Simulating Fuzzy Reasoning SN P systems on the GPU

L.F. Macías-Ramos, M.Á. Martínez-del-Amor,
M. García Quismondo, M.J. Pérez-Jiménez

Research Group on Natural Computing
Universidad de Sevilla
NVIDIA CUDA Research Center

February 2015
13th BWMC, Sevilla (Spain)

Contents

Outline

1. Preliminaries
   - Simulators
   - GPU Computing
2. Fuzzy Reasoning Spiking Neural P systems
   - Informal specification
   - Applications
3. Coding Roadmap
4. CUDA Design Roadmap
5. Conclusions and future work

Need for efficient simulation

- P system simulators assist us in:
  - Formal verification of P systems.
  - Computational modeling based on P systems:
    - Experimentally validate models.
    - Conduct virtual experiments.
- Necessity of efficiency to handle large-size instances.
- Sequential simulators serialize the parallelism: twice inefficient.

Need for efficient simulation

- P system simulators assist us in:
  - Formal verification of P systems.
  - Computational modeling based on P systems:
    - Experimentally validate models.
    - Conduct virtual experiments.
- Necessity of efficiency to handle large-size instances.
- Sequential simulators serialize the parallelism: twice inefficient.

Outline

Why is the GPU interesting for simulating P systems?

- Interesting properties:
  - High level of parallelism *(from 16 to 2800 cores)*
  - Shared memory system *(easily synchronized)*
  - Scalability *(multi-GPU systems)*
  - Cheap technology *(cost and maintenance)*

- NVIDIA's Tesla GPUs at RGNC
  - Tesla C1060: *240 cores, 4 GB memory.*
  - Tesla K40: *2880 cores, 12 GB memory.*

Why is the GPU interesting for simulating P systems?

- Interesting properties:
  - High level of parallelism *(from 16 to 2800 cores)*
  - Shared memory system *(easily synchronized)*
  - Scalability *(multi-GPU systems)*
  - Cheap technology *(cost and maintenance)*

- NVIDIA's Tesla GPUs at RGNC
  - Tesla C1060: *240 cores, 4 GB memory.*
  - Tesla K40: *2880 cores, 12 GB memory.*

JAVA vs GPU computing

- JAVA is clearly inefficient in comparison to GPU computing.
- JAVA is much more flexible than GPU computing.
- P-Lingua framework: a very complete library of parsers and simulators.
- P-Lingua is written in JAVA so...
- ... how can we join JAVA (flexibility) and GPU computing (power)?
- Line 1: create a client/server architecture.
- Line 2: call directly GPU code from JAVA!
- This is accomplished (first time in the community) with JCUDA library [1].

[1] JCUDA - A JAVA binding for NVIDIA CUDA. http://www.jcuda.org/.

Outline

FRSN P systems - Specification

- Special class of SN P systems.
- Incorporate fuzzy logic elements.
- Truth values are modelled with fuzzy numbers.
- Neurons only contain one spike.
- The spike has a potential or pulse value, a fuzzy number.
- Two kind of neurons: proposition neurons and rule neurons.
- Arcs connecting neurons model fuzzy production rules.
- Matrix-based simulation algorithms.

Models under study

- Fuzzy Reasoning Spiking Neural P systems
  - Real Numbers [2]
  - Trapezoidal Numbers [3]

---

[2] H. Peng, J. Wang, M.J. Pérez, H. Wang, J. Shao, T. Wang. Fuzzy reasoning spiking neural P system for fault diagnosis. Information Sciences, **235**, (2013), 106–116.

[3] T. Wang, G. Zhang, J. Zhao, Z. He, J. Wang, M.J. Pérez. Fault Diagnosis of Electric Power Systems Based on Fuzzy Reasoning Spiking Neural P Systems. IEEE Transactions on Power Systems, (2014), ISSN 0885-8950.

Matrix-based algorithm

Fuzzy reasoning algorithm based on FRSN P system.

INPUT: parameter matrixes U, V, Λ, H_1, H_2, λ_p, λ_r, and initial inputs α_p^0, a_p^0.

