Sisteme cu membrane

Răzvan Vasile[1],
Prof. Emerit Marian Gheorghe[2]

razvan.vasile@yahoo.com
m.gheorghe@bradford.ac.uk

[1] **Department of Computer Science, University of Bucharest, Bucharest, Romania**
[2] **Department of Computer Science, University of Bradford, Bradford, UK**

11 martie 2025

Overview

1. kernel P Systems (kPS)
2. Exemplu în kPS (Problema Partiției)
3. SN P systems (SN P)
4. Exemplu în SN P
5. KPWorkBench
6. Exemple în KPWorkBench - demonstrație

Kernel P System – introducere informală

- Structura dinamica, sub forma de graf
- Utilizeaza multiseturi de obiecte
- Regulile pot avea garzi (conditii booleene)
  - rescriere si comunicare
  - structurale (ex. diviziune celulara/de membrane)
- Fiecare compartiment are un tip din care deriva, iar tipul compartimentului defineste setul de reguli. Fiecare instanta are propriul multiset initial.
- Vom ilustra utilizarea modelului cu o problema NP-completa - ‘Partition problem’.

Partition Problem

Given a finite set $V = \{v_1, \ldots v_n\}$, a function *weight* on V with positive integer values,

$weight : V \rightarrow Z^+$, such that $weight(v_i) = k_i$,

and for any subset W of V, $weight(W)$ means the sum of the weights of the elements of W.

For a given k, from Z^+ decide whether there exists a partition V_1, V_2 of V, such that $weight(V_1) = weight(V_2) = k$.

Obviously, $weight(V) = 2k$.

Schița de rezolvare

Ideea. Se genereaza toate submultimile – brute force. Se verifica daca una dintre submultimile stricte, W, are proprietatea $weight(W) = k$.

Modelul. kernel P (kP) system. Vom folosi un kP sistem cu doua tipuri, t_1 si t_2, din care initializam doua instante It_1 si It_2.
It_1 va functiona ca o interfata unde primim raspunsul, iar It_2 va fi masinaria supusa procesului de divizare celulara unde obtinem toate submultimile. Cele 2^n submultimi se obtin in n pasi, simuland o recursie.

Regulile recursiei:
$[A_i] \rightarrow [BiA_{i+1}]_2[A_{i+1}]_2; 1 \leq i < n$ – cazul iterativ
$[A_n]_2 \rightarrow [B_nX]_2[X]_2;$ – cazul de baza

Complexitatea algoritmului: lineara in raport de cardinalul lui V.

kP Sisteme: Tipuri

Se dau n compartimente, fiecare cu un tip $t_i, i = 1, \ldots, n$. Fiecare t_i este dat de un set de reguli, R_i, si o strategie de executie, s_i.

Strategia, s_i, poate fi paralelism maxim, arbitrar, executie secventiala, selectie.

Obs.
1. Cand sunt instantiate tipurile atunci se introduce multiseturile initiale.
2. kP sistemul obtinut poate avea diferite strategii de executii in compartimente.
3. Regulile de comunicare implica tipuri, iar la executie se alege o instanta arbitrara, daca sunt mai multe de acelasi tip (nedeterminism).

Modelul de Tip kP Sistem (1)

Modelul folosit pentru Partition Problem implica doua tipuri:

- t_1 pentru interfata
- t_2 pentru generarea tuturor submultimilor.

Strategia de executie, $s_i, i = 1, 2$, este **paralelism maxim**.

Modelul de Tip kP Sistem (2)

$kP = (A, \mu, C1, C2, 0)$, unde μ conecteaza C_1, C_2, de tipuri t_1 si respectiv t_2. Rezultatul se obtine in mediu.

