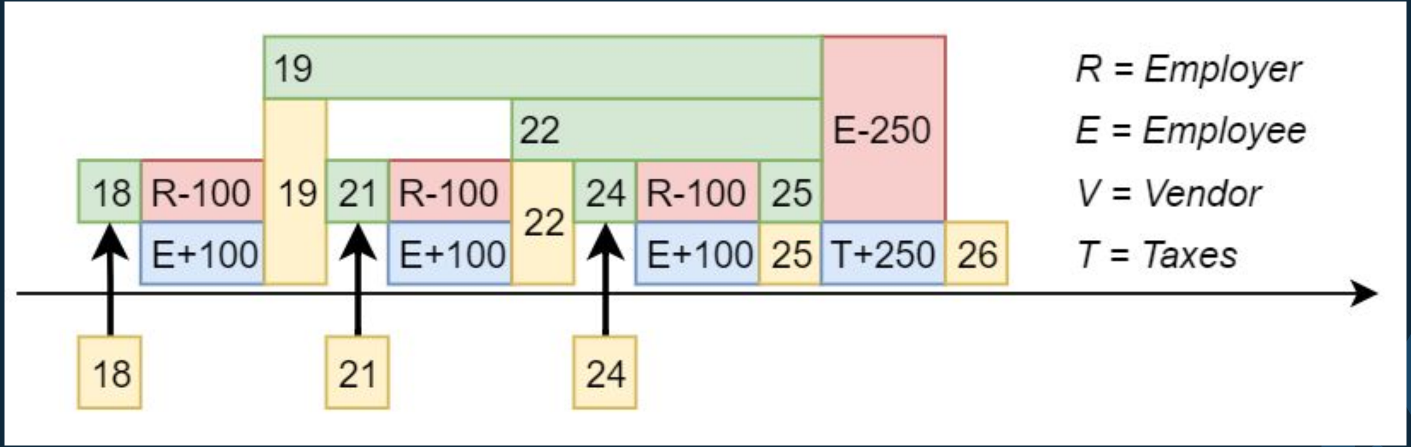


Banks resolve this by picking one withdrawal and processing it before checking to see if the other is valid.

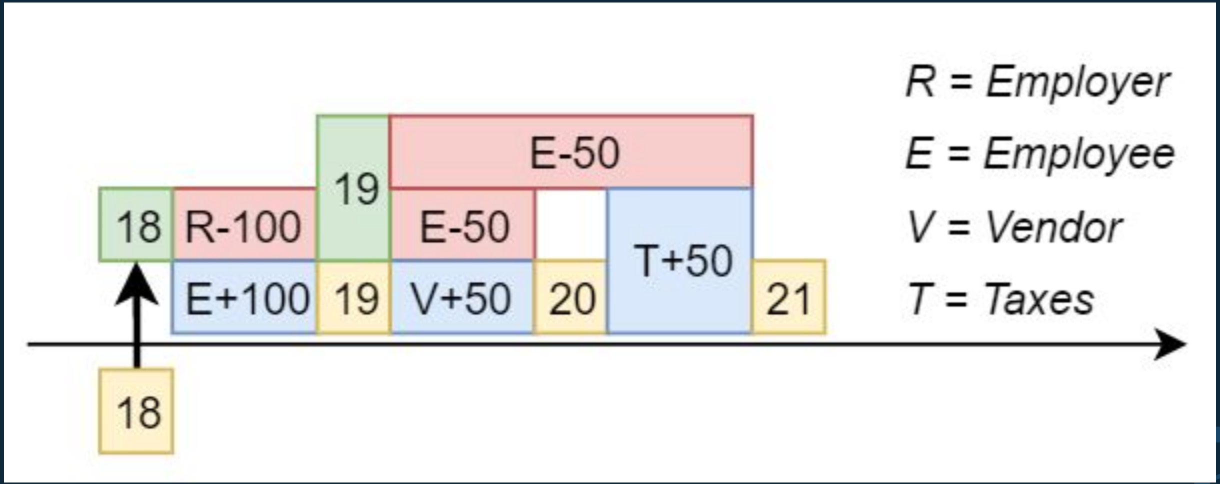


Solution: add reference tags that represent a previous deposit being consumed to fill a withdrawal.

