6-inputs networks in base k

Rémy Cerda, Rémi Pellerin, Nicolas Pinson,
Pierre-Etienne Polet and Tristan Stérin

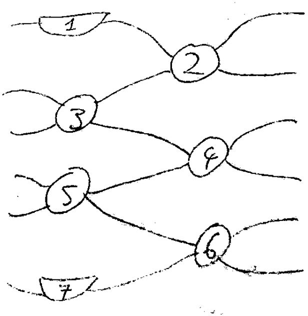
ENS Lyon

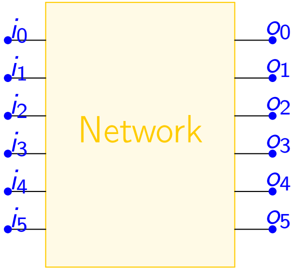
January 20, 2017

ENS DE LYON

Recall the framework...

We will use the following network with 6 inputs and 6 outputs.





3 functions in base 3

We managed to compute 3 "interesting" functions in base 3 over DNA-nanotube network:

3 functions in base 3

We managed to compute 3 "interesting" functions in base 3 over DNA-nanotube network:

- a indicator of 1

3 functions in base 3

We managed to compute 3 "interesting" functions in base 3 over DNA-nanotube network:

- a indicator of 1
- a counter of 1's mod 3

3 functions in base 3

We managed to compute 3 "interesting" functions in base 3 over DNA-nanotube network:

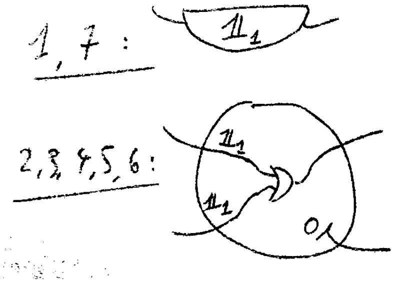
- a indicator of 1
- a counter of 1's mod 3
- a binary additioner (using base 3 network!)

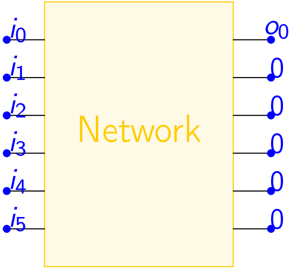
The "1 indicator"

This function outputs $o_0 = 1$ iff there some $i_j = 1$ (others outputs are always 0).

The "1 indicator"

This function outputs $o_0 = 1$ iff there some $i_j = 1$ (others outputs are always 0). To do it, we used these cells:



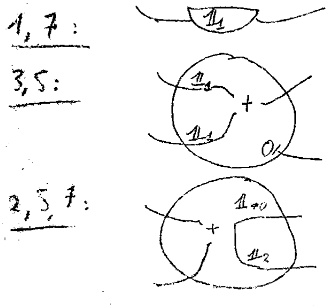


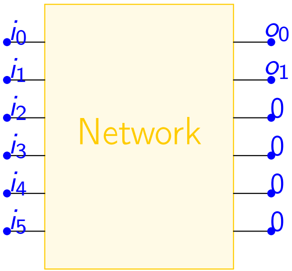
The counter of 1's mod 3

This functions computes the number of ones in the input and sets o_0,o_1 (which must be a fixed point for the function). Other outputs are always 0.

The counter of 1's mod 3

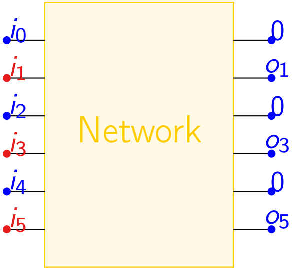
This functions computes the number of ones in the input and sets o_0,o_1 (which must be a fixed point for the function). Other outputs are always 0.



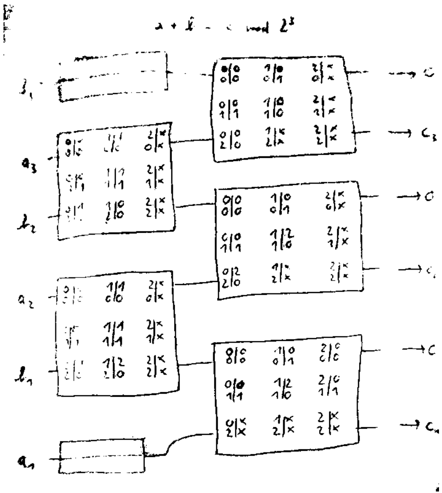


Binary additioner: What it does...