Regulile tipului t_1, R_1 (trimit raspunsul in mediu, identificat cu 0)
$r_{1,1} : S \rightarrow (yes, 0)\{\geq T\}$ – exista cel putin o solutie
$r_{1,2} : S \rightarrow (no, 0)\{\geq F \;\wedge\; < T\}$ – nu exista solutie

Regulile tipului t_2, R_2, sunt
- diviziune de membrane (genereaza 2^n compartimente de tip t_2)
$r_{2,i} : [A_i]_2 \rightarrow [B_i A_{i+1}]_2 [A_{i+1}]_2; 1 \leq i < n$;
$r_{2,n} : [A_n]_2 \rightarrow [BnX]^{-}2[X]_2$;
- rescriere (identifica un subset care se “potrivește” cu complementara - ponderi)
$r_{2,i,j} : v_i v_j \rightarrow v\{= B_i \wedge \neq B_j \wedge = X \vee = B_j \wedge \neq B_i \wedge = X\}, 1 \leq i < j \leq n$
$r_{2,n+1} : X \rightarrow Y$.

Modelul de Tip kP Sistem (2)

- Comunicare (F – nu este solutie; T – solutie)

$r_{2,n+2} : Y \rightarrow (F, 1)\{\geq v1 \ldots \geq v_n \vee \; \neq v_k\}$ – apare cel putin un v_i sau v-uri in $nr \neq k$

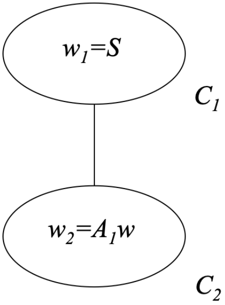
$r_{2,n+2} : Y \rightarrow (T, 1)\{< v_1 \wedge \ldots < v_n \wedge = v_k\}$ – niciun v_i si v-uri in numar de k.

Modelul de Tip kP Sistem (3)

Multiseturile initiale din C_1, C_2, sunt:

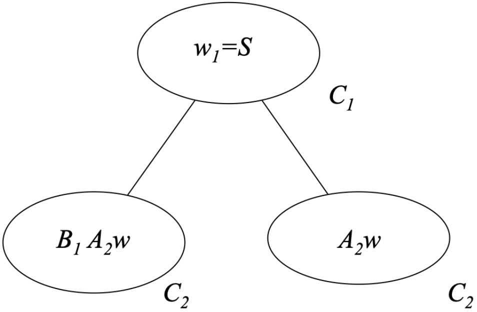
$w_1 = S$

$w_2 = A_1 v_1 k_1 \dots v_n k_n = A_1 w$ (notatie)



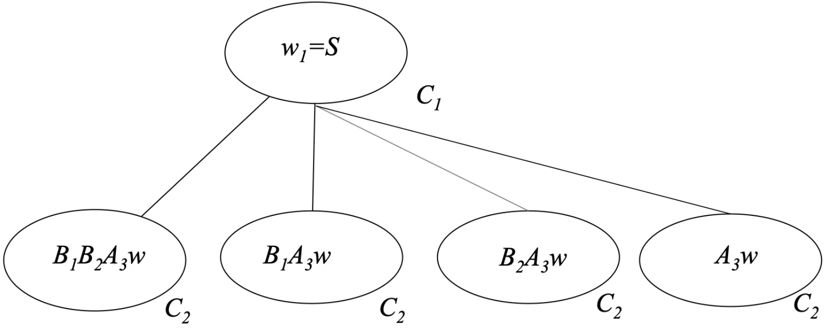
Pasul 1

Aplicam in C_2,
$r2,1 : [A_1]_2 \rightarrow [B_1 A_2]_2 [A_2]_2$
Se obtin doua compartimente C_2 – submulțimile cu max un (1) element.



Pasul 2

Aplicam in C_2,
$r2, 2 : [A_2]_2 \rightarrow [B_2A_3]_2[A_3]_2$
Se obtin patru compartimente C_2 – submulțimile cu max doua (2) elemente.



Pasul n

Aplicam in C_2,
$r2, n : [A_n]_2 \rightarrow [B_n X]2[X]_2$

Se obțin 2^n compartimente C_2 – submultimile cu max n elemente.

Compartimentele C_2 sunt toate legate de C_1.

Pașii finali

Aplicam în C_2:
$r_{2,i,j}$; $r_{2,n+1}$ si apoi una dintre $r_{2,n+2}$ sau $r_{2,n+3}$.

In final $r_{1,1}$ sau $r_{1,2}$ în C_1 trimite rezultatul (yes sau no) in mediu.

