A universal spiking neural P system with 11 neurons

Turlough Neary

Boole Centre for Research in Informatics,
University College Cork, Ireland,

Funded by Science Foundation Ireland Research
Frontiers Programme grant number 07/RFP/CSMF641.

August 24, 2010

Introduction

- For many years there has been much interest in the search for simple universal models of computation (e.g. Turing machines, Post systems, cellular automata and many P system variants).
- In this talk we continue the search for the smallest universal spiking neural P system (SN P system) where size is the number of neurons.

Small universal SN P systems

number of neurons	simulation time/space	author
4	exponential	Neary 2010
84	double-exponential	Păun & Păun 2007
67	double-exponential	Zhang et al. 2008
17	exponential	Neary 2009
11	**exponential**	**new result**

Table: The "simulation time/space" column gives the overheads used by each system when simulating a standard single tape Turing machine. Red indicates extended rules, and black indicates standard SN P systems.

Small universal SN P systems and input encodings

number of neurons	simulation time/space	author
4	exponential	Neary 2010†
84	double-exponential	Păun & Păun 2007†
67	double-exponential	Zhang et al. 2008†
17	exponential	Neary 2010†
11	**exponential**	**new result⋆**

Table: The "simulation time/space" column gives the overheads used by each system when simulating a standard single tape Turing machine. Red indicates extended rules and black indicates standard SN P systems.

Given a function f encoded as the natural number y and its input x the SN P systems labeled with † take the input sequence $10^{x-1}10^{y-1}1$ and the system labeled with $\star$ takes the input $10^{4hx}10^{4hy}1$.

SN P systems and time/space complexity

- Many small universal SN P systems simulate Korec's simple universal counter machine.
- The SN P systems that simulate Korec's counter machines suffer from double-exponential time and space overheads when simulating Turing machines.

$$\text{universal SN P system} \to \text{Korec's simple machine} \to \text{R3A-machines} \to \text{2-counter machines} \to \text{Turing machines}$$

where $A \to B$ denotes that A simulates B.

Counter machine definition

A counter machine is a tuple $C = (z, R, c_m, Q, q_1, q_h)$, where

- z gives the number of counters,
- R is the set of input counters,
- c_m is the output counter,
- $Q = \{q_1, q_2, \cdots, q_h\}$ is the set of instructions,
- $q_1, q_h \in Q$ are the initial and halt instructions, respectively.

Counter machine operation

Each counter c_j stores a natural number value $x \geqslant 0$ and each instruction q_i is of one of the following two forms:

- $q_i : INC(j), q_l$
- $q_i : DEC(j), q_l, q_k$

Counter machine operation

Each counter c_j stores a natural number value $x \geqslant 0$ and each instruction q_i is of one of the following two forms:

- $q_i : INC(j), q_l$ increment the value x stored in counter c_j by 1 and move to instruction q_l.
- $q_i : DEC(j), q_l, q_k$

.

Counter machine operation

Each counter c_j stores a natural number value $x \geqslant 0$ and each instruction q_i is of one of the following two forms:

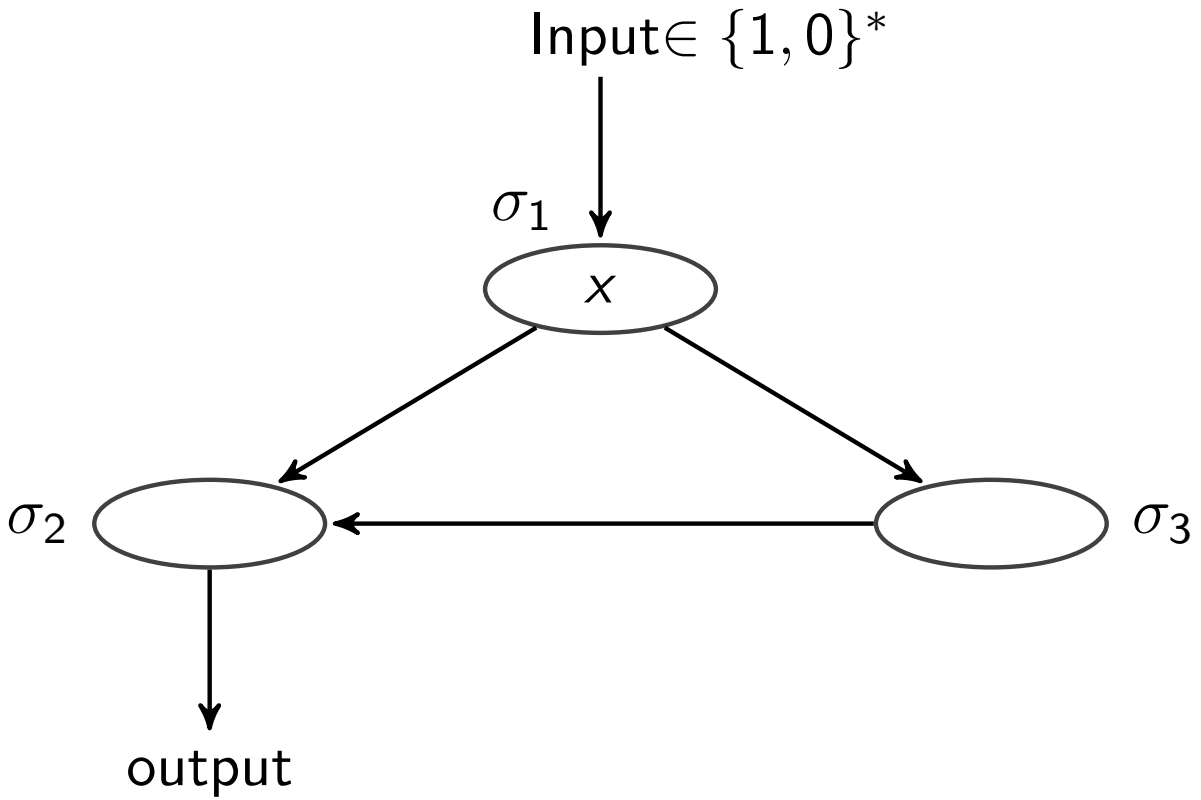
- $q_i : INC(j), q_l$ increment the value x stored in counter c_j by 1 and move to instruction q_l.
- $q_i : DEC(j), q_l, q_k$ if $x > 0$ then decrement c_j by 1 and move to instruction q_l, otherwise if $x = 0$ move to instruction q_k.

SN P system

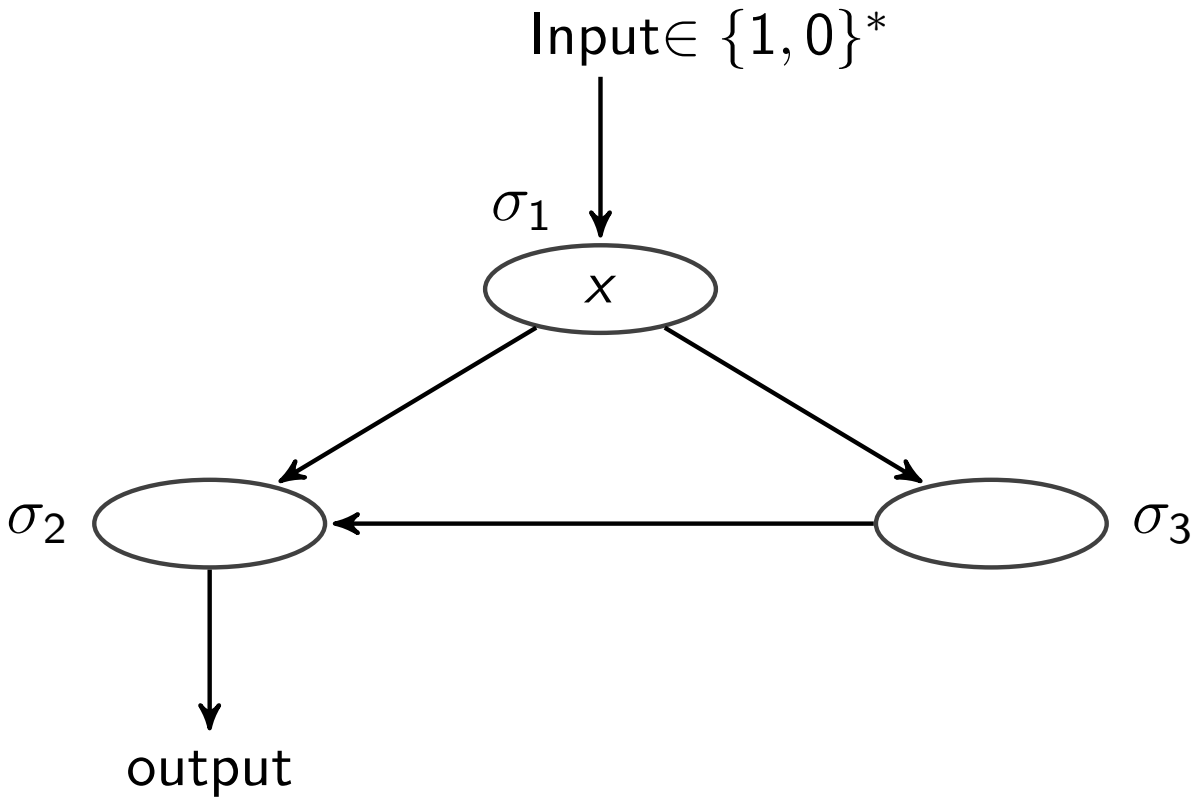
An SN P system is a tuple $\Pi = (O, \sigma_1, \sigma_2, \cdots, \sigma_m, syn, in, out)$, where:

