On some classes of P systems

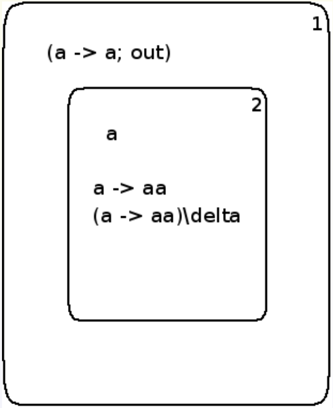
Avadhut M Sardeshmukh
Roll No 06329905

Guide : Prof S N Krishna
Department of Computer Science and Engineering
IIT Bombay

June 30, 2009

Outline

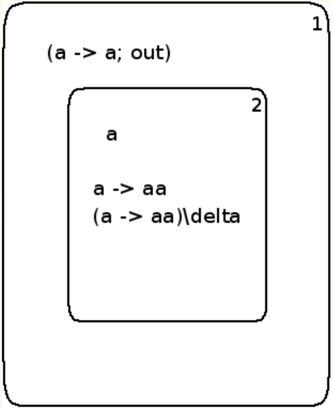
P systems with symbol objects



A membrane system (or P system) consists of :

- A hierarchical membrane structure (string of balanced parentheses).
- A multiset of objects in each region of this structure.
- Evolution rules.

P systems with symbol objects



Evolution :

- Initially, only membrane 2 has objects
- There is a nondeterministic choice of rules.
- But, *all* objects must evolve, so they double.
- If one or more evolve using second rule, the membrane dissolves.
- All 2^n objects are sent to environment.

Motivation

Membrane systems have been typically studied with following objectives in mind:

- From Biology to Mathematics – defining models
- From Mathematics to biology – modelling biological systems
- Efficiently solving computationally hard problems
- Simulation/Implementation

P systems with worm objects

- String objects (called as worms) inspired from the structure of DNA molecules.
- Operate on multisets of strings instead of multisets of symbols.
- Four types of operations – replication, splitting, recombination and mutation.
  - Replication - Rewrite a symbol by a string and replicate.
  - Recombination and splitting as in DNA computing.
  - Mutation rules - context free rewriting rules.

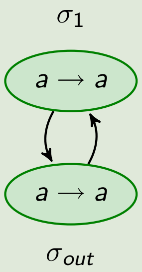
Spiking Neural (SN) P systems

- Inspired by *in silico* structure of neuron cells.
- Represented by a graph (Neurons ⇒ vertices, Synapses ⇒ edges).
- Neurons act as the compartments containing spikes–the only type of objects.
- At each clock tick, all enabled neurons send a spike to all the neighbors.
- Halting configuration : All neurons “open”, none firable.
- Various possibilities for defining the output.

Synchronous v/s Asynchronous mode of Operation

- Synchronous operation – system loops forever.
- Asynchronous operation – system can stop at any moment.

An SN P where synchronization matters



Universality of P systems with worm objects

- Best Known result : $NCP_m = NRE$ for all $m \geq 6$.
- Obtained result : $NCP_m = NRE$ for all $m \geq 4$.

Stochastic SN P systems

- Extension of SN P – Probabilistic asynchronism.
- After a rule is enabled, it does not fire immediately.
- The amount of time required for the rule to fire is determined by a probability distribution.
- The neuron is “open” for this time interval, unlike in the SN P systems.
- In between synchronous and asynchronous SN P systems.
- Asynchronous behavior controlled by probability distribution.

Experiments with Reliability of SSN P systems

- SSN P systems proved to be universal in [1] by simulating Register machines.
- Probability of correct simulation (of synchronous behavior) is called reliability.
- Reliability falls with increasing variance. How to improve it?

The Approach and results

- Redefine the ADD and SUB modules using suggestions from [1].
- Simulate using Mobius to measure the results.

The Approach and results

- Redefine the ADD and SUB modules using suggestions from [1].
- Simulate using Mobius to measure the results.
- Redundancy - number of neurons.

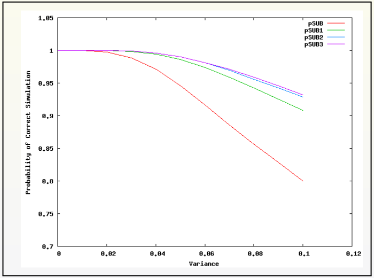


Figure: Reliability of SUB module for increasing redundancy

The Approach and results

- Redefine the ADD and SUB modules using suggestions from [1].
- Simulate using Mobius to measure the results.
- Redundancy - number of neurons.
- Reducing probability of incorrect simulation - Modification of rules in neurons.

