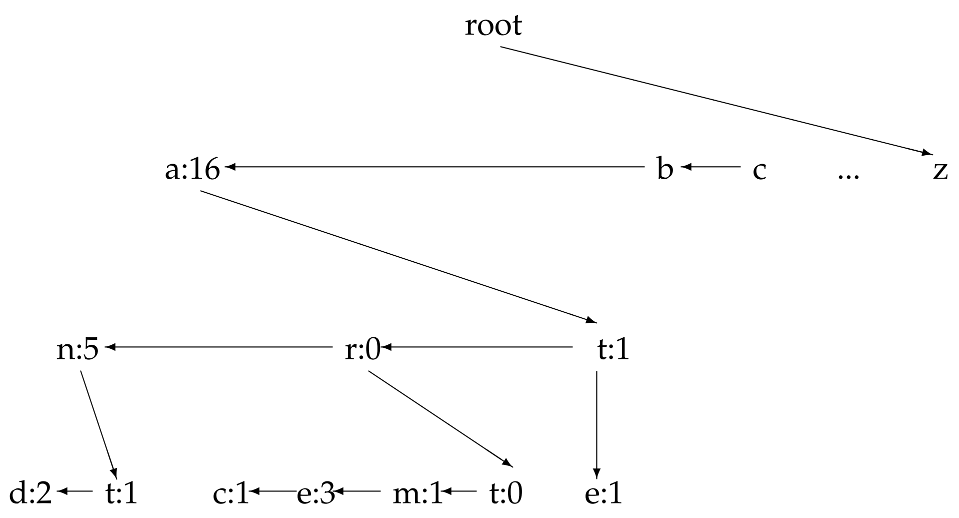

Figure 2: Example plain trie

7.1 The `trie`

For our discussion here, `cellid` is any arbitrary type that has a "sufficient" number of values. We define four tables whose keys (i.e., the first components of its elements) are values from this type. A trie is a subset of `cellids`, and the four tables that collectively satisfy certain constraints. These constraints amount to requiring that the structure we are defining better be a binary tree.

```
~~type cellid;~~
~~value emt: iletter := 1 + max(#upletter, #loletter);~~
~~value hdr: iletter := 0;~~

~~value nilid : cellid := new-cellid();~~
~~value rootid: cellid := new-cellid();~~

~~type trie := table (~~
  ~~cid: cellid,~~
  ~~ltr: iletter,~~
  ~~cnt: nat,~~
  ~~nxt: cellid,~~
  ~~hic: cellid~~
```