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

1. Introduction
   - P systems: The basic model
   - Two Variants

2. Summary of work done in past
   - Universality of P systems with worm objects
   - Reliability of Stochastic SN P systems

3. Present work: Asynchronous SN P systems
   - Computational Power
   - A Hierarchy: Synchronous v/s Asynchronous
   - A bound on the complexity
   - A decision problem

4. Summary and future work
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

\[ a \rightarrow a \]
\[ a \rightarrow a \]
\[ \sigma_1 \]
\[ \sigma_{out} \]
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 $m \geq 4$. 
Introduction

Summary of work done in past

Present work: Asynchronous SN P systems

Summary and future work

Universality of P systems with worm objects

Reliability of Stochastic SN P systems

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 synchronous 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 $F = \{n_1, n_2, \ldots, n_{\text{max}}\}$ be in $FIN$.
  
  1. $a^{n_{\text{max}}} \rightarrow a^{n_{\text{max}}}$
  2. $a^{n_{\text{max}}} / a^{n_i} \rightarrow a^{n_i}$ ; $a^{n_{\text{max}} - n_i} \rightarrow \lambda$, $\forall n_i \in F$
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 cannot increase.
- Number of possible different configurations are finite.
- Configurations serve as states of a finite automata.
3 Neurons can generate at least $\text{REG}$

- 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 $\sigma_2$ contains rules as shown.

Asynch SN P $\Pi$ to simulate $G$

\[
\begin{align*}
\sigma_1 & : a^n \rightarrow a^n \\
\sigma_2 & : a^{n+i}/a^{n+i-j} \rightarrow a^n \\
& \quad a^{n+i} \rightarrow a^n \\
\sigma_{out} & 
\end{align*}
\]
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 \( \{ a^p | p \text{ is a prime number} \} \) is non-semilinear.
4 Neurons can generate non-seminilear language

\[ a^2(a^4)^+ / a^4 \rightarrow a^4 \]
\[ a^2 \rightarrow a \]

\[ a^4 \rightarrow a^4 \]
\[ a \rightarrow a \]

\[ a^4 \rightarrow a \]
\[ a^2 \rightarrow \lambda \]

\[ a^2(a^4)^+ / a^4 \rightarrow a^4 \]
\[ a^2 \rightarrow a^2 \]
4 Neurons can generate non-seminilereal language

\[ a^2(a^4)^+/a^4 \rightarrow a^4 \]
\[ a^2 \rightarrow a \]

\[ a^4 \rightarrow a^4 \]
\[ a \rightarrow a \]

\[ \sigma_1 \]

\[ a^2(a^4)^+/a^4 \rightarrow a^4 \]
\[ a^2 \rightarrow a^2 \]

\[ \sigma_3 \]

\[ a^4 \rightarrow a \]
\[ a^2 \rightarrow \lambda \]

\[ \sigma_{out} \]
4 Neurons can generate non-semiunlinear language

\[ a^2(a^4)^+ / a^4 \rightarrow a^4 \]
\[ a^2 \rightarrow a \]
\[ a^4 \rightarrow a^4 \]
\[ a \rightarrow a \]
\[ a^4 \rightarrow a \]
\[ a^2 \rightarrow \lambda \]
\[ a^2(a^4)^+ / a^4 \rightarrow a^4 \]
\[ a^2 \rightarrow a^2 \]

\[ \sigma_1 \]
\[ \sigma_2 \]
\[ \sigma_{out} \]
\[ \sigma_3 \]
4 Neurons can generate non-seminillear language

\[
\begin{align*}
\sigma_1 &: \quad a^2(a^4)^+ / a^4 \rightarrow a^4 \\
&\quad a^2 \rightarrow a \\
\sigma_2 &: \quad a^4 \rightarrow a^4 \\
&\quad a \rightarrow a \\
\sigma_3 &: \quad a^2(a^4)^+ / a^4 \rightarrow a^4 \\
&\quad a^2 \rightarrow a^2 \\
\sigma_{out} &: \quad a^4 \rightarrow a \\
&\quad a^2 \rightarrow \lambda
\end{align*}
\]
A summary

<table>
<thead>
<tr>
<th>No. of neurons</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous</td>
<td>FIN</td>
<td>FIN</td>
<td>?</td>
<td>?</td>
<td>RE</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>FIN†</td>
<td>⊆ REG</td>
<td>⊇ REG†</td>
<td>⊇ REG††</td>
<td>RE††</td>
</tr>
</tbody>
</table>

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
  \[ \Pi = (O, \sigma_1, \ldots, \sigma_m, \text{syn, out}). \]
- We construct a conditional grammar \( G = (N, T, S, P) \) to simulate \( \Pi \)
  - \( N = \{ S \} \cup \{ A_i | 1 \leq i \leq m \} \).
  - \( T = \{ a, B \} \).
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 \( \sigma_i \).
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 $\sigma_i$.
- $r^j_i : a^x/a^y \rightarrow a$. Then, we add the rule $(r, R)$ where, $r$ is $A_i^y \rightarrow B^y A_{j_1} A_{j_2} \ldots A_{j_n}$ for all neighbors.
- $R = A_1^x A_2^x \ldots A_i^x \ldots A_m^x B^x$. 

Avadhut M Sardeshmukh  
Roll No 06329905  
On some classes of P systems
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 \( \sigma_i \).
- \( r_j^i : a^x(a^y)^/a^z \rightarrow a. \) 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 \( r^i_1, \ldots, r^i_n \) be the \( j^{th} \) spiking rule in neuron \( \sigma_i \).
- \( r^i_j : a^x \rightarrow \lambda. \) Then, we add the rule \( (r, R) \) where, \( r \) is \( A_i^x \rightarrow B^x \) and \( R = A_1^* A_2^* \ldots A_i^* \ldots A_m^* B^*. \)
Simulate using Conditional Grammars (2/2)

- \( S \to A_1^n A_2^n \ldots A_m^n \in P. \)
- Let \( r_{ij}^j \) be the \( j^{th} \) spiking rule in neuron \( \sigma_i \).

Also add:

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

Given an asynchronous SN P system \( \Pi \) and a number \( n \), can \( \Pi \) generate the number \( n \)?

- Assume, without loss of generality, that number of spikes in \( \sigma_{out} \) can only increase.

- Iteratively compute the set \( \psi \) of sentential forms \( \alpha \) reachable from \( S \), until \( |\alpha|_{A_{out}} \geq n \).

- If \( \psi \) contains any halting configuration with \( |\alpha|_{A_{out}} = n \), accept, otherwise reject.
Membership problem : Semidecidable

Given an asynchronous SN P system \( \Pi \) and a number \( n \), can \( \Pi \) generate the number \( n \)?

- Assume, without loss of generality, that number of spikes in \( \sigma_{out} \) can only increase.
- Iteratively compute the set \( \psi \) of sentential forms \( \alpha \) reachable from \( S \), until \( |\alpha|_{A_{out}} \geq n \).
- If \( \psi \) 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 cannot 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?
Introduction
Summary of work done in past
Present work: Asynchronous SN P systems
Summary and future work

References


Avadhut M Sardeshmukh Roll No 06329905 On some classes of P systems