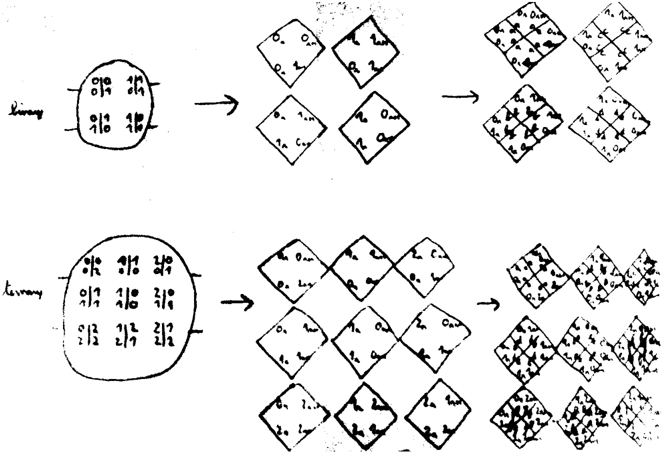
Given 2 binary inputs of length 3 ($i_0 i_2 i_4$ and $i_1 i_3 i_5$), this function computes the sum mod 8 on outputs $o_1 o_3 o_5$ (other outputs are set to 0).



... and how it works

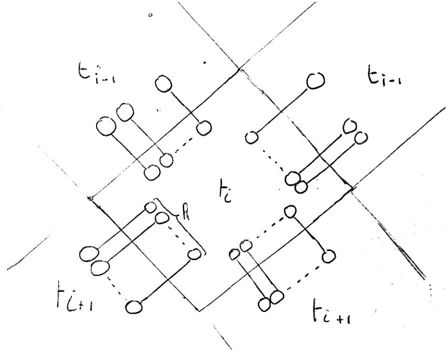


Compiling the network into tiles



Represent circuit as a graph.

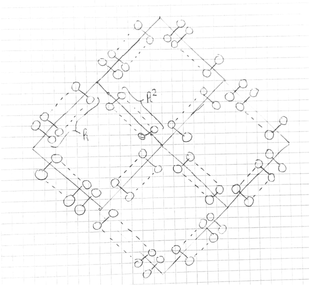
We have 7 possible positions and we have k^2 possible tile choice for each.



Each time a tile can be placed next to an other their colors become linked.

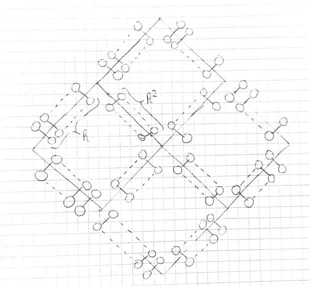
Convert tiles into proofreading tiles

We add 8 new colors per tile and all the previous colors were replaced by two new colors.



Convert tiles into proofreading tiles

We add 8 new colors per tile and all the previous colors were replaced by two new colors.

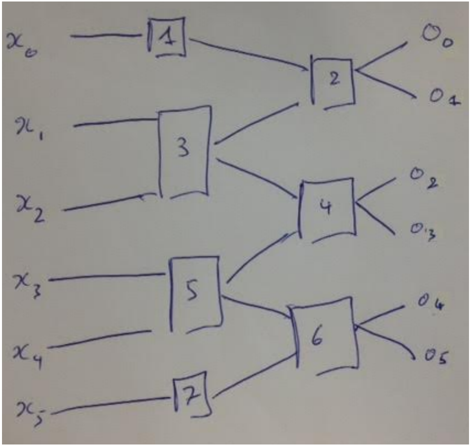


Each color will be associated to a DNA-string of size fixed L. Each tile will be the concatenation of it's 4 colors.

Does a 6 bit counter exists?

Can we create a **deterministic** network that will iterate over all 6 bits string — in any order?

Does a 6 bit counter exists ?



Can we bruteforce that problem ?

There are $\mathbf{2^{44}}$ differents networks which is approximatively 17000 billions.
We could hope for an answer in a few days.
But we can drastically restrict our search space with a **few** observations.

Can we bruteforce that problem ?

There are $\mathbf{2^{44}}$ differents networks which is approximatively 17000 billions.
We could hope for an answer in a few days.
But we can drastically restrict our search space with a **few** observations.

Main ideas :

- Get rid of **redundancy**
- Look at interesting **sub-networks**

6bit Networks are HUGELY redundant

There are $\mathbf{2^{44}}$ differents networks which is approximatively 17000 billions.