OUTPUT: The fuzzy truth values of propositions associated with the neurons in O.

Step 1) Let $\alpha_r^0 = (0, 0, \ldots, 0)^T, a_r^0 = (0, 0, \ldots, 0)^T$.

Step 2) Let $t = 0$.

Step 3)

(1) Process the firing of proposition neurons.

$$\beta_p^t = fire\left(\alpha_p^t, a_p^t, \lambda_p\right), b_p^t = fire\left(1, a_p^t, \lambda_p\right), \alpha_p^t = update\left(\alpha_p^t, a_p^t, \lambda_p\right),$$
$$a_p^t = update\left(a_p^t, a_p^t, \lambda_p\right), B_p^t = diag\left(b_p^t\right).$$

(2) Compute the truth values of rule neurons and the number of received spikes.

$$\alpha_r^{t+1} = \alpha_r^t \oplus \left[\left(H_1 \cdot \left((B_p^t \cdot U)^T \odot \beta_p^t\right)\right) + (H_2 \cdot \left(\left(B_p^t \cdot U\right)^T \otimes \beta_p^t\right)\right],$$
$$a_r^{t+1} = a_r^t + \left[\left(B_p^t \cdot U\right)^T \cdot b_p^t\right].$$

(3) Process the firing of rule neurons.

$$\beta_r^{t+1} = fire(\Lambda \cdot \alpha_r^{t+1}, a_r^{t+1}, \lambda_r), b_r^{t+1} = fire(1, a_r^{t+1}, \lambda_r),$$
$$\alpha_r^{t+1} = update\left(\alpha_r^{t+1}, a_p^t, \lambda_p\right), a_r^{t+1} = update\left(a_r^{t+1}, a_p^t, \lambda_p\right), B_r^{t+1} = diag(b_r^{t+1}).$$

(4) Compute the truth values of proposition neurons and the number of received spikes.

$$\alpha_p^{t+1} = \alpha_p^t \oplus \left[(V \cdot B_r^{t+1}) \otimes \beta_r^{t+1}\right], a_p^{t+1} = a_p^t + \left[(V \cdot B_r^{t+1}) \cdot b_r^{t+1}\right].$$

Step 4) If $a_p^{t+1} = (0, 0, \ldots, 0)^T$ and $a_r^{t+1} = (0, 0, \ldots, 0)^T$ (computation halts), the reasoning results are obtained; otherwise, $t = t + 1$, go to Step 3).

Step 1) Let $g = 0$ be the reasoning step;

Step 2) Set initial values of $\boldsymbol{D}_1$, $\boldsymbol{D}_2$, $\boldsymbol{D}_3$, $\boldsymbol{E}$, $\boldsymbol{C}$ and the termination condition $\mathbf{0}_1 = (unknown, unknown, \ldots, unknown)_t^T$. The initial values of $\boldsymbol{\theta}$ and $\boldsymbol{\delta}$ are set to $\boldsymbol{\theta}_g = (\theta_{1g}, \theta_{2g}, \ldots, \theta_{sg})$ and $\boldsymbol{\delta}_g = (\delta_{1g}, \delta_{2g}, \ldots, \delta_{tg})$, respectively.

Step 3) g is increased by one.

Step 4) The firing condition of each input neuron ($g = 1$) or each proposition neuron ($g > 1$) is evaluated. If the condition is satisfied and there is a presynaptic rule neuron, the neuron fires and transmits a spike to the next rule neuron.