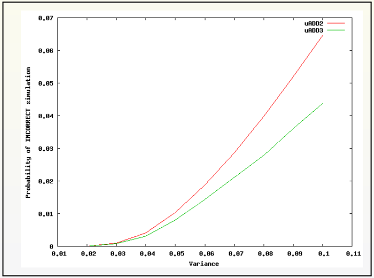


Figure: Probability of incorrect simulation of ADD

Directions for present work

- Achieving high reliability with asynchronous behavior is difficult.
- Asynchronous SN P systems are probably not as powerful as the synchronous ones.

Directions for present work

- Achieving high reliability with asynchronous behavior is difficult.
- Asynchronous SN P systems are probably not as powerful as the synchronous ones.
- Theoretical questions :
  1. What is the power of asynchronous SN P Systems?
  2. If less than synchronous, which features can make up for this loss?

Known results

- Synchronous SN P are universal. ([2])
- Asynchronous SN P with extended rules $(E/a^r \rightarrow a^p,\ r \geq p)$ are universal. ([3])
- Are Synchronous SN P with standard rules also universal?
- If not, extended rules make up for loss in power.

A Hierarchy

- We define the computing power of asynchronous SN P systems with increasing number of neurons (1, 2, 3..etc).
- Objective is to compare with synchronousn SN P system with corresponding number of neurons.
- Ideas from [7] borrowed/modified to define these systems.

1 Neuron can generate *FIN*

- For synchronous systems, this was argued (using delays) in [2].
- For asynchronous, we need extended rules (no delays here).
- Let $F = \{n_1, n_2, \ldots, n_{max}\}$ be in *FIN*.
  1. $a^{n_{max}} \longrightarrow a^{n_{max}}$
  2. $a^{n_{max}}/a^{n_i} \longrightarrow a^{n_i}$; $a^{n_{max}-n_i} \longrightarrow \lambda, \forall n_i \in F$

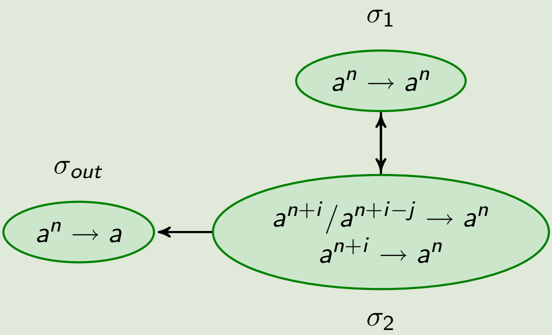
2 Neurons can generate at most *REG*

- 2 Neurons in synchronous mode can only generate *FIN*.
- In asynchronous, the output neuron can spike any number of times (unlike in synchronous).
- But still, number of spikes in the system can not increase.
- Number of possible different configurations are finite.
- Configurations serve as states of a finite automata.

3 Neurons can generate at least *REG*

- Regular grammar G, (N, T, S, P), and, N is $\{A_i | 1 \leq i \leq n\}$.
- $S = A_n$.
- For $A_i \rightarrow bA_j \in P$, neuron σ_2 contains rules as shown.

Asynch SN P Π to simulate G



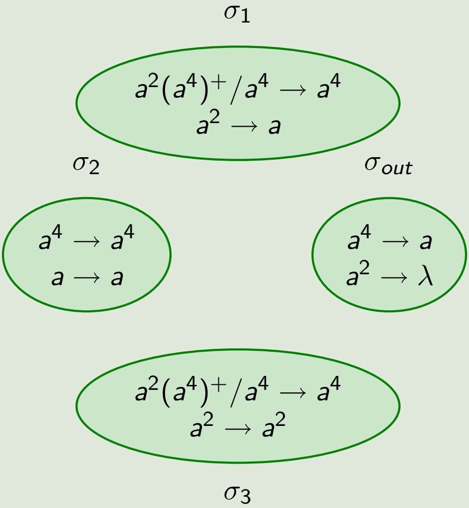
Non-Semilinear Sets : beyond context-freeness

Definition (Non-Semilinear set)