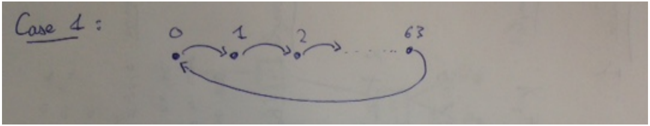
But these networks allow to compute only about **32 billions** different layer functions.

So there is only about **0.2%** interesting networks in the sense that they compute distinct layer functions.

This model allows to calculate on a layer only **8.2e-104 %** of all functions of $\{0, 1\}^6 \to \{0, 1\}^6$ (there are 64^{64}).

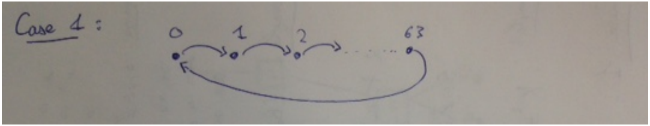
How would such a network's dynamic look like ?

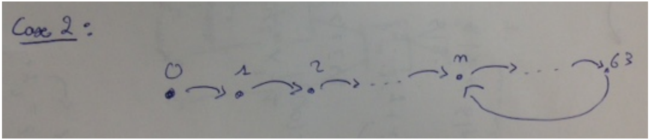
There are only two possibilities (up to permutation):



How would such a network's dynamic look like ?

There are only two possibilities (up to permutation):





Implication on the layer function of the network

The function that our counting network coumputes on one layer is either:

- **A bijection**
- **An almost-bijection**

Implication on the layer function of the network

The function that our counting network coumputes on one layer is either:

- **A bijection**
- **An almost-bijection** i.e. f such that $\exists! x_0, y_0 \in \{0,1\}^6$ with f' being a bijection and:

$$\forall x \neq x_0 \quad f'(x) = f(x)$$
$$f'(x_0) = y_0$$

Implication on the layer function of the network

The function that our counting network coumputes on one layer is either:

- **A bijection**
- **An almost-bijection** i.e. f such that $\exists! x_0, y_0 \in \{0,1\}^6$ with f' being a bijection and:

$$\forall x \neq x_0 \quad f'(x) = f(x)$$
$$f'(x_0) = y_0$$

We are going to check on both cases.

The $4 * 16$ property

If we have a bijection it will **enumerate** all strings of $\{0, 1\}^6$. After reordering it will look like:

```
0000000000000000000000000000000011111111111111111111111111111111
0000000000000000111111111111111100000000000000001111111111111111
0000000011111111000000001111111100000000111111110000000011111111
0000111100001111000011110000111100001111000011110000111100001111
0011001100110011001100110011001100110011001100110011001100110011
0101010101010101010101010101010101010101010101010101010101010101
```

The $4 * 16$ property

Let's focus on the first two bits, we can organise our sequences this way:

0000000000000000	0000000000000000	1111111111111111	1111111111111111
0000000000000000	1111111111111111	0000000000000000	1111111111111111
0000000011111111	0000000011111111	0000000011111111	0000000011111111
0000111100001111	0000111100001111	0000111100001111	0000111100001111
0011001100110011	0011001100110011	0011001100110011	0011001100110011
0101010101010101	0101010101010101	0101010101010101	0101010101010101

The $4 * 16$ property

Let us get rid of the last 4 bits:

0000000000000000	0000000000000000	1111111111111111	1111111111111111
0000000000000000	1111111111111111	0000000000000000	1111111111111111

The $4 * 16$ property

Let us get rid of the last 4 bits:

0000000000000000	0000000000000000	1111111111111111	1111111111111111
0000000000000000	1111111111111111	0000000000000000	1111111111111111

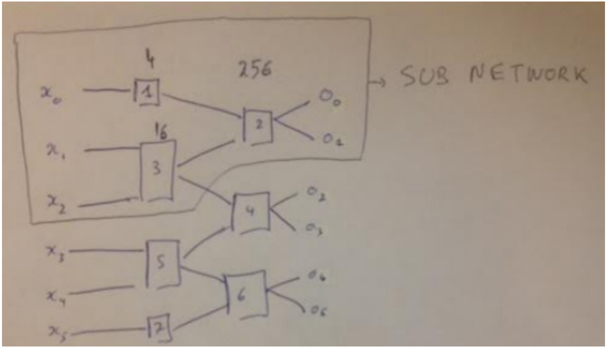
We see that each of the 2 bits patterns: **00, 01, 10, 11** occurs **16** times in this enumeration.
It's the **$4 * 16$ property**.