AI: Spiking Neural Systems

- **Spiking neural networks** (SNNs) are artificial neural networks that more closely mimic natural neural networks[1].
- They include neurons and synapses, but consider time in their operating model. Neurons transmit information (spikes) in relation to membrane potential (electrical charge).
- The output is decoded in accordance with various rules – frequency of spikes, time-to-first-spike after stimulation, interval between spikes
- SNNs operate producing continuous output

[1] Maass, Wolfgang (1997). "Networks of spiking neurons: The third generation of neural network models". Neural Networks. 10 (9): 1659–1671

Spiking Neural P (SNP) Systems – Definition

$$P = (O, \sigma_1, \ldots, \sigma_n, syn, i_0) \text{ – P system, where:}$$

- O – an alphabet (has a letter, a, called spike)
- $\sigma_1, \ldots, \sigma_n$ – neurons each $\sigma_1 i$ consists of a number n_i of spikes and a set of rules R_i.
- $R_1, \ldots, R_n$ – sets of rules of the form $E/a^c \rightarrow a; d$ or $a^s \rightarrow \lambda$ and a^s does not satisfy E.
- syn – synapses linking neurons.
- i_0 – the output neuron (or environment)

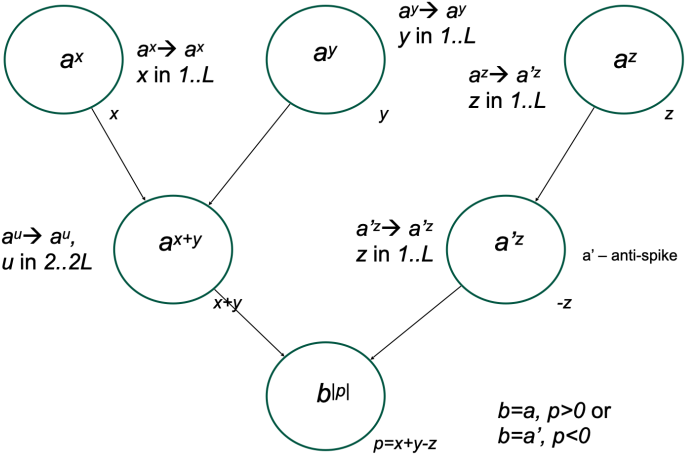
SNP Systems with Anti-spikes and Weights

$$P = (O, \sigma_1, \ldots, \sigma_n, syn, i_0) \text{ – P system, where:}$$

- O – an alphabet (a, called spike and a', anti-spike).
- $\sigma_1, \ldots, \sigma_n$ – neurons each σ_i consists of a number n_i of spikes and a set of rules R_i.
- $R_1, \ldots, R_n$ – sets of rules of the form $E/b^c \to b'; d$ or $b^s \to \lambda$ and b^s does not satisfy E; b and b' are either a or a'.
- syn – synapses linking neurons, with **weights**.
- i_0 – the **output neuron (or environment)**

Ex.: Execute $x + y - z; x, y, z$ natural numbers $\leq L$; and $w_1x + w_2y - w_3z$;

Compute x+y-z; x,y, z – Natural Numbers



SNP Systems and kP Systems

- For the SN P systems mentioned so far equivalent kP systems have been identified
- Complexity results
- SN P systems analysed with methods and tools developed for kP systems

Results

- Classes of SN P systems (standard, extended, with delay, coloured, with polarizations, with anti-spikes etc) mapped into equivalent kP systems
- Complexity aspects investigated
- Examples mapped into suitable kP system tools
- SN P systems with anti-spikes solving key agreement protocols
- SN P systems in cryptography

Medii de Specificare/Simulare

Generale:

- P-Lingua, acopera un set de modele de tipul sisteme de membrane: `http://www.cs.us.es/~ignacio/curso-plingua/doc/paper2.pdf`
- MeCoSim – mediu de dezvoltare atasat la P-Lingua: `http://www.p-lingua.org/mecosim/doc/`

Specifice:

- Infobiotics – aplicatii in biologie: `https://infobiotics.org/`
- Meta PLab – aplicatii scrise in metabolic P systems: `http://mplab.sci.univr.it/index.html`
- kPWorkbench - kP sisteme specificare, simulare, verificare: `https://github.com/Kernel-P-Systems/kPWorkbench`

kPWorkbench

- **kPWorkbench** is a cross-platform framework,
- designed for computational analysis of **kernel P systems**.