1. $O = \{s\}$ is the unary alphabet (s is known as a spike),
2. $\sigma_1, \sigma_2, \cdots, \sigma_m$ are neurons, of the form $\sigma_i = (n_i, R_i), 1 \leqslant i \leqslant m$, where:
   - 2.1 $n_i \geqslant 0$ is the initial number of spikes contained in σ_i,
   - 2.2 R_i is a finite set of rules of the following two forms:
     - 2.2.1 $E/s^b \rightarrow s; d$, where E is a regular expression over s, $b \geqslant 1$ and $d \geqslant 0$,
     - 2.2.2 $s^e \rightarrow \lambda$, where λ is the empty word,
3. $syn \subseteq \{1, 2, \cdots, m\} \times \{1, 2, \cdots, m\}$ is the set of synapses between neurons, where $i \neq j$ for all $(i, j) \in syn$,

An SN P system executing rule $E/s^b \rightarrow s$

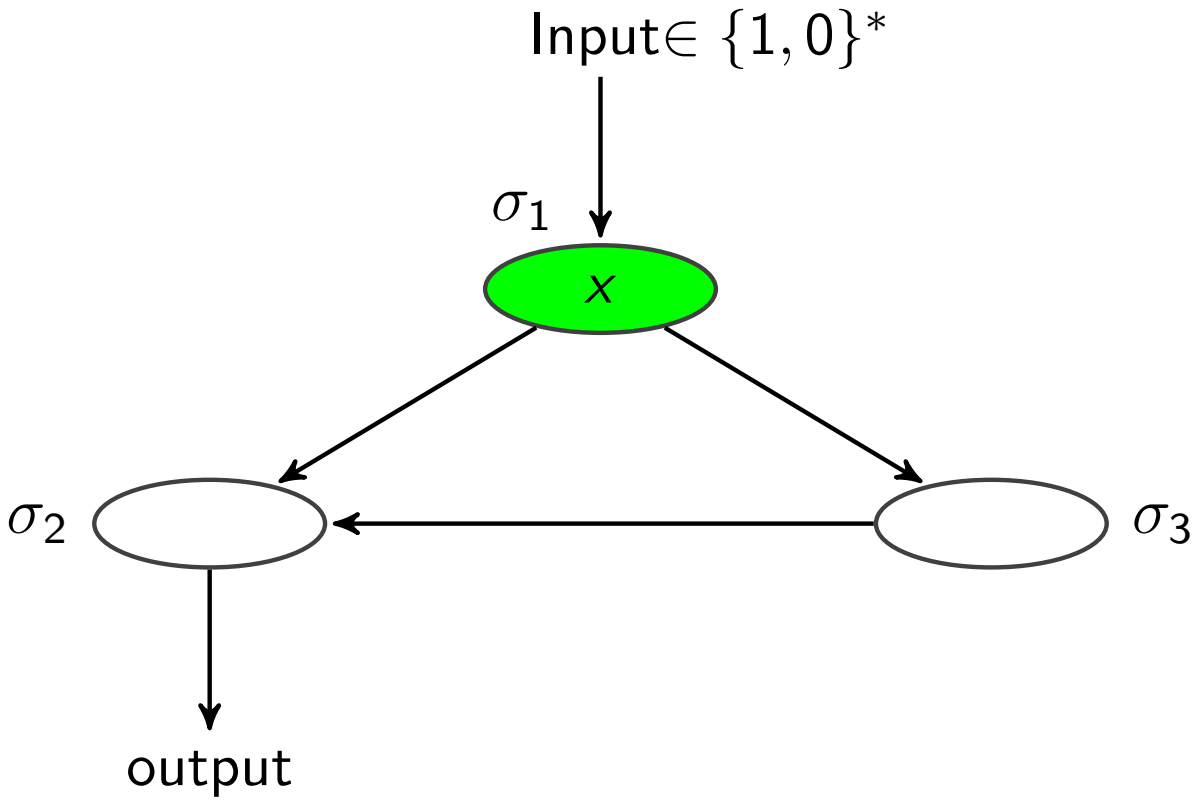


An SN P system executing rule $E/s^b \to s$



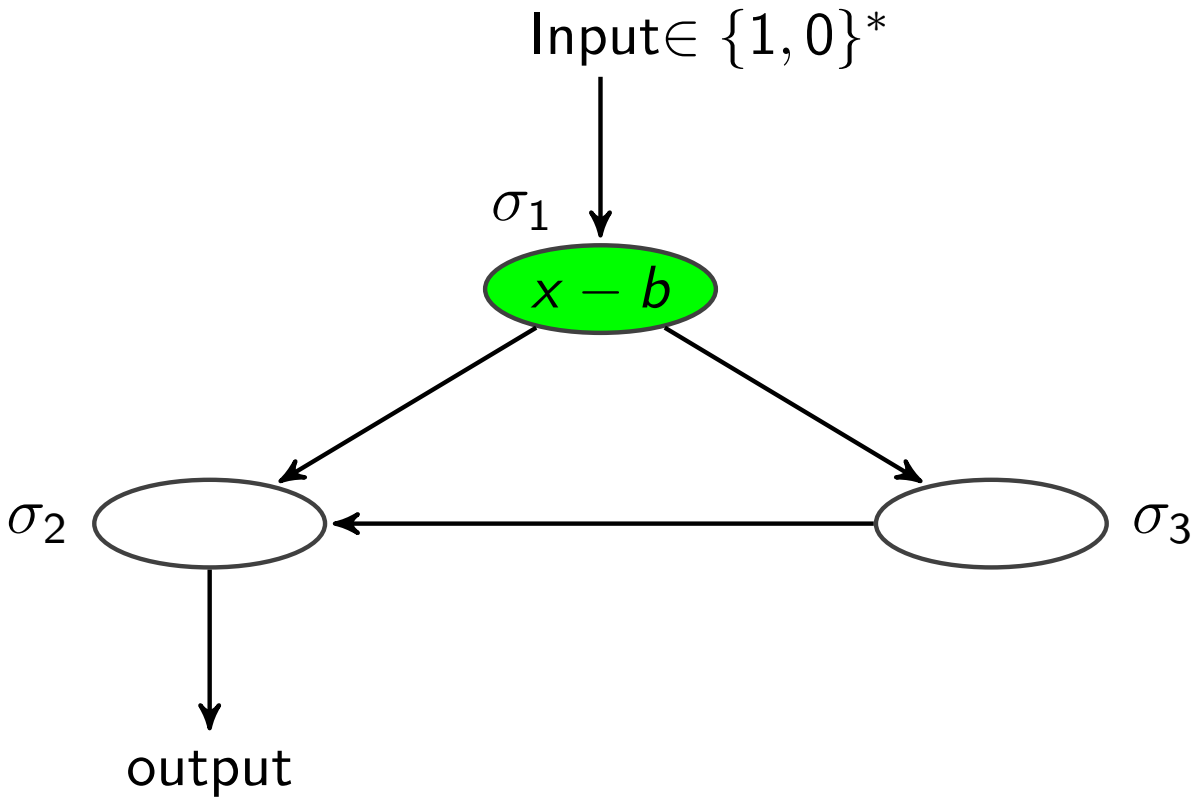
If $s^x \in L(E)$ ($L(E)$ is the language given by E) and $x > b$, then we execute the firing rule $E/s^b \to s$.