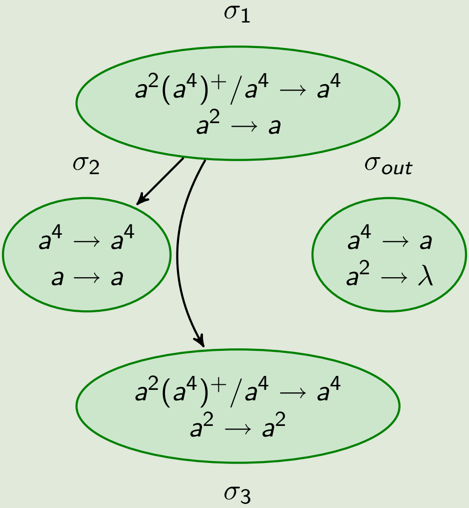
A set of integers is linear if it is of the form $\{c + pi | i \geq 0\}$. A set is semilinear if it is a finite union of linear sets. A set which is not semilinear is called non-semilinear.

- Parikh's Theorem : If L is context-free, then $\psi(L)$ is semilinear.
- So, non-semilinear sets are Parikh images of languages beyond context-free.
- For example, the Parikh image of $\{a^n b^n | n \geq 1\}$ is semilinear and Parikh image of the language $\{a^p | p \textit{ is a prime number}\}$ is non-semilinear.

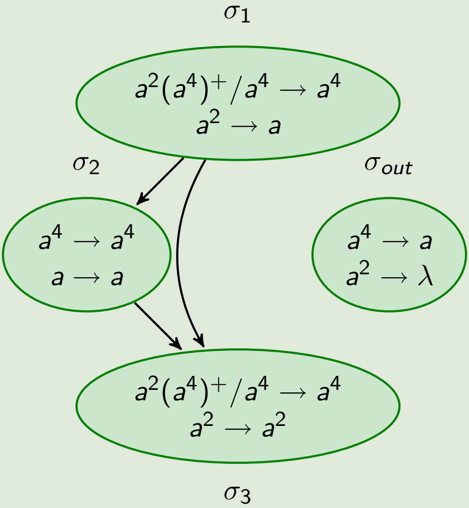
4 Neurons can generate non-seminilear language



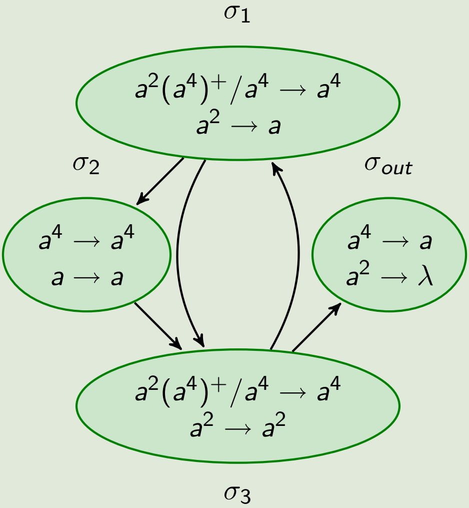
4 Neurons can generate non-seminilear language



4 Neurons can generate non-seminilear language



4 Neurons can generate non-seminilear language



A summary

No. of neurons →	**1**	**2**	**3**	**4**	*****
Synchronous	FIN	FIN	?	?	RE
Asynchronous	$FIN^{\dagger}$	$\subseteq REG$	$\supseteq REG^{\dagger}$	$\supset REG^{\dagger\dagger}$	$RE^{\dagger\dagger}$

Table: A Hierarchy : Synchronous v/s Asynchronous SN P

† : Extended rules of unbounded length needed.

†† : Extended rules of length 4 enough.

Extended rules of length 4 suffice (1/3)

- Earlier, extended rules capable of emitting unbounded no of spikes were used.
- We prove that extended rules of length 4 are enough for universality.
- Simulate a matrix grammar with appearance checking.