- kPWorkbench permits *simulation* and *formal verification* of kernel P system models.
- *kernel P Lingua (kPL)* is the "programming language" in which the models of *kernel P Systems* are written.

kP-Lingua

- Limbaj pentru specificarea kP sistemelor
- inclus in kP Workbench – specificarea modelor, simulare si verificare
- kP Lingua & kPW documents:
  - kP Lingua.pdf (sintaxa limbajului);
  - kPW – README.pdf & kPW – Installation and Execution.pdf – ghiduri de instalare si executie a mediului kPW
  - lista comenzi – Commands.txt, Help.txt

kP Sisteme. Exemplul 1.

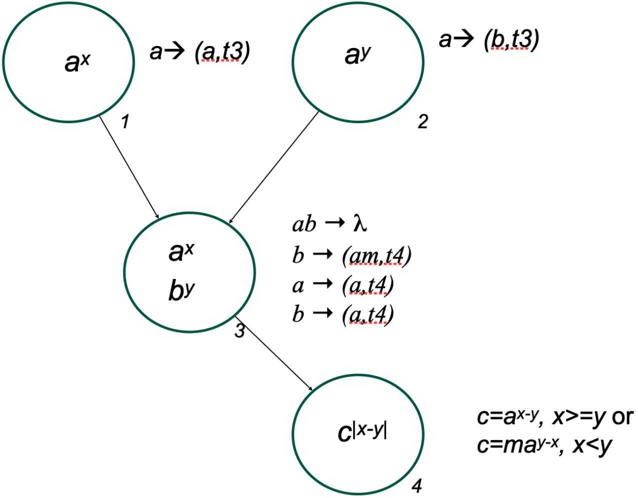
Scaderea a doua numere intregi positive, reprezentare unara. kP sistemul are componentele C_1, C_2, C_3, C_4 si tipurile t_1, t_2, t_3, t_4.

t_1 cu regula $a \to (a, t_3)$ aplicata conform paralelismului maxim
t_2 cu regula $a \to (b, t_3)$ aplicata conform paralelismului maxim
t_3 cu regulile de mai jos aplicate intr-o secventa de 3 blocuri
$\{ab \to \lambda\}^T \{b \to (am, t_4)\}^C \{a \to (a, t_4), b \to (a, t_4)\}^T$
t_4 este tipul vid.

Multiseturile initiale sunt n elemente a in C_1 si m elemente a in C_2.

C_1, C_2 sunt legate de C_3 care este legat de C_4.
Ideea modelului. Cele n elemente a in C_1 si m elemente a in C_2 trec in C_3 sub forma de a si b. In C_3 se face diferența, cu semn, care trece in C_4.

Exemplul 1. Calculeaza x-y



Figura

kP Lingua. Exemplul 1.

```
type t1 {
  max {
    a -> a (t3).
  }
}

type t2 {
  max {
    a -> b (t3).
  }
}

type t3 {
  max {
    a, b -> {}.
  }
  choice {
    b -> {a, m} (t4).
  }
  max {
    a -> a (t4).
    b -> a (t4).
  }
}

type t4 {
}

C1 {3a} (t1) – C3 {} (t3).
C2 {5a} (t2) – C3.
C3 - {} (t4).
```

Recap SN P: P Sisteme Neuronale

- Modelul numit Spiking Neural P system
- Compartimentele contin multiseturi peste un alphabet $\{a\}$
- Regulile sunt de rescriere sau stergere, cu garzi expresii regulate
- Doar o regula pe compartiment este executata
- Compartimente de intrare si iesire
- Intrari din afara sistemului in compartimentele de intrare
- Ilustram translatarea P sistemelor neuronale in kP sisteme

Recap SN P: P Sisteme Neuronale. Exemplul 2.