An SN P system executing rule $E/s^b \rightarrow s$



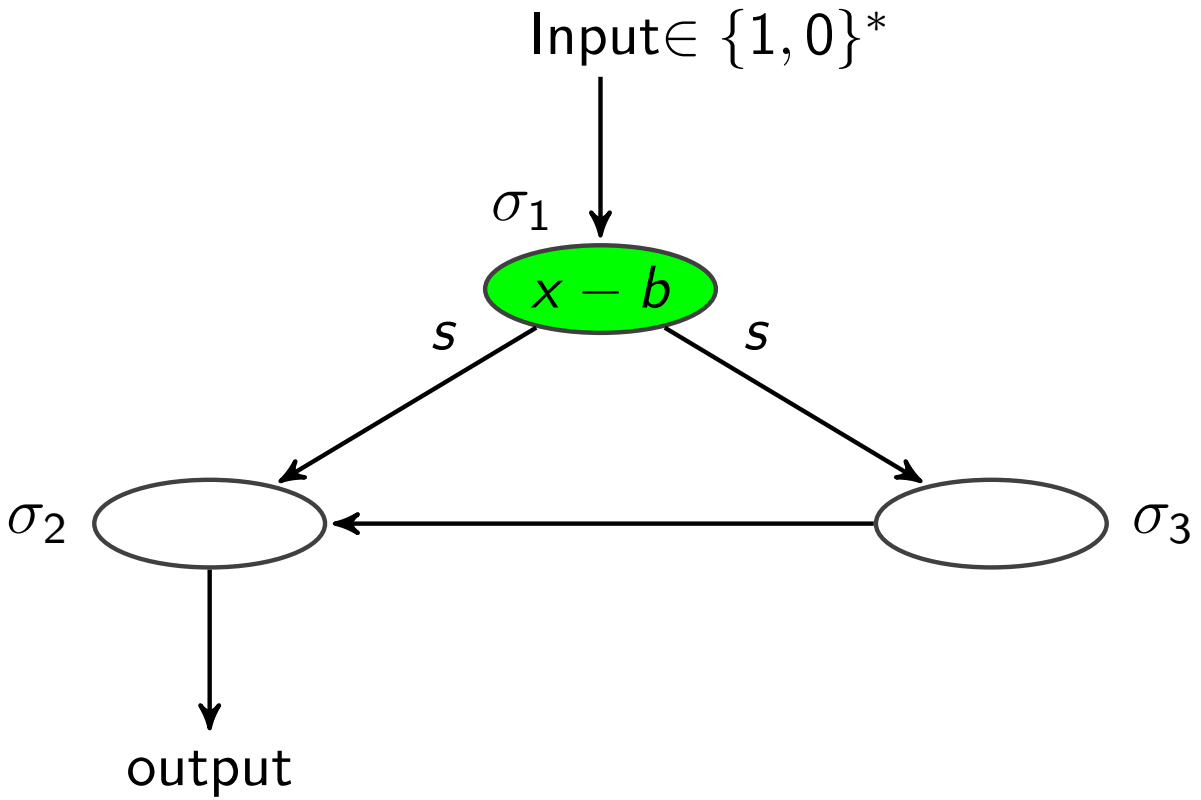
If $s^x \in L(E)$ ($L(E)$ is the language given by E) and $x > b$, then we execute the firing rule $E/s^b \rightarrow s$.

An SN P system executing rule $E/s^b \to s$



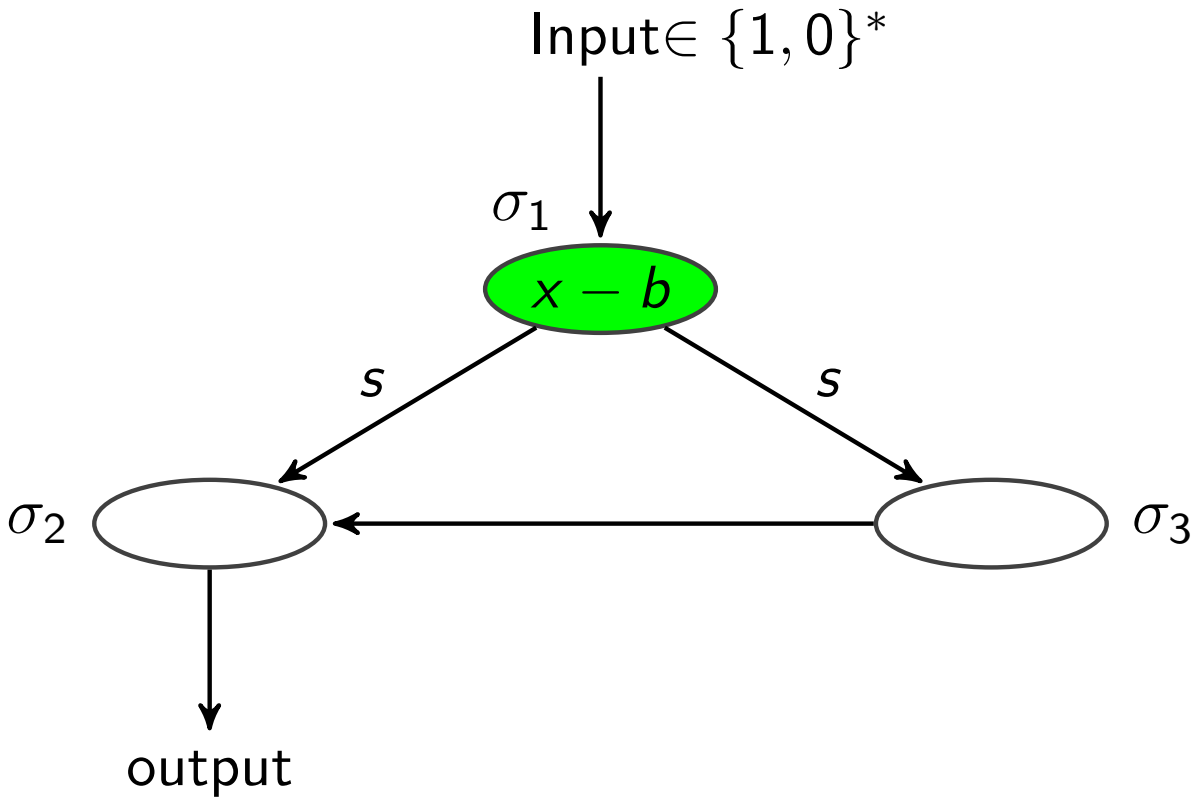
If $s^x \in L(E)$ ($L(E)$ is the language given by E) and $x > b$, then we execute the firing rule $E/s^b \to s$.

An SN P system executing rule $E/s^b \to s$



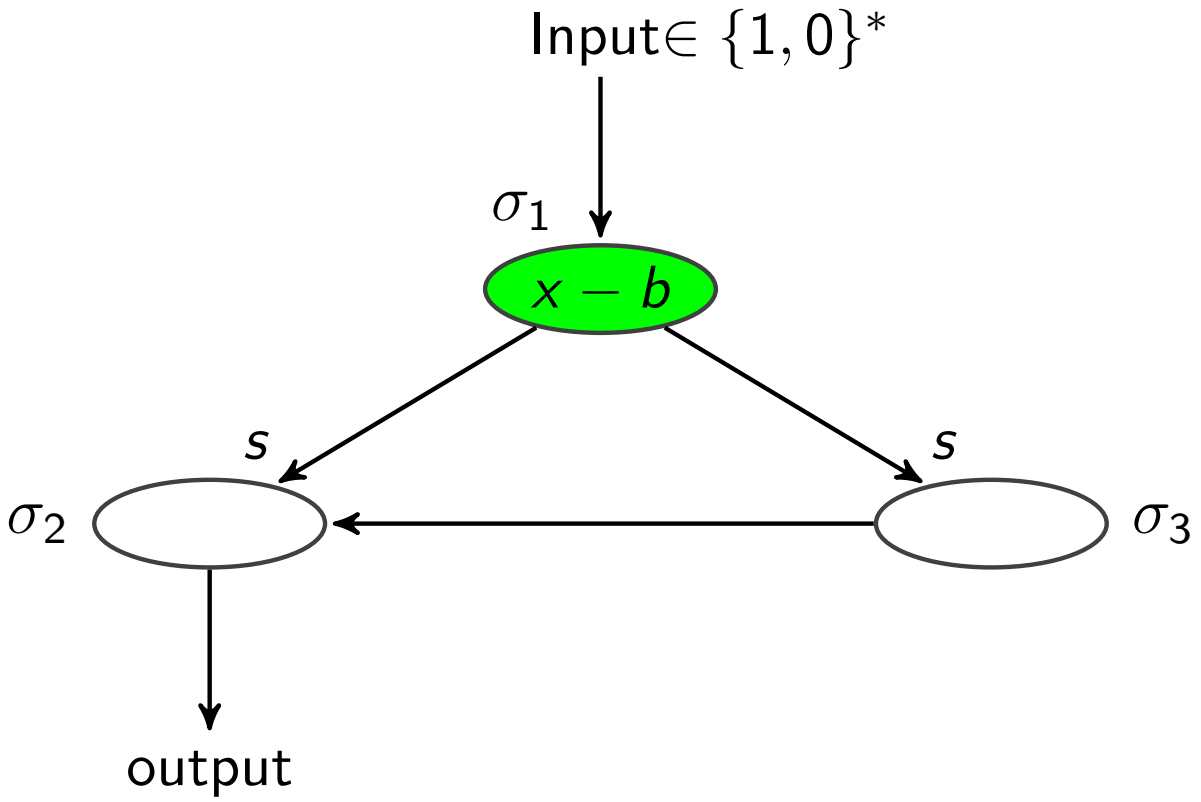
If $s^x \in L(E)$ ($L(E)$ is the language given by E) and $x > b$, then we execute the firing rule $E/s^b \to s$.