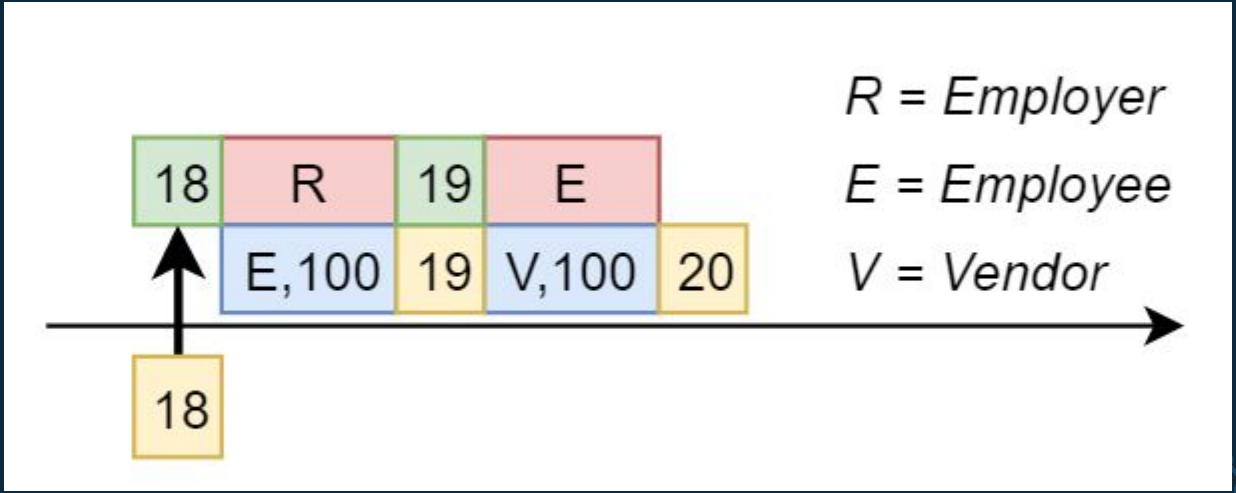
This maintains order and forces every transaction to be uniquely consumed.



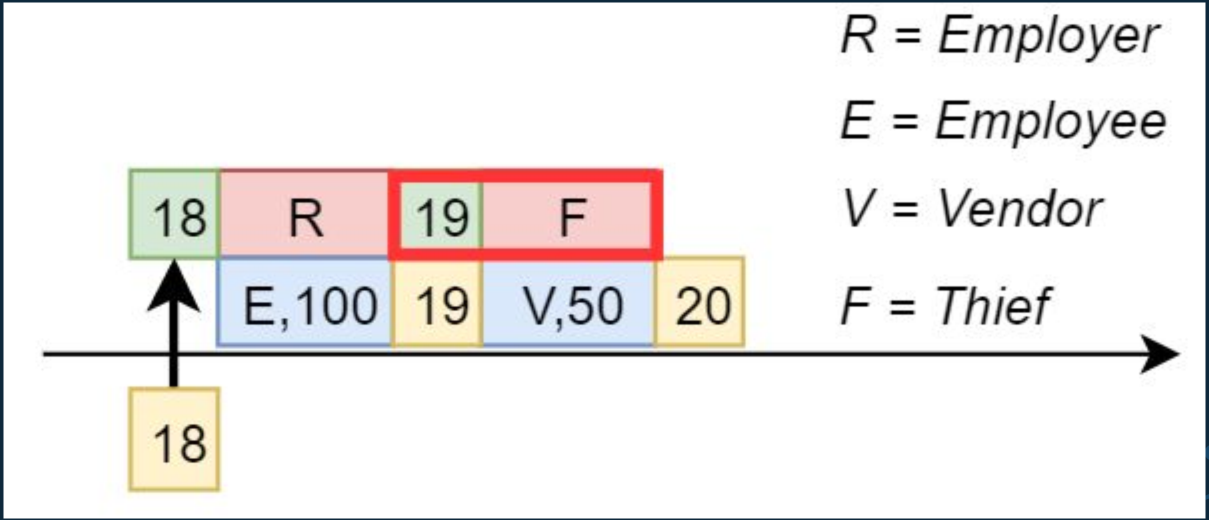
Can consume multiple previous deposits to form one large withdrawal.



