Self-Reductions in Beancount

Martin Blais - April 2017

http://furius.ca/beancount/doc/self-reductions

A description of the difficulties with implementing transactions with postings that reduce one another in the same transaction.

There's a problem with the full booking method. Currently, it works like this (pseudo-code):

> *for each cost-currency group:*
>
> > *book the reducing postings against the ante-inventory*
> >
> > *apply interpolation to fill in missing numbers, convert CostSpec to Cost*

This is problematic, because of the following two conflicting goals:

1. We want to book the reductions against the augmentations within the transaction. This means that if the Assets:Invest inventory is empty, this transaction should succeed:

   ```
   2016-12-04 *
     Assets:Invest     10 HOOL {50 USD}             ;; A
     Assets:Invest    -10 HOOL {50 USD} @ 72.00 USD ;; B
     Assets:Cash      220 USD
     Income:Profit
   ```

   In this transaction, [B] should reduce [A]. Currently, in the new full booking method, it doesn't.

2. When a single number is missing from a currency group, we want to fill it in automatically. For example, this ought to work:

   ```
   2016-12-04 *
     Assets:Invest     10 HOOL {USD}
     Assets:Cash       -500 USD
   ```

   and should resolve the cost to 50 USD. Any single number can be filled in automatically.

   A special case of this, and this is the case that's causing us some problems w.r.t. to the problem described above, is that a reducing leg might be subject to this. For example, if the ante-inventory is this:

   ```
     Assets:Invest     10 HOOL {50 USD, 2016-01-01} ;; A
     Assets:Invest     10 HOOL {51 USD, 2016-01-02} ;; B
   ```

   this transaction should be legal:

```
2016-12-04 *
  Assets:Invest    -5 HOOL {2016-01-02}
  Assets:Cash         USD
```

This would first disambiguate the reduction to the [B] posting using the date, and then interpolate the missing number to figure out that the cash amount would be 5 x 51 = 255. It would result in this:

```
2016-12-04 *
  Assets:Invest    -5 HOOL {51 USD, 2016-01-02}
  AssetCash       255 USD
```

In short, two things occur: the reduction is matched against the available inventory lots, and then the interpolation is run.

Moreover, another thing we'd like to be able to do is infer reduction numbers by interpolation, like this:

```
2016-12-04 *
  Assets:Invest    -5 HOOL {}
  AssetCash       255 USD
```

In this case, this would automatically infer that the cost is 51 USD in order to balance the transaction.

One solution would be to apply the augmentations before we run the reductions. But that's impossible: some of those augmenting postings might have missing numbers which need to be interpolated. It's like a dog chasing its own tail.

I haven't been able to come up with an elegant solution to this problem yet. I need to find some compromise in the meantime.

Here's what we could do:

1. if there's an augmentation and a reduction of the same commodity in the same currency group...

   For each cost-currency group:

   don't allow interpolation, but convert CostSpec to Cost trivially

   apply all the augmentations

   book the reducing postings against the resulting inventory

2. if there aren't:

   For each cost-currency group:

   apply interpolation to fill in missing numbers, converting CostSpec to Cost

   book the reducing postings against the accumulated inventory

Essentially, the entire problems stems from when we decide to update the balances inventory. Do we update just the reductions and later on do the augmentations? Or do we

Another fold of this problem is that whether we choose to update the balance as we categorize the reductions, it may influence the categorization of the postings as reductions or augmentation itself, because this depends on the latest content of the inventory.

I'd like to suggest that the categorization be always based on the ante-inventory in order to be consistent. So we could run the categorization first, then decide which order to do the balance updates as per the above. However, that's not really possible: on an empty directory, adding some shares and remove them would fail, because the removal would be interpreted as a augmentation (of negative values from a zero balance).

Note that the current situation can cause subtle problems. For example, if the ante-inventory is empty, a single transaction which both adds and removes from an account will be interpreted as two augmenting legs, and will succeed, e.g.

```
2016-05-02 *
  Assets:Account            2 HOOL {115.00 USD}
  Assets:Account           -2 HOOL {116.00 USD}
```

This will result in an unfortunately mixed inventory. Fixing this problem should not allow this to occur.