An SN P system executing rule $E/s^b \to s$



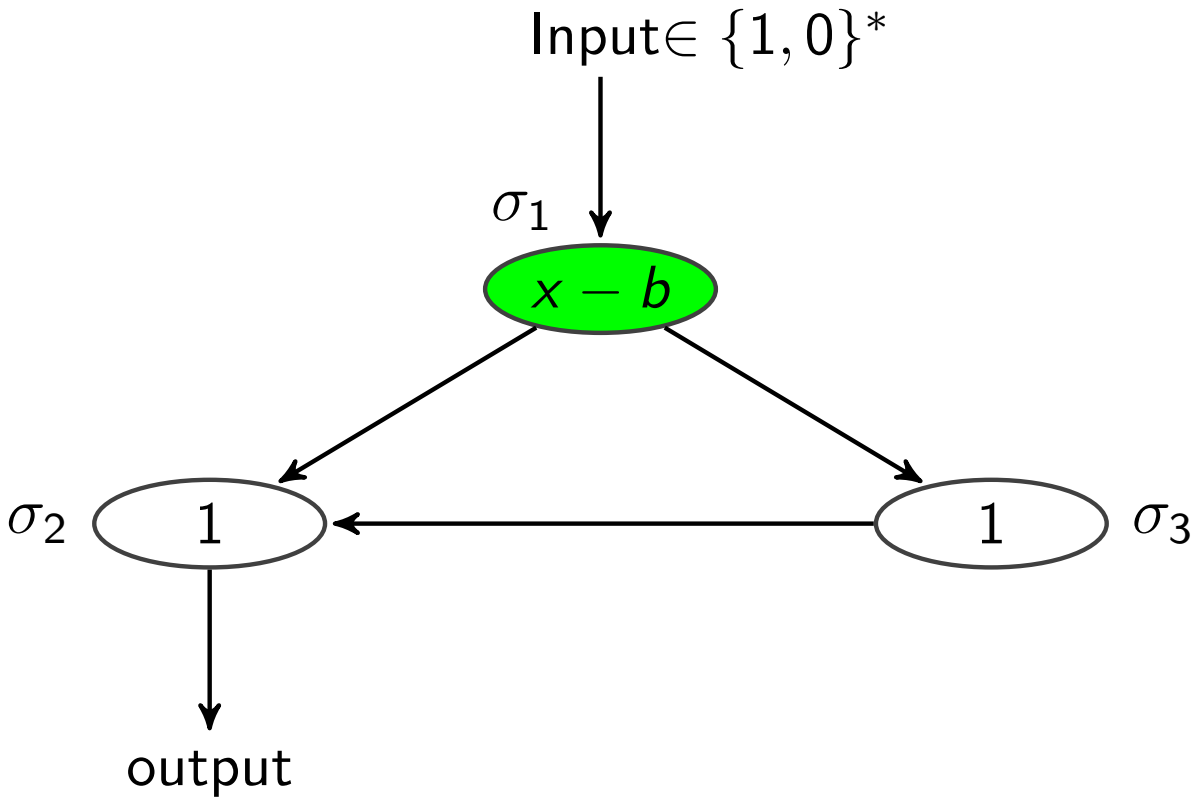
If $s^x \in L(E)$ ($L(E)$ is the language given by E) and $x > b$, then we execute the firing rule $E/s^b \to s$.

An SN P system executing rule $E/s^b \to s$



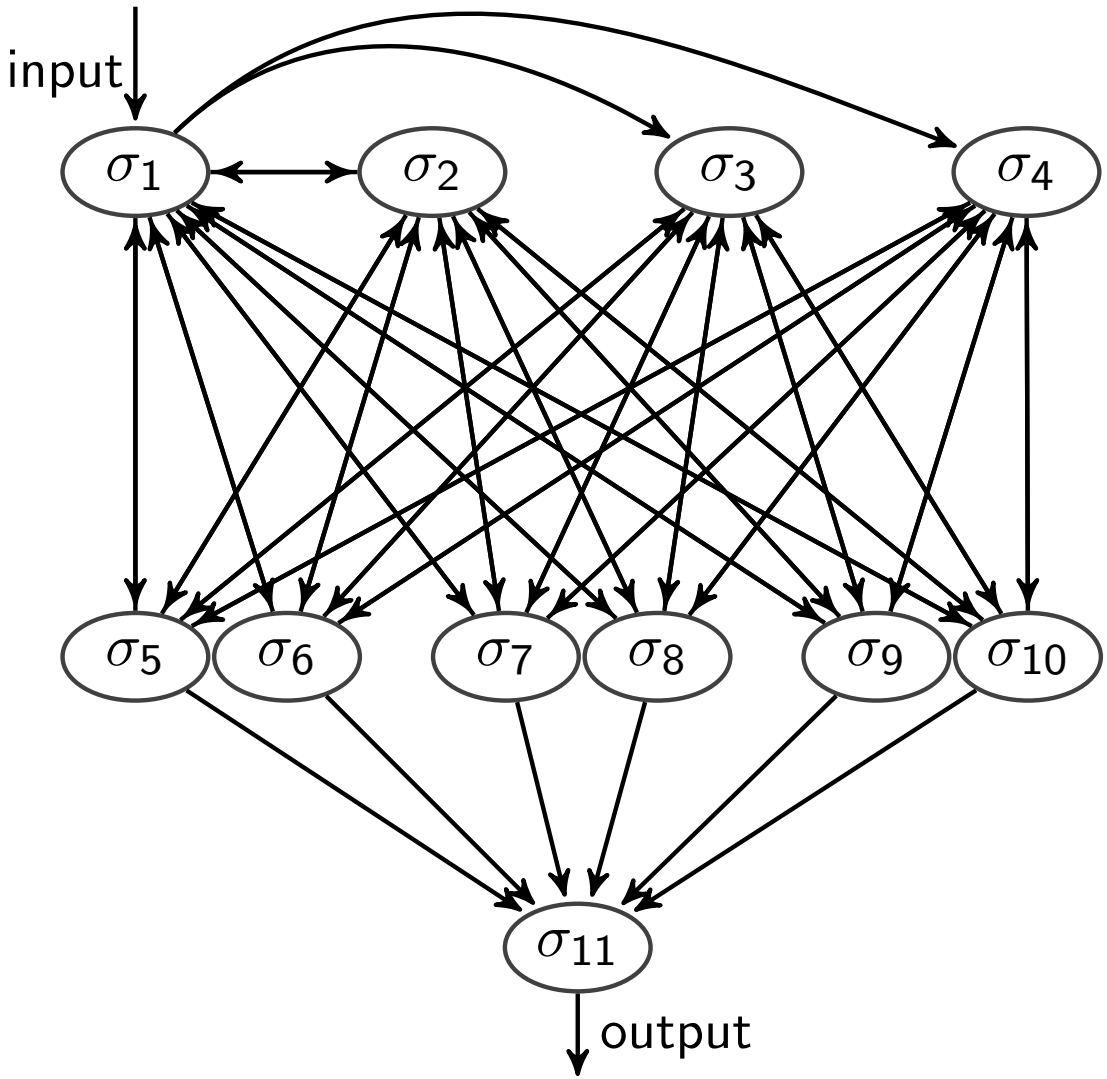
If $s^x \in L(E)$ ($L(E)$ is the language given by E) and $x > b$, then we execute the firing rule $E/s^b \to s$.

An SN P system executing rule $E/s^b \to s$

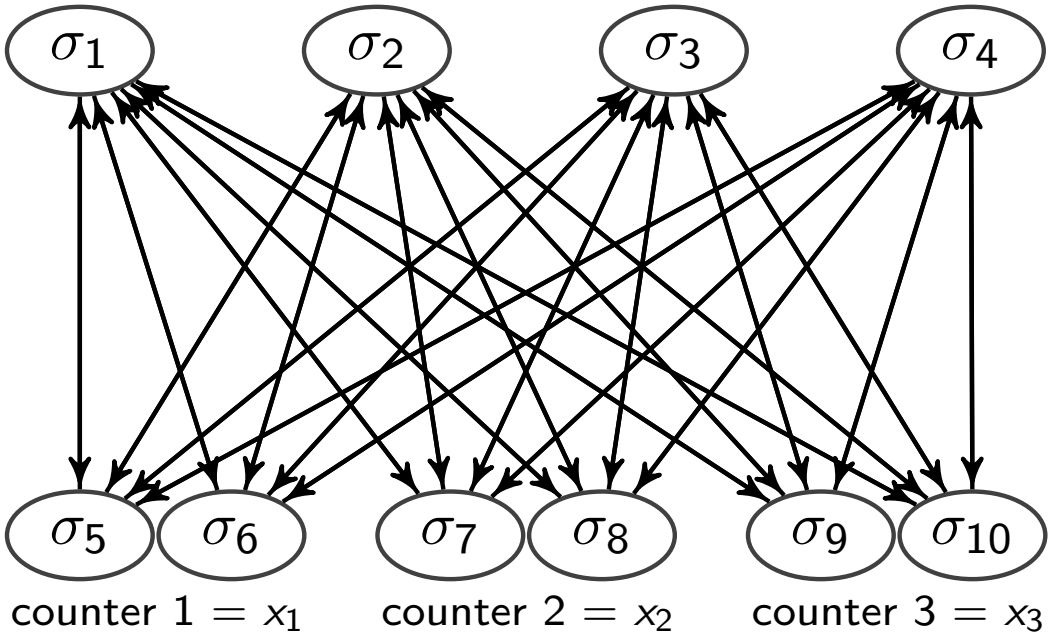


If $s^x \in L(E)$ ($L(E)$ is the language given by E) and $x > b$, then we execute the firing rule $E/s^b \to s$.