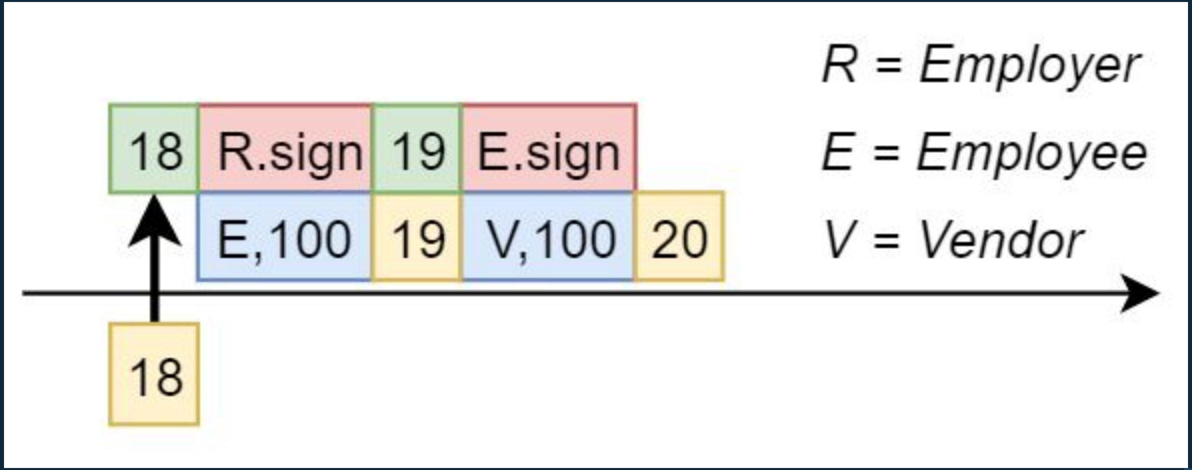
The opposite can also be done; consume a single previous deposit to make multiple previous smaller withdrawals



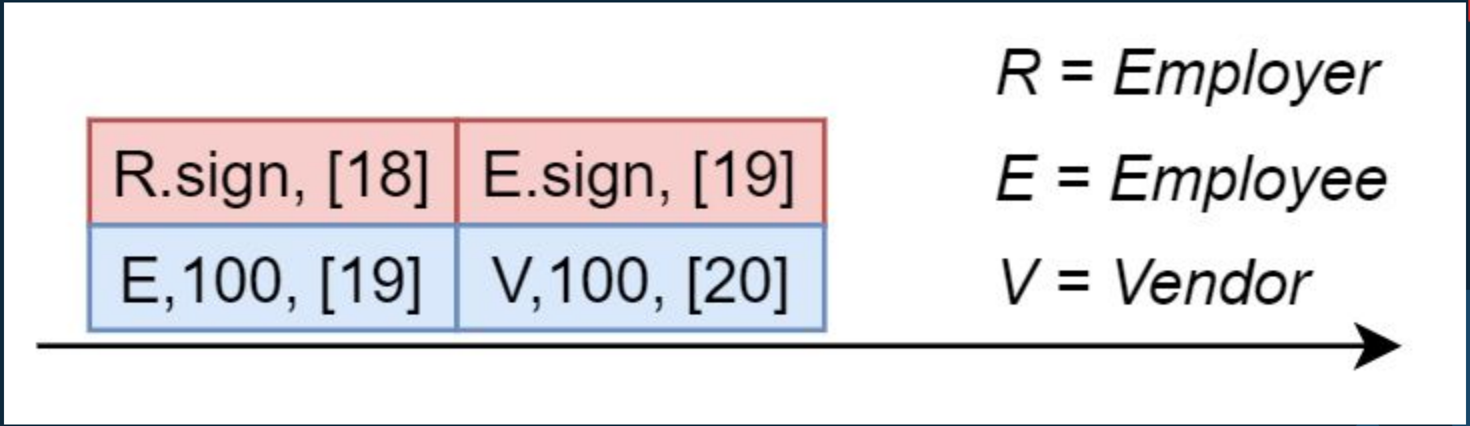
Let's eliminate some redundancy. Blue means "this person can use this amount", and red means "this person is consuming this deposit. We don't need to say explicitly how much they're consuming because we can just figure it out by looking at the records. This is also a means to check to see if they have enough to make the withdrawal.