The $4 * 16$ property

Hence the sub network responsible for these 2 bits must have the **4*16** property to be eligible as a being a sub network of a 6 bits bijective counter network.

The $4 * 16$ property

Hence the sub network responsible for these 2 bits must have the **4*16** property to be eligible as a being a sub network of a 6 bits bijective counter network.

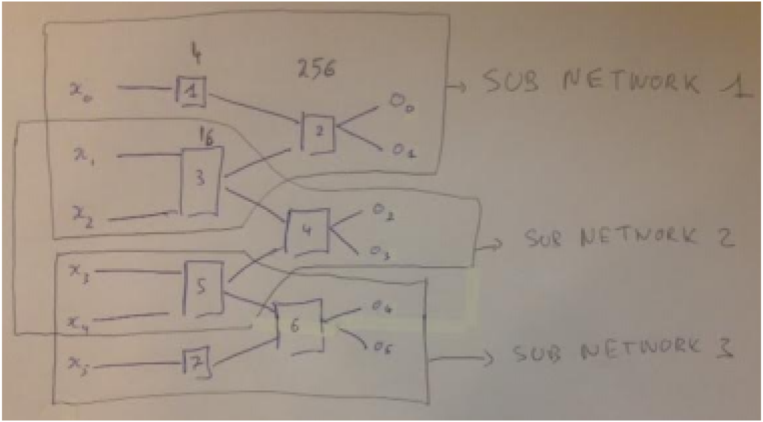


Compute all the $4 * 16$ sub networks

By exausthive search we find **288** (over $4 * 16 * 256 = 16384$) sub networks with this property.

By removing equivalent networks we are left with **72 circuits for the first 2 bits**.

4*16 holds for middle and ending bits



4*16 holds for middle and ending bits

- **72** networks for the first 2 bits
- **216** networks for the middle 2 bits
- **72** networks for the last 2 bits

How to conclude? — 1. Combine it!

Hence by combining these we have $72 * 216 * 72 \simeq 10^6$ 6 bits networks to test.

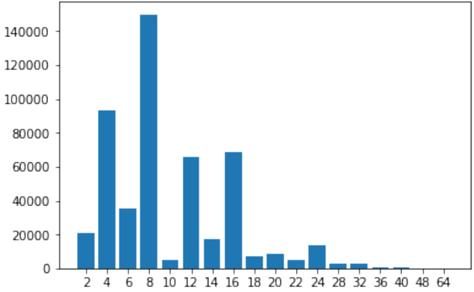
How to conclude? — 2. Count orbits!

Over all these potential networks we count **497664** bijections.
For each of these bijections we have to **count their orbits**, we have a winner **iff it has only 1 orbit**.

How to conclude? — 2. Count orbits!

Over all these potential networks we count **497664** bijections. For each of these bijections we have to **count their orbits**, we have a winner **iff it has only 1 orbit**.

We do not find such a network, here there's the histogram of orbits:



Conclusion

Conclusion: There are no bijective 6bits counters.

2*161517 property

If we have an almost-bijection **all 6 bits sequences will be reached by our network but one**.

2*161517 property

If we have an almost-bijection **all 6 bits sequences will be reached by our network but one**.

Also one bit string will be reached **twice**.

2*161517 property

If we have an almost-bijection **all 6 bits sequences will be reached by our network but one**.

Also one bit string will be reached **twice**.

It means that **at least one** of our sub network will see:

- 2 patterns **16** times
- 1 pattern **15** times
- 1 pattern **17** times

2*161517 property

If we have an almost-bijection **all 6 bits sequences will be reached by our network but one**.

Also one bit string will be reached **twice**.

It means that **at least one** of our sub network will see:

- 2 patterns **16** times
- 1 pattern **15** times
- 1 pattern **17** times

We call this the **2*161517 property**.

2*161517 does not occur!

By enumeration, **there exist no sub network with the 2*161517** property.
Hence, we cannot hope for an **almost-bijective counter**.

No 6bit counter network :'(

By case distinction, **there's no 6bit counter network**.

No 6bit counter network :'(

By case distinction, **there's no 6bit counter network**.

But there are **0 to 62** counters and this kind of methods helps to exhibit **at least 24**.

Perspectives

- The fact that no sub-network has the 2x161517 property helps to find an argument for a **formal proof** in the almost-bijective case.
- We saw that our network model was hugely redundant. To make formal proof it would nice if we could find a **canonical** form for each network.

Sources

All our sources for these computations are available here:
`https://github.com/cosmo-sterin/ER_MolProg_Project2/tree/master/circuit_sim`