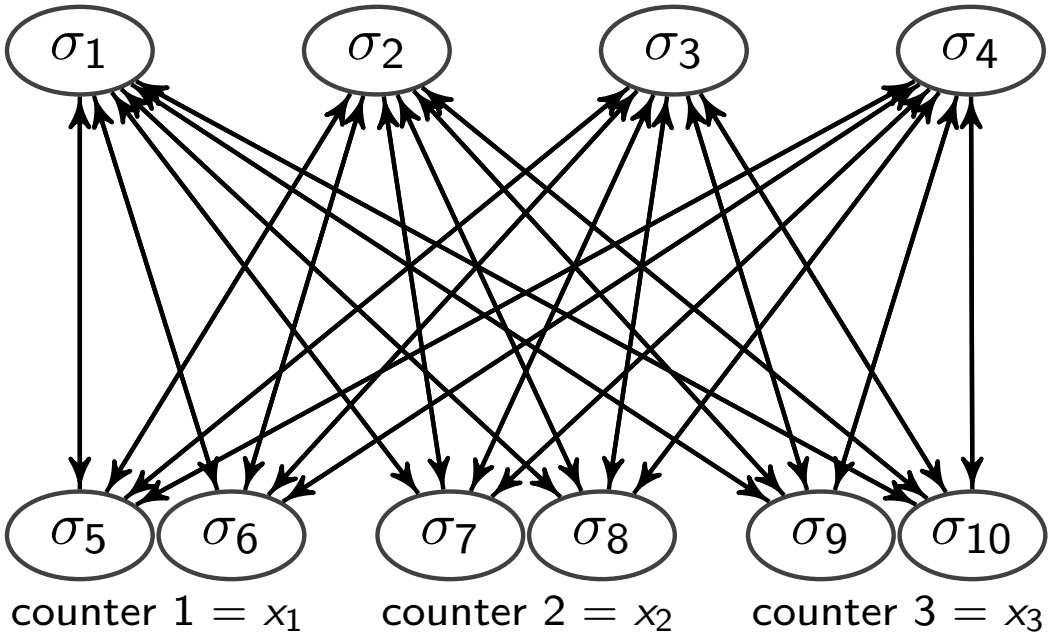
A universal SN P system with 11 neurons



Encoding a counter machine configuration

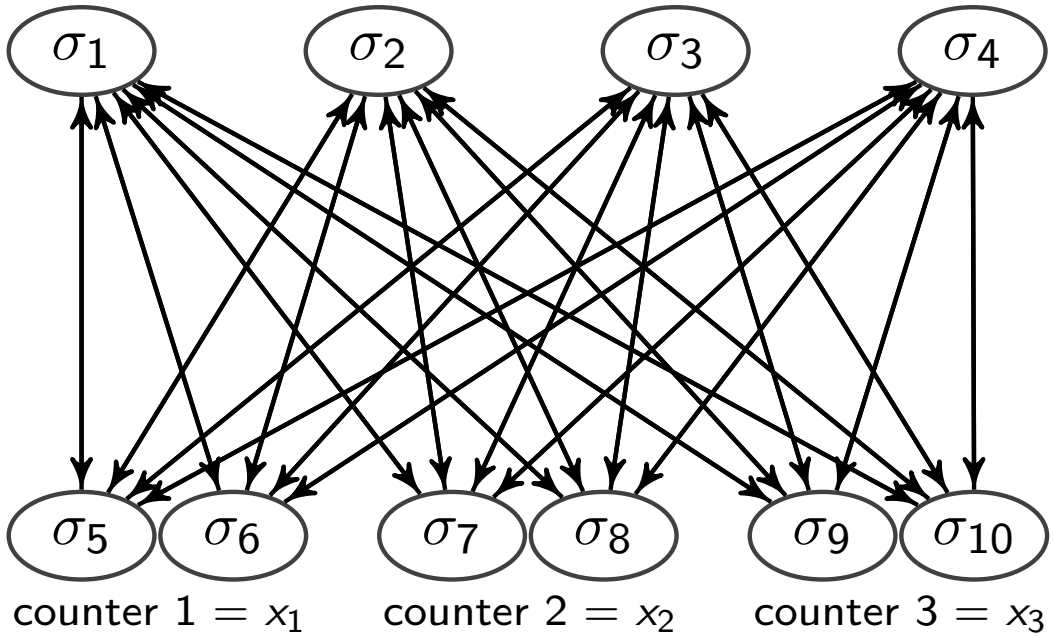


Encoding a counter machine configuration



$$\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 0,$$
$$\sigma_5, \sigma_6 : 16h(x_1 + 1) + 8i,$$
$$\sigma_7, \sigma_8 : 16h(x_2 + 1) + 8i,$$
$$\sigma_9, \sigma_{10} : 16h(x_3 + 1) + 8i,$$

Simulating $q_i : INC(1)q_l$



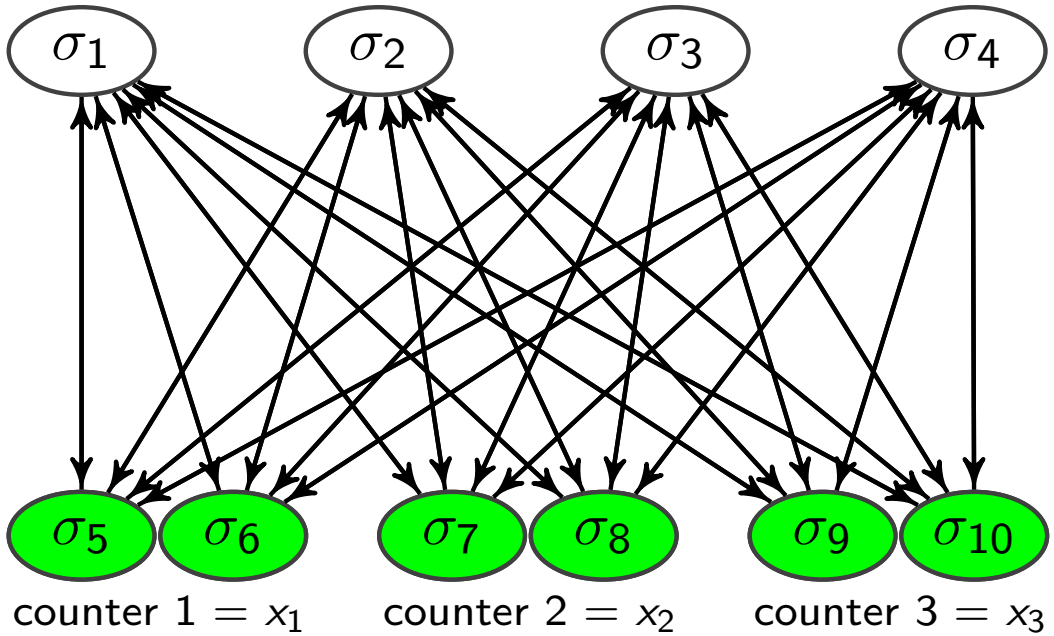
step 1 : $\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 0,$

$\sigma_5, \sigma_6 : 16h(x_1 + 1) + 8i, \quad s^{8i} \to s^{2i},$

$\sigma_7, \sigma_8 : 16h(x_2 + 1) + 8i, \quad s^{16h+8i} \to s^{3h},$

$\sigma_9, \sigma_{10} : 16h(x_3 + 1) + 8i, \quad s^{16h+8i} \to s^{3h}$

Simulating $q_i : INC(1)q_l$



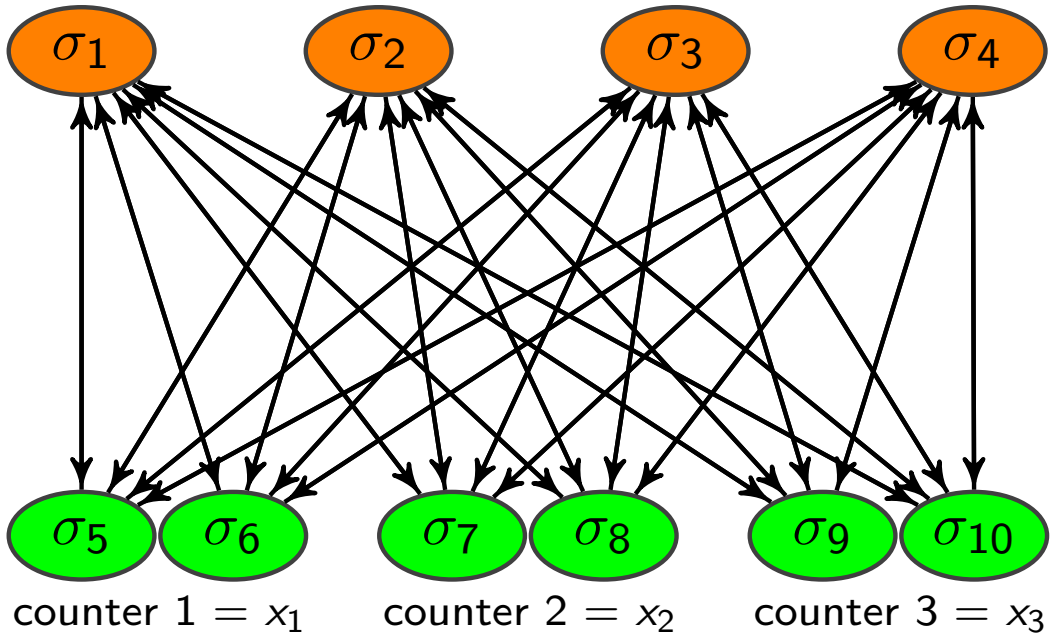
step 1 : $\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 0,$