Step 5) Compute the fuzzy truth value vector $\boldsymbol{\delta}_g$ according to (2):

$$\boldsymbol{\delta}_g = \left(\boldsymbol{D}_1^T \odot \boldsymbol{\theta}_{g-1}\right) \oplus \left(\boldsymbol{D}_2^T \odot \boldsymbol{\theta}_{g-1}\right) \oplus \left(\boldsymbol{D}_3^T \circledast \boldsymbol{\theta}_{g-1}\right). \quad (2)$$

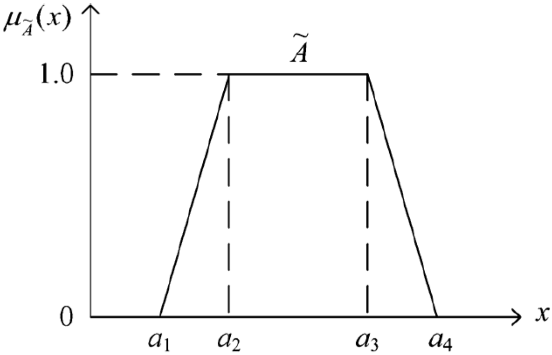
Step 6) If $\boldsymbol{\delta}_g = \mathbf{0}_1$, the algorithm stops and outputs the reasoning results, otherwise, it go to *Step 7)*.

Step 7) Evaluate the firing condition of each rule neuron. If the condition is satisfied, the rule neuron fires and transmits a spike to the next proposition neuron.

Step 8) Compute the fuzzy truth value vector $\boldsymbol{\theta}_g$ according to (3). Go to *Step 3)*:

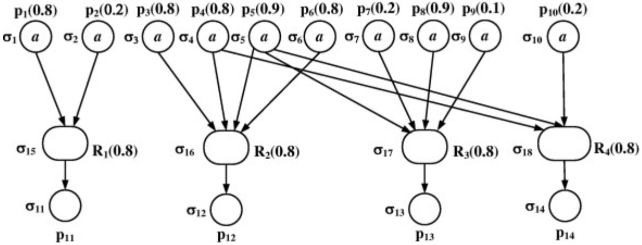
$$\theta_g = \boldsymbol{E}^T \circledast (\boldsymbol{C} \odot \delta_g). \quad (3)$$

Trapezoidal Numbers



Trapezoidal fuzzy number.

Example



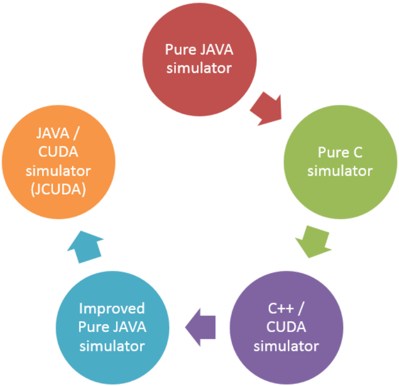
p1(0.8)
p2(0.2)
p3(0.8)
p4(0.8)
p5(0.9)
p6(0.8)
p7(0.2)
p8(0.9)
p9(0.1)
p10(0.2)
σ1
σ2
σ3
σ4
σ5
σ6
σ7
σ8
σ9
σ10
a
σ15
R1(0.8)
σ16
R2(0.8)
σ17
R3(0.8)
σ18
R4(0.8)
σ11
σ12
σ13
σ14
p11
p12
p13
p14

Outline

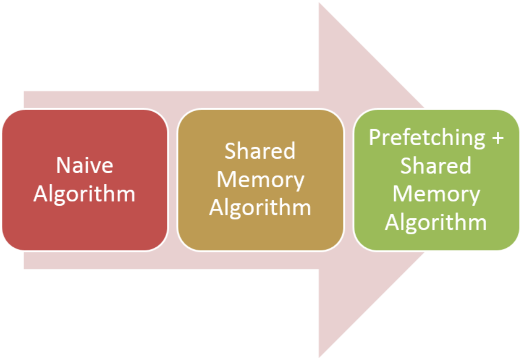
FRSN P systems - Applications

- Fault diagnosis on power systems.
- Fault diagnosis on electric systems on high speed trains.
- Watermarking.

Coding Roadmap



Coding Design Roadmap



Conclusions and future work

- CUDA development is alive.
- First time invoking CUDA code from JAVA in the community.
- First time implementing simulators for FRSNPS in P-Lingua.
- To do: testing, performance analysis, new binding technologies (IBM), new FRSNPS models (wights, etc.).