Extended rules of length 4 suffice (2/3)

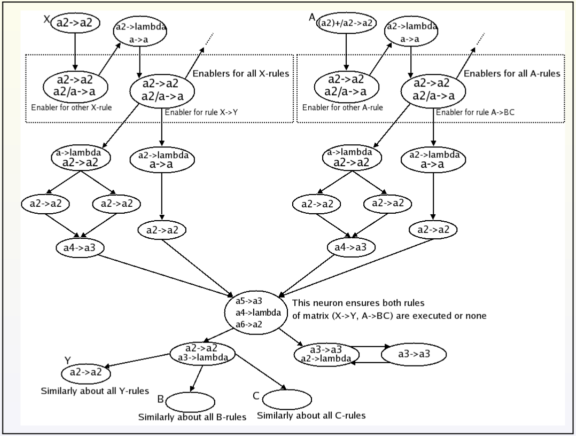


Figure: Module to simulate non-appearance-checking

Extended rules of length 4 suffice (3/3)

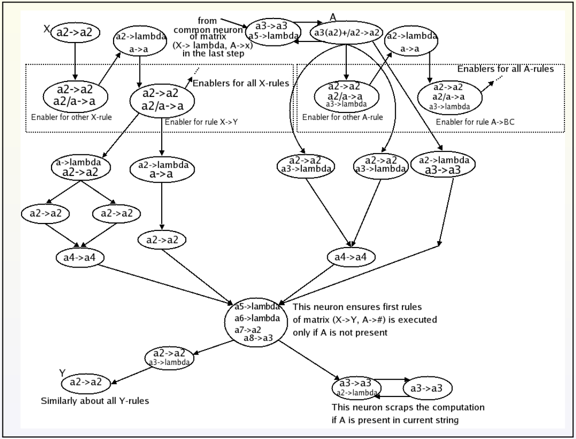


Figure: Module to simulate appearance-checking

Simulate using Conditional Grammars (1/2)

- Consider an asynchronous SN P system $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, out)$.
- We construct a conditional grammar $G = (N, T, S, P)$ to simulate Π
  - $N = \{S\} \cup \{A_i | 1 \leq i \leq m\}$.
  - $T = \{a, B\}$.

Simulate using Conditional Grammars (2/2)

- $S \longrightarrow A_1^{n_1} A_2^{n_2} \ldots A_m^{n_m} \in P$.
- Let r_j^i be the j^{th} spiking rule in neuron σ_i.

Simulate using Conditional Grammars (2/2)

- $S \longrightarrow A_1^{n_1} A_2^{n_2} \ldots A_m^{n_m} \in P$.
- Let r_j^i be the j^{th} spiking rule in neuron σ_i.
- $r_i^j : a^x/a^y \longrightarrow a$. Then, we add the rule (r, R) where, r is $A_i^y \longrightarrow B^y A_{j_1} A_{j_2} \ldots A_{j_n}$ for all neighbors.
  $R = A_1^* A_2^* \ldots A_i^x \ldots A_m^* B^*$.

Simulate using Conditional Grammars (2/2)

- $S \rightarrow A_1^{n_1} A_2^{n_2} \ldots A_m^{n_m} \in P$.
- Let r_j^i be the j^{th} spiking rule in neuron σ_i.
- $r_i^j : a^x(a^y)^*/a^z \rightarrow a$. Then, we add the rule (r, R) where, r is $A_i^z \rightarrow B^z A_{j_1} A_{j_2} \ldots A_{j_n}$ for all neighbors.
  $R = A_1^* A_2^* \ldots A_i^x (A_i^y)^* \ldots A_m^* B^*$.

Simulate using Conditional Grammars (2/2)

- $S \rightarrow A_1^{n_1} A_2^{n_2} \ldots A_m^{n_m} \in P$.
- Let r_j^i be the j^{th} spiking rule in neuron σ_i.
- $r_i^j : a^x \rightarrow \lambda$. Then, we add the rule (r, R) where, r is $A_i^x \rightarrow B^x$ and $R = A_1^* A_2^* \ldots A_i^x \ldots A_m^* B^*$.

Simulate using Conditional Grammars (2/2)

- $S \rightarrow A_1^{n_1} A_2^{n_2} \ldots A_m^{n_m} \in P$.
- Let r_j^i be the j^{th} spiking rule in neuron σ_i.

Also add :

- $A_i A_j \rightarrow A_j A_i, 1 \leq i < j \leq m$ to rearrange A_i's in the order.
- $BA_i \rightarrow A_i B$ to move all B's at the end.
- $A_{out} \rightarrow a$; $A_i \rightarrow B$ for all i, such that they are enabled only in halting configuration.

Membership problem : Semidecidable

Given an asynchronous SN P system Π and a number n, can Π generate the number n?

- Assume, without loss of generality, that number of spikes in σ_{out} can only increase.
- Iteratively compute the set ψ of sentential forms α reachable from S, until $|\alpha|_{A_{out}} \geq n$.
- If ψ contains any halting configuration with $|\alpha|_{A_{out}} = n$, accept, otherwise reject.