$\sigma_5, \sigma_6 : 16h(x_1 + 1),$ $\quad s^{8i} \to s^{2i},$

$\sigma_7, \sigma_8 : 16hx_2,$ $\quad s^{16h+8i} \to s^{3h},$

$\sigma_9, \sigma_{10} : 16hx_3,$ $\quad s^{16h+8i} \to s^{3h}$

Simulating $q_i : INC(1)q_l$



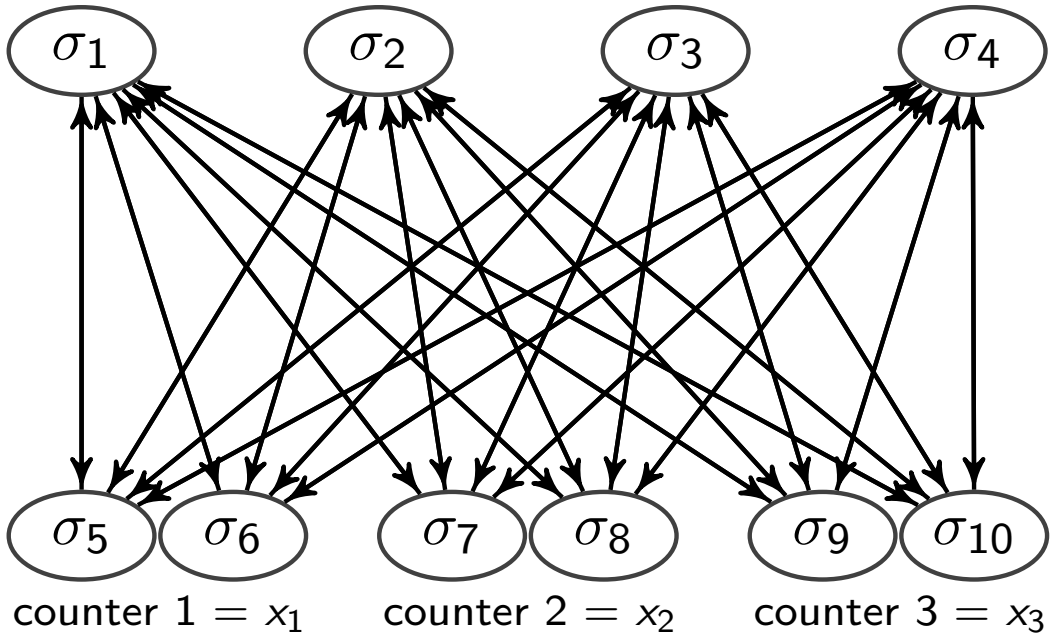
step 1 : $\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 0,$

$\sigma_5, \sigma_6 : 16h(x_1 + 1),$ $\quad s^{8i} \to s^{2i},$

$\sigma_7, \sigma_8 : 16hx_2,$ $\quad s^{16h+8i} \to s^{3h},$

$\sigma_9, \sigma_{10} : 16hx_3,$ $\quad s^{16h+8i} \to s^{3h}$

Simulating $q_i : INC(1)q_l$



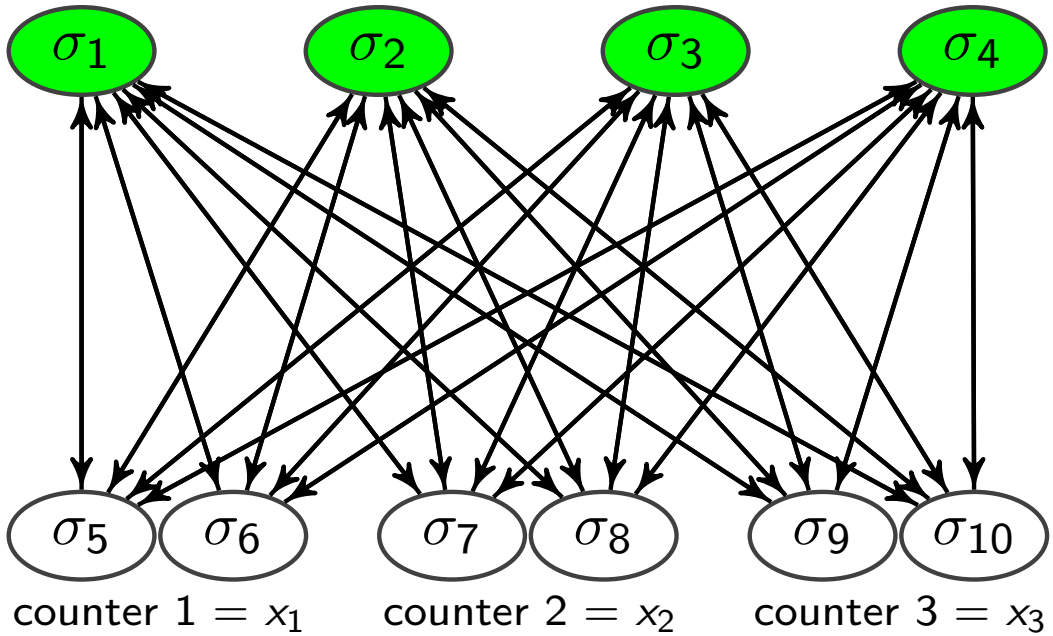
step 2 : $\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 12h + 4i, \quad s^{12h+4i} \to s^{4h+2l},$

$\sigma_5, \sigma_6 : 16h(x_1 + 1),$

$\sigma_7, \sigma_8 : 16hx_2,$

$\sigma_9, \sigma_{10} : 16hx_3,$

Simulating $q_i : INC(1)q_l$



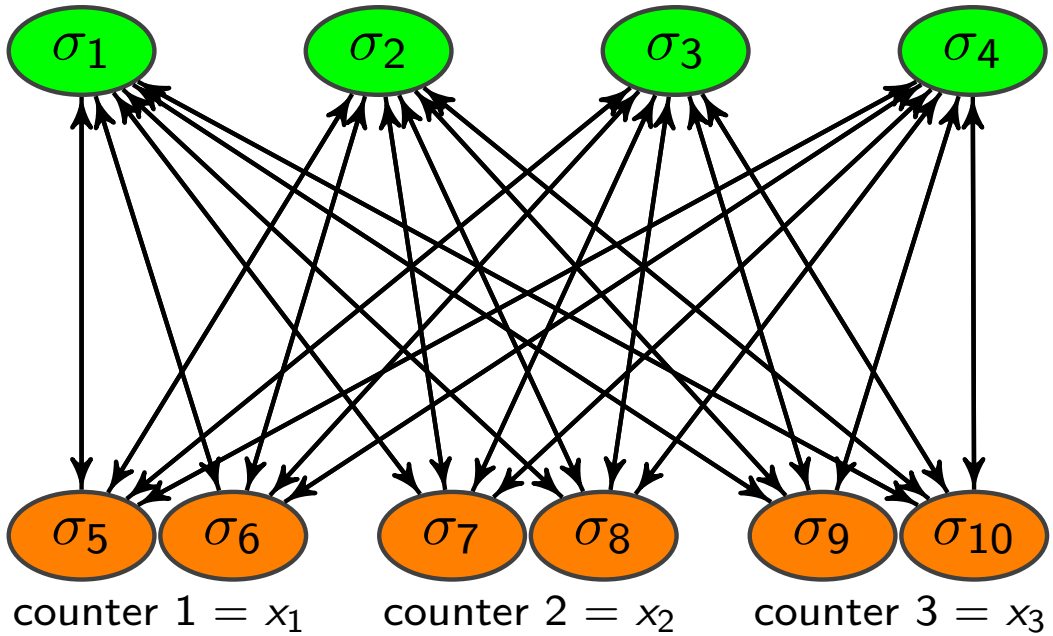
step 2 : $\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 12h + 4i, \quad s^{12h+4i} \to s^{4h+2l},$