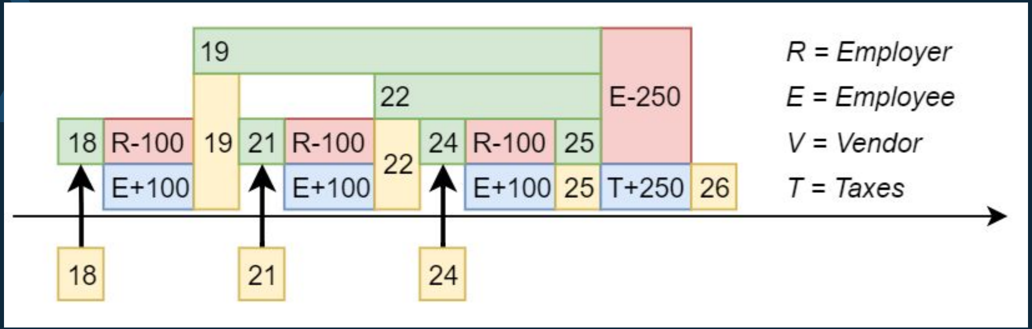
A security issue: how to prevent thieves claiming deposits that don't belong to them? It's not the vendor's fault in this case, so it should be resolved before they can make a withdrawal.

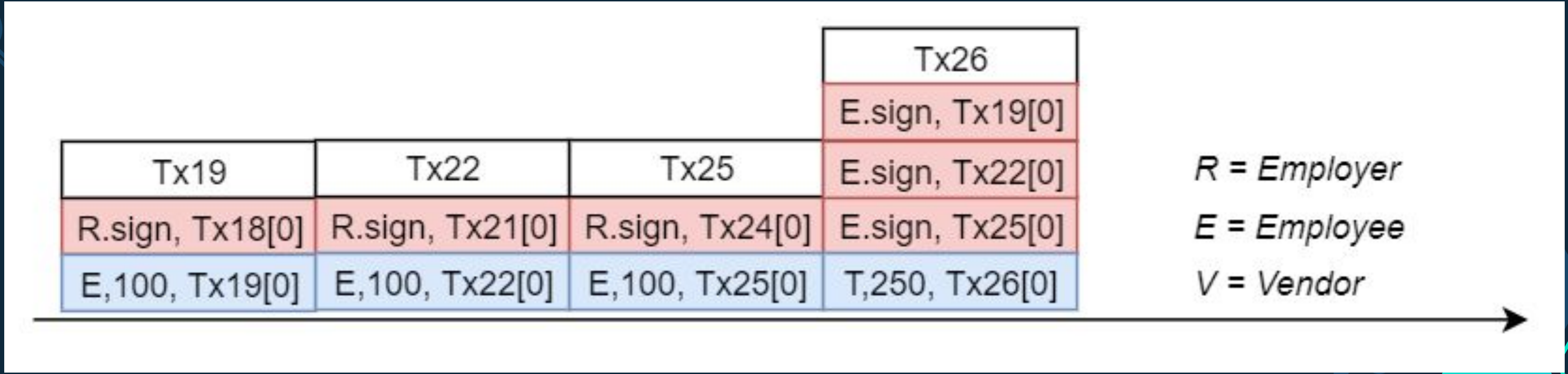


Add a unique signature to every withdrawal.
This will make sure only the person who has a right to that deposit can consume it for withdrawal.

