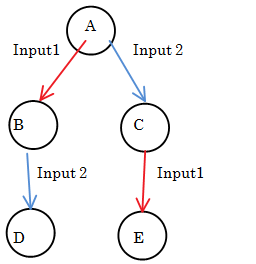
**Definitions of terms:**

States



Output codes:

|  |  |
| --- | --- |
| 0 | no output |
| 1 | output 1 |
| 2 | output 2 |
| 3 | output3 |
| 4 | output 4 |
| 5 | output 5 |
| 6 | any output |
| 7 | disabled |

"Disabled" is allowed to give any output. It is assumed that the state will never be reached. Is this always the same as "any output?"

**Processing FSMs before query:**

* Reassign output numbers so the outputs of states so A always gets priority to be the lowest output number, followed by B, etc.
  + Eg if user sets B=1; A,D=2, reorder so A,D=1; B=2

**Querying:**

* Lookup all equivalent FSMs
  + All states with an output from 0-5 match
* Get all circuits that return those FSMs
* Sort by score
* Return the best circuit

**Expressing a machine's output as a searchable string**

There are 12 genes, each independently expressible in each of five states. For each gene, this means there's a 5-bit character that denotes which state it's expressed in.

Convert the 12 characters for a machine into ASCII, then sort the characters. This should be consistent for a pattern of expression, regardless of the IDs for each gene the user input.

*How do you handle ambiguous states?*

Create a rule that no gene can be expressed only in an ambiguous state

-the number of "0"s at the end of the string must be the same.

For each remaining character, create each combination of off and on in each ambiguous state

-possibly large, but only in edge cases

Alternate option: each state is encoded in twelve bits (numbers 0:4096), corresponding to which genes are expressed. Then you just have to query at most 5 fields, don't query a specific state if it's disabled.

This makes a db of 4096^5 = 1E18 rows. Too big.

You can reduce the size by normalizing the gene ids first, as above