$\sigma_5, \sigma_6 : 16h(x_1 + 1),$

$\sigma_7, \sigma_8 : 16hx_2,$

$\sigma_9, \sigma_{10} : 16hx_3,$

Simulating $q_i : INC(1)q_l$



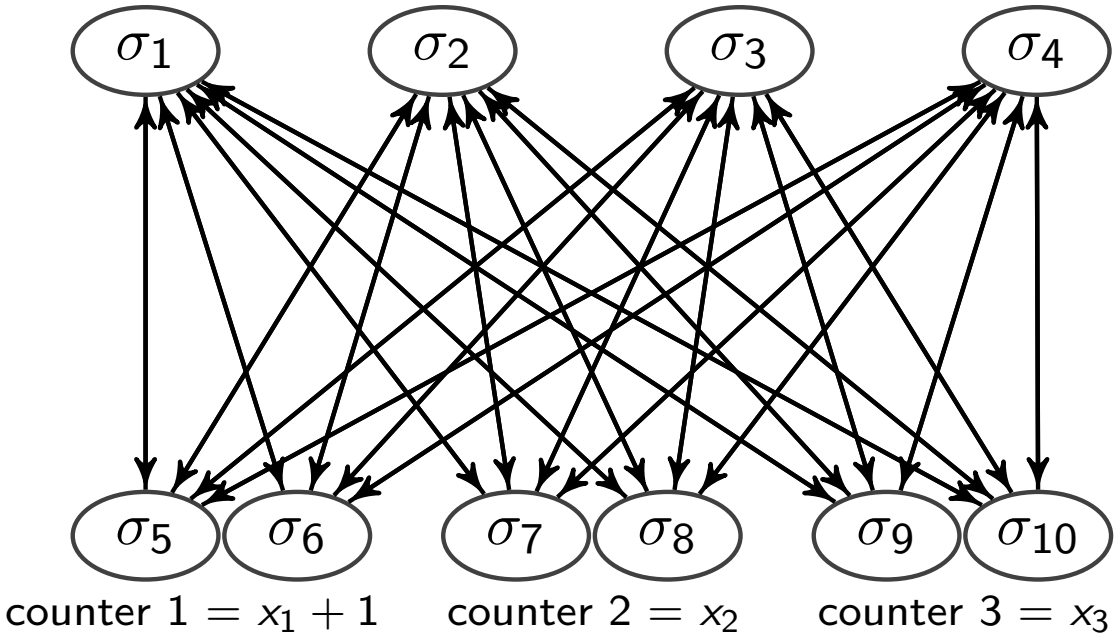
step 2 : $\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 12h + 4i, \quad s^{12h+4i} \to s^{4h+2l},$

$\sigma_5, \sigma_6 : 16h(x_1 + 1),$

$\sigma_7, \sigma_8 : 16hx_2,$

$\sigma_9, \sigma_{10} : 16hx_3,$

Simulating $q_i : INC(1)q_l$



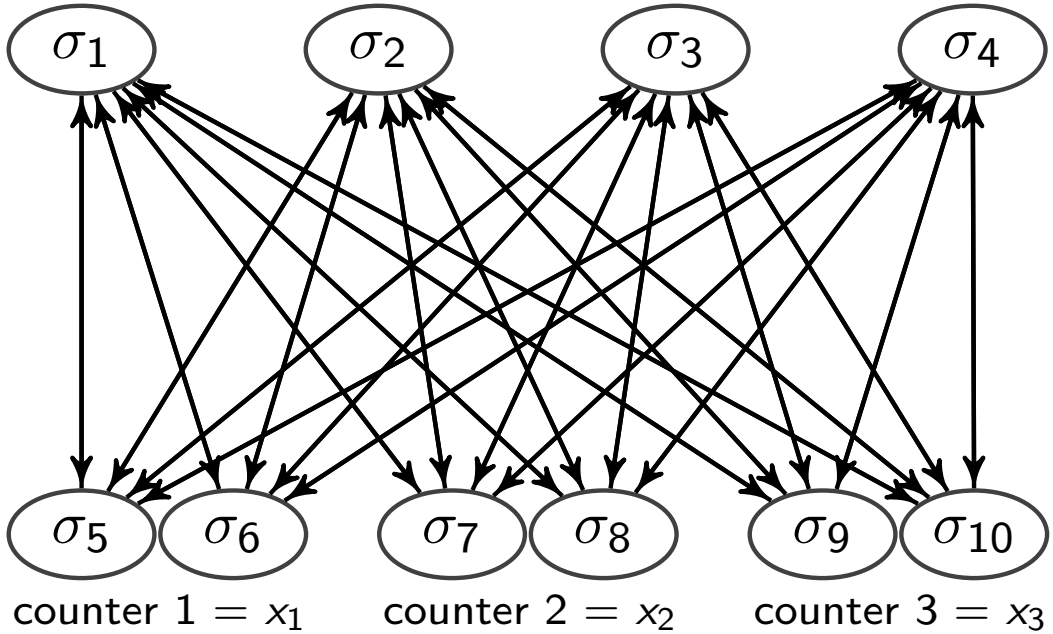
$\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 0,$

$\sigma_5, \sigma_6 : 16h(x_1 + 2) + 8l,$

$\sigma_7, \sigma_8 : 16h(x_2 + 1) + 8l,$

$\sigma_9, \sigma_{10} : 16h(x_3 + 1) + 8l,$

Simulating $q_i : DEC(1)q_l, q_k$ when $x > 0$



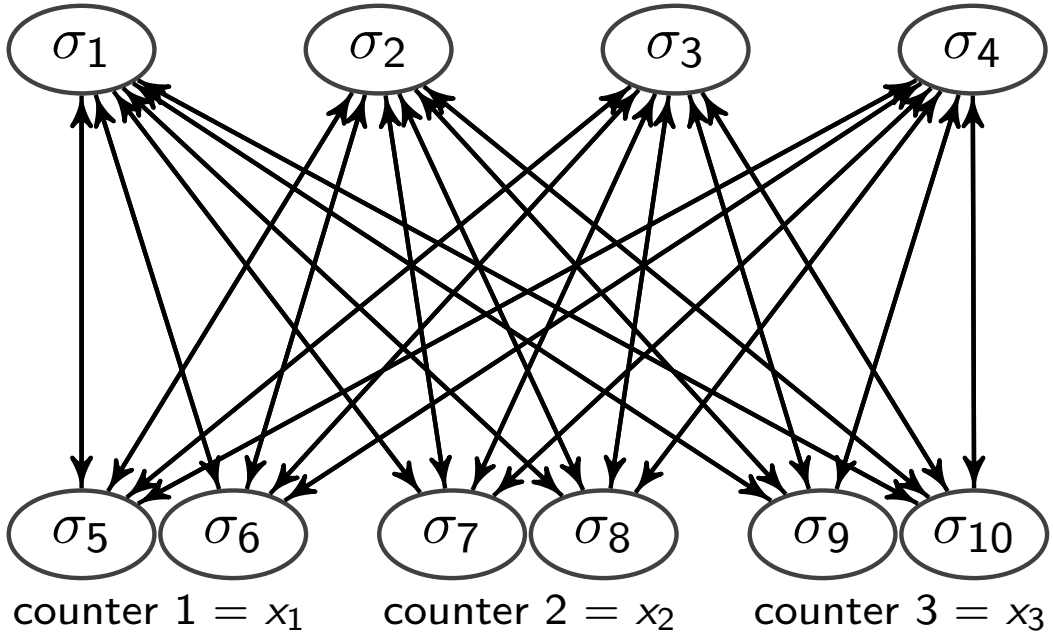
step 1 : $\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 0,$

$\sigma_5, \sigma_6 : 16h(x_1 + 1) + 8i, \quad s^{32h+8i} \to s^{2i},$

$\sigma_7, \sigma_8 : 16h(x_2 + 1) + 8i, \quad s^{16h+8i} \to s^{3h},$

$\sigma_9, \sigma_{10} : 16h(x_3 + 1) + 8i, \quad s^{16h+8i} \to s^{3h}$

Simulating $q_i : DEC(1)q_l, q_k$ when $x > 0$



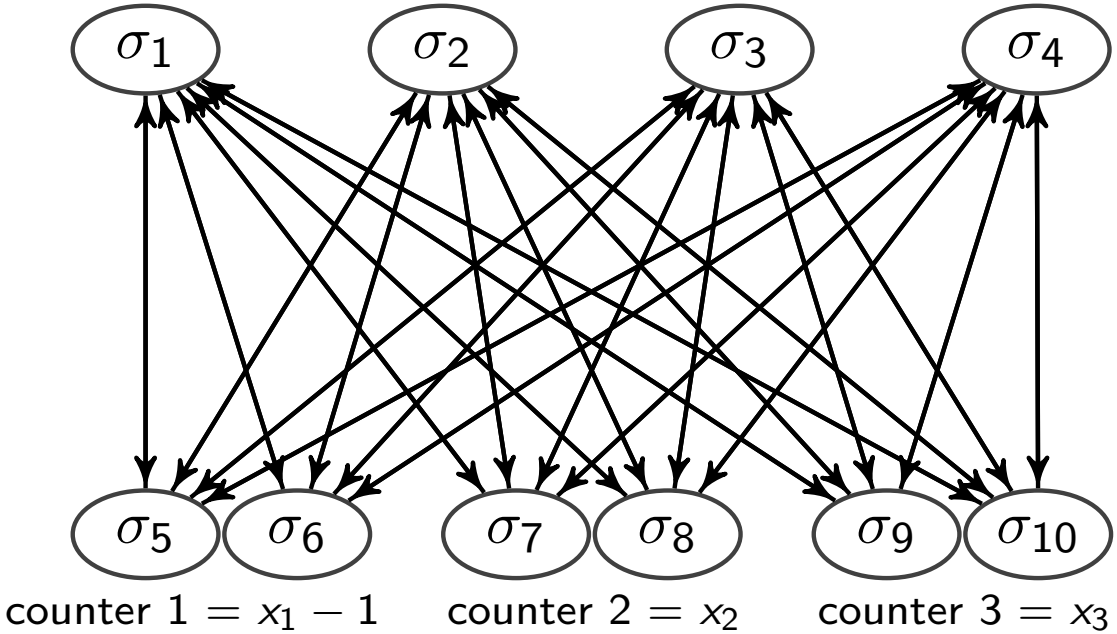
step 2 : $\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 12h + 4i, \quad s^{12h+4i} \to s^{4h+2l},$