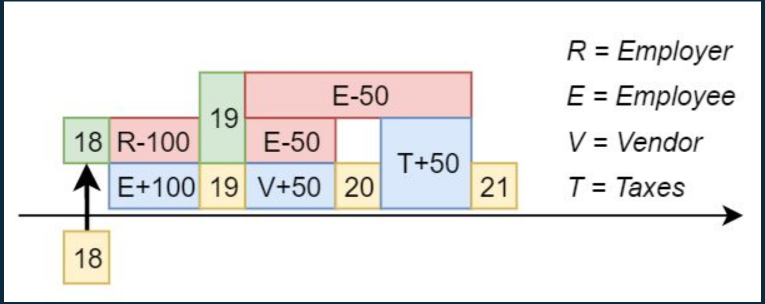


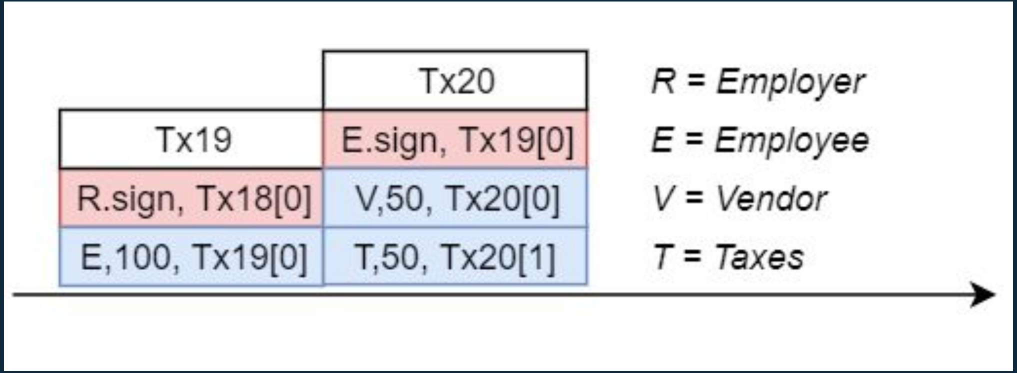
Further consolidation. Red means “With X’s signature, this person consumed this ID’d previous deposit” and blue means “person Y can use this amount, and if they want to consume it, reference the following ID”.



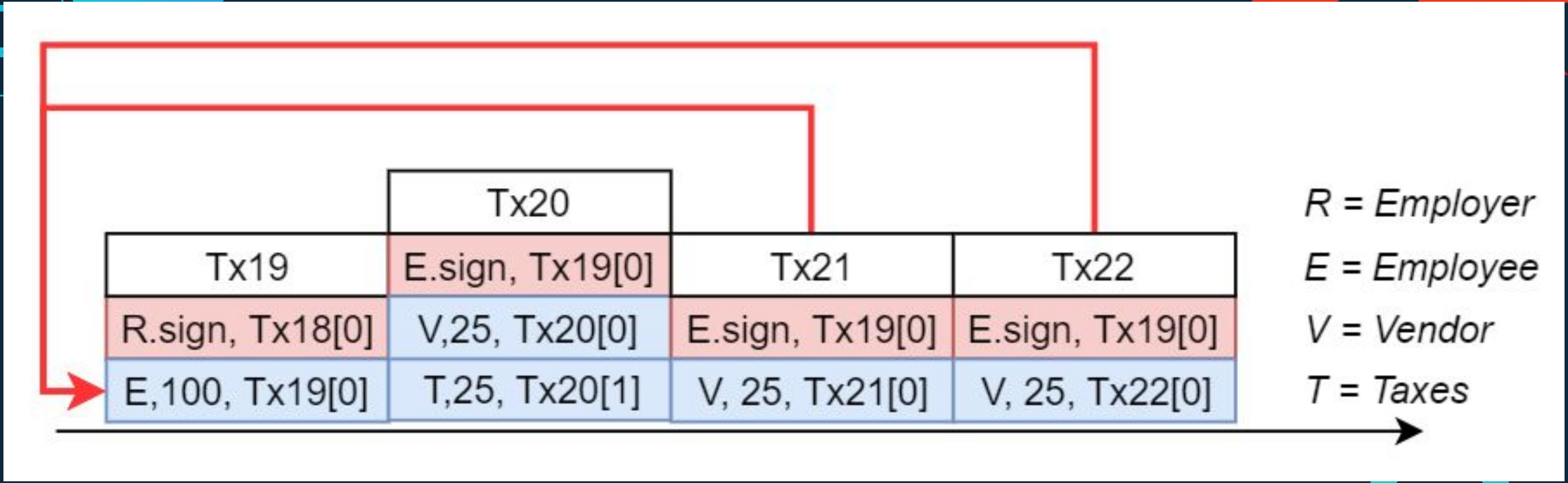


Further simplification. In this scenario, we give each transaction the deposit id. Then we have an array of withdrawals, each referencing arrays of deposits from previous deposits. This allows us to use arrays for transactions involving multiple withdrawals or deposits.