Membership problem : Semidecidable

Given an asynchronous SN P system Π and a number n, can Π generate the number n?

- Assume, without loss of generality, that number of spikes in σ_{out} can only increase.
- Iteratively compute the set ψ of sentential forms α reachable from S, until $|\alpha|_{A_{out}} \geq n$.
- If ψ contains any halting configuration with $|\alpha|_{A_{out}} = n$, accept, otherwise reject.
- The process may never stop, so the problem is semi-decidable.

Conclusions

- In many cases, adding extended rules to asynchronous systems makes them equal in power to synchronous ones.
- That we can simulate asynchronous SN P using conditional grammars hints at their sub-universality.
- Asynchronous SN P systems using extended rules of length up to four become universal.
- It is less likely to happen with rules of length less than four :
  - In asynchronous mode, we can not distinguish the absence of a signal from the delay in its arrival.

Conclusions

- In many cases, adding extended rules to asynchronous systems makes them equal in power to synchronous ones.
- That we can simulate asynchronous SN P using conditional grammars hints at their sub-universality.
- Asynchronous SN P systems using extended rules of length up to four become universal.
- It is less likely to happen with rules of length less than four :
  - In asynchronous mode, we can not distinguish the absence of a signal from the delay in its arrival.
  - We have to define each (presence and absence) with a different number of spikes, so extended rules become unavoidable.

Conclusions

- In many cases, adding extended rules to asynchronous systems makes them equal in power to synchronous ones.
- That we can simulate asynchronous SN P using conditional grammars hints at their sub-universality.
- Asynchronous SN P systems using extended rules of length up to four become universal.
- It is less likely to happen with rules of length less than four :
    - With length only 2, we can assign 2 spikes to mean presence and 1 spike to mean absence.

Conclusions

- In many cases, adding extended rules to asynchronous systems makes them equal in power to synchronous ones.
- That we can simulate asynchronous SN P using conditional grammars hints at their sub-universality.
- Asynchronous SN P systems using extended rules of length up to four become universal.
- It is less likely to happen with rules of length less than four :
  - With length only 2, we can assign 2 spikes to mean presence and 1 spike to mean absence.
  - The problem ??

Conclusions

- In many cases, adding extended rules to asynchronous systems makes them equal in power to synchronous ones.
- That we can simulate asynchronous SN P using conditional grammars hints at their sub-universality.
- Asynchronous SN P systems using extended rules of length up to four become universal.
- It is less likely to happen with rules of length less than four :
  - With length 3, again, how to detect presence of one signal and absence of the other?

Future work

- Optimality of number of membranes required for a P system with worm objects to become universal, is unanswered yet.
- The basic question about the (sub)universality of asynchronous SN P systems using standard rules needs to be further investigated.
- We observe that features such as number of neurons, number of rules per neuron, extended rules, etc are intricately related.
- Defining systematic hierarchy changing only one parameter and keeping all others constant will be interesting.

Thank you

Questions?

References

- Matteo Cavaliere and Ivan Murra.
  Experiments on the reliability of stochastic spiking neural p systems.
  Technical Report 26/2007, The Microsoft Research-University of Trento Center for Computational and Systems Biology, July 2007.
- Mihai Ionescu, Gheorghe Păun, and Takashi Yokomori.
  Spiking neural p systems.
  Fundam. Inf., 71(2,3):279–308, 2006.
- Matteo Cavaliere and Mihai Ionescu and Gheorghe Păun and Oscar Ibarra and Omer Egecioglu and Sara Woodworth.
  Asynchronous Spiking Neural P Systems.
  Proceedings of the 13th International Meeting on DNA Computing, 2007.
- Klemm, Konstantin and Bornholdt, Stefan.
  Topology of biological networks and reliability of information processing
  PNAS, 102(51):18414–18419, December, 2005.
- G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J.M.Doyle, W.H. Sanders, P.Webster
  The Mobius Modeling Tool
  Proceedings of International Workshop on Petri Nets and Performance Models, IEEE Computer Society, 2001.
- Gheorghe Paun.
  Introduction to membrane computing.
  1998.
- Xingyi Zhang, Xiangxiang Zeng, and Linqiang Pan
  On languages generated by asynchronous spiking neural P systems
  Theor. Comput. Sci., 410(26):2478-2488, 2008.