$\sigma_5, \sigma_6 : 16h(x_1 - 1),$

$\sigma_7, \sigma_8 : 16hx_2,$

$\sigma_9, \sigma_{10} : 16hx_3,$

Simulating $q_i : DEC(1)q_l, q_k$ when $x > 0$



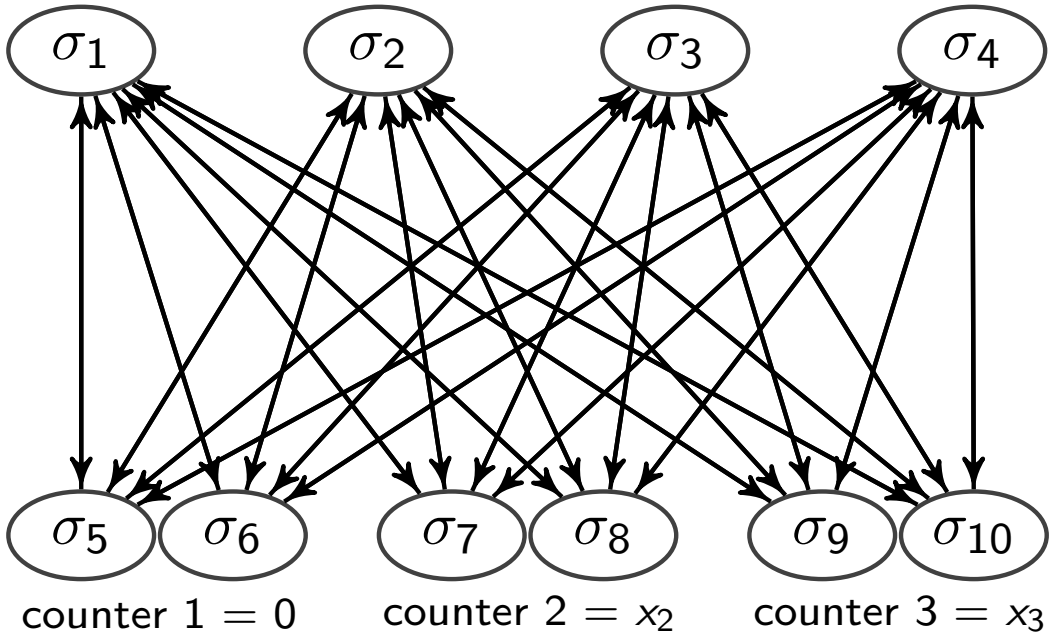
$\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 0,$

$\sigma_5, \sigma_6 : 16hx_1 + 8l,$

$\sigma_7, \sigma_8 : 16h(x_2 + 1) + 8l,$

$\sigma_9, \sigma_{10} : 16h(x_3 + 1) + 8l,$

Simulating $q_i : DEC(1)q_l, q_k$ when $x = 0$



step 1 :

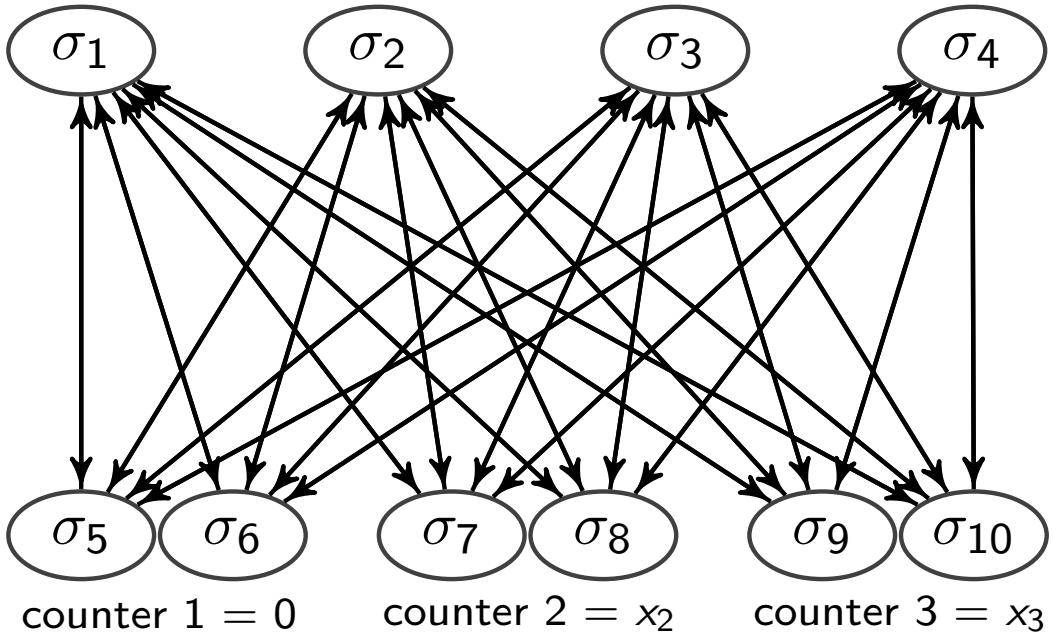
$\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 0,$

$\sigma_5, \sigma_6 : 16h + 8i,$ $\quad s^{16h+8i} \to s^{2i+1},$

$\sigma_7, \sigma_8 : 16h(x_2 + 1) + 8i,$ $\quad s^{16h+8i} \to s^{3h},$

$\sigma_9, \sigma_{10} : 16h(x_3 + 1) + 8i,$ $\quad s^{16h+8i} \to s^{3h}$

Simulating $q_i : DEC(1)q_l, q_k$ when $x = 0$



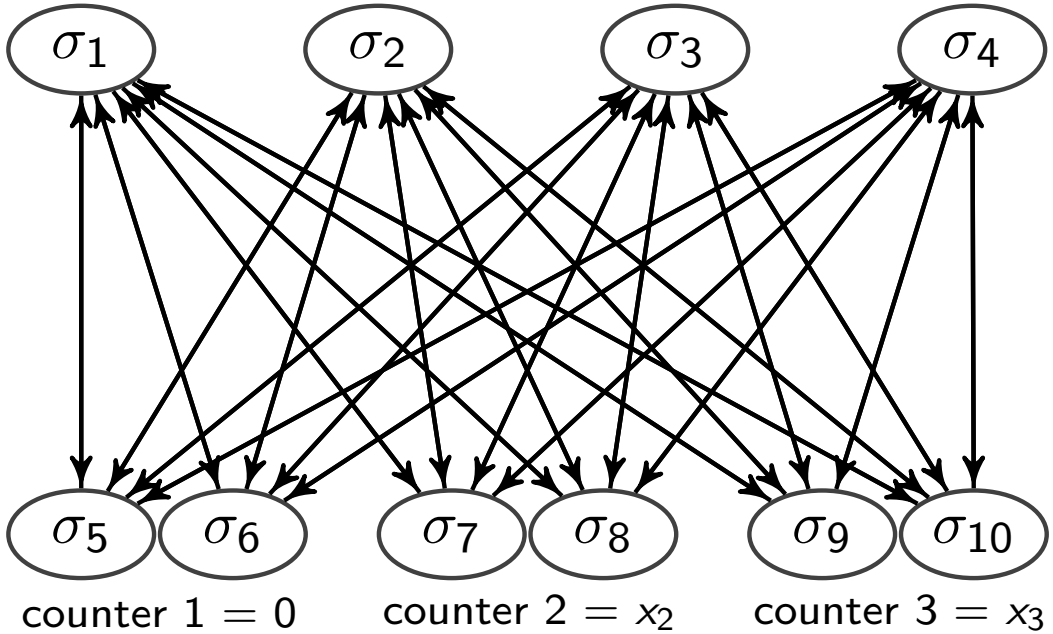
step 2 : $\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 12h + 4i + 2, \quad s^{12h+4i+2} \to s^{4h+2k},$

$\sigma_5, \sigma_6 : 0,$

$\sigma_7, \sigma_8 : 16hx_2,$

$\sigma_9, \sigma_{10} : 16hx_3,$

Simulating $q_i : DEC(1)q_l, q_k$ when $x = 0$



$$\sigma_1, \sigma_2, \sigma_3, \sigma_4 : 0,$$

$$\sigma_5, \sigma_6 : 16h + 8k,$$

$$\sigma_7, \sigma_8 : 16h(x_2 + 1) + 8k,$$

$$\sigma_9, \sigma_{10} : 16h(x_3 + 1) + 8k,$$

Conclusions

- Significant reduction in the number of neurons for universality in standard SN P systems.
  - Very simple input and output encodings.
  - No significant trade-off between number of neurons and the time/space complexity.
- Future work
  - System with less neurons and lower bounds
  - Simpler rules and simple neurons.
  - Improve the simulation time overheads used by Korec's small machines.

Thank you