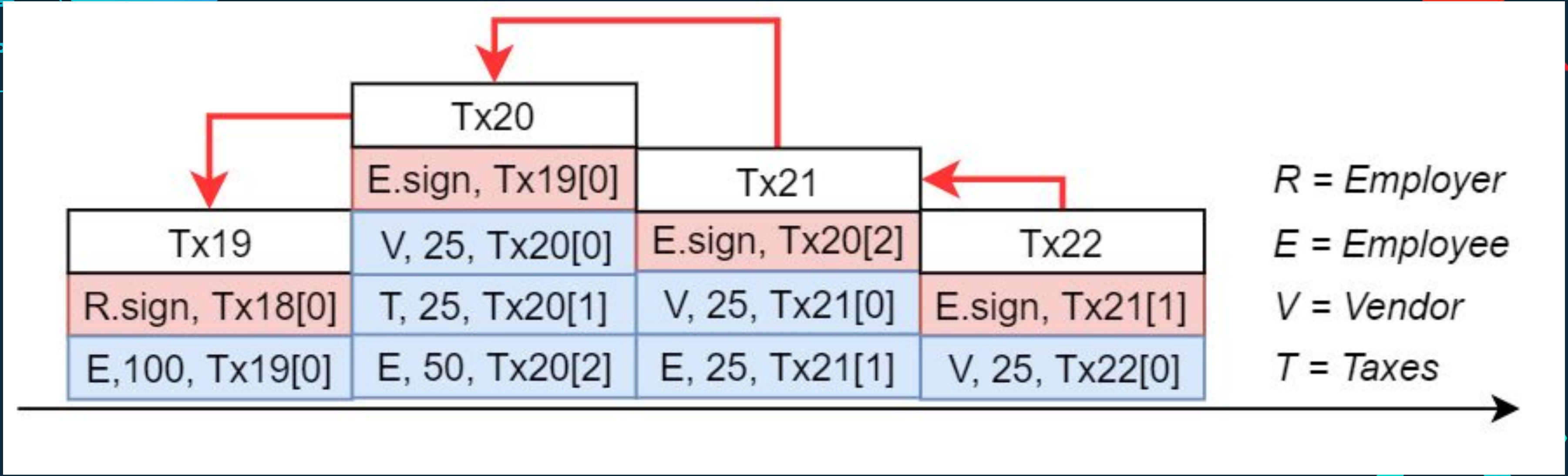




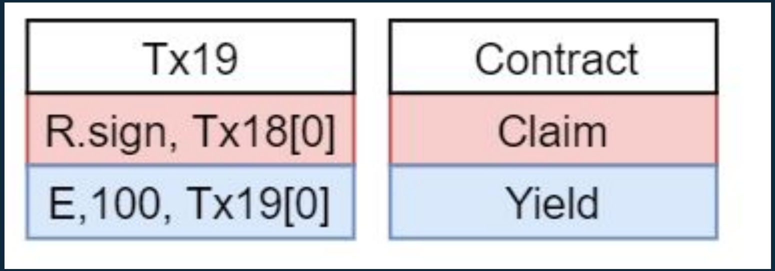
In this scenario we reduce the length of the ledger by incorporating simultaneous deposits into one transaction.



Awkward and inefficient: if we decide to only partially consume a previous deposit over and over, the ledger needs to be checked far in the past each time.



It would help to maintain chronological ordering by introducing the concept of "change". Every time a deposit is claimed, we consume the **whole** deposit, and then make a deposit to *ourselves*.



Tx19	Contract
R.sign, Tx18[0]	Claim
E,100, Tx19[0]	Yield

Since our transaction structure has grown, we're going to assign some *Aurum* terms for them:

- A **claim** is a consumption of a deposit.
  - Signature
  - Target of consumption.
- A **yield** is a deposit yet to be claimed.
  - Authorized claimant
  - Value.
- A **contract** is a transaction. Just because :)