Adunarea a doua numere intregi positive in baza 2.
P sistemul are 3 componente (neuroni): $\sigma_1, \sigma_2, \sigma_3$

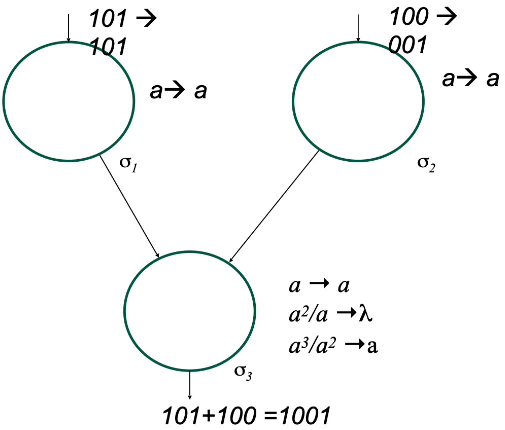
Regulile neuronilor:
$\sigma_1, \sigma_2 : a \to a$
$\sigma_3 : a \to a, a^2/a \to \lambda, a^3/a^2 \to a$

Multiseturile initiale sunt vide.

In neuronii σ_1, σ_2 sunt introduse numerele in baza 2, fiecare in ordinea inversa a cifrelor (exemplu 101, 100). In neuronul σ_3 se calculeaza suma, iar rezultatul este trimis in exterior (mediu).

Conventie. 0 este interpretat ca lipsa simbolului a (spike) la intrare/iesire, iar 1 ca prezenta acestuia.

Recap SN P: Exemplul 2. Calculeaza x+y



kP Lingua. Exemplul 2 (translatat in kPL !!).

```
type In1 {
  choice {
    =a:  a  -> a (S3).
  }
}

type In2 {
  choice {
    =a:  a  -> a (S3).
  }
}

type S3 {
  choice {
    =a:  a -> a1 (S4).
    =2a: a -> a0 (S4).
    =3a: 2a -> a1 (S4).
  }
}

in1 {} (In1).
in2 {} (In2).
c3 {} (S3).

in1 - c3.
in2 - c3.
```

Problema Partiției (Translatat in kPL !!); Variabile Indexate

```
#define n = 4, k = 10
#define weigths = [2, 3, 4, 9]

type T0 {
}
type T1 {
        max {
                >=t : s -> yes (T0) .
                >=f & <t : s -> no (T0) .
        }
}
type T2 {
        max {
                a$i$ -> [b$i$, a$i+1$][a$i+1$] . : 1<=i<n
                a$n$ -> [b$n$, x][x] . : n<=i<=n
        }
        max {
                =b$i$ & !=b$j$ & =x | =b$j$ & !=b$i$ & =x : v$i$, v$j$ -> v . : 1<=i<=n, 1<=j<=n, i<j
        }
        max {
                x -> y .
                >=v1 | >=v2 | >=v3 | >=v4 | !=$k$v : y -> f (T1) . : n<=i<=n
                <v1 & <v2 & <v3 & <v4 & =$k$v : y -> t (T1) . : n<=i<=n
        }
}

t0 {} (T0) .
t1 {s} (T1) .
t2 {a1} (T2) .
t2 += {$weigths[i]$a$i$} . : 1<=i<n
t0 - t1 - t2 .
```

Problema Partiției (Translatat in kPL !!); Fără Variabile Indexate

```
type T0 {}
type T1 {
        max {
                >=t : s -> yes (T0) .
                >=f & <t : s -> no (T0) .
        }
}
type T2 {
        max {
                a1 -> [b1, a2][a2] .
                a2 -> [b2, a3][a3] .
                a3 -> [b3, a4][a4] .
                a4 -> [b4, x][x] .
        }
        max {
                =b1 & !=b2 & =x | =b2 & !=b1 & =x : v1, v2 -> v .
                =b1 & !=b3 & =x | =b3 & !=b1 & =x : v1, v3 -> v .
                =b2 & !=b3 & =x | =b3 & !=b2 & =x : v2, v3 -> v .
                =b1 & !=b4 & =x | =b4 & !=b1 & =x : v1, v4 -> v .
                =b2 & !=b4 & =x | =b4 & !=b2 & =x : v2, v4 -> v .
                =b3 & !=b4 & =x | =b4 & !=b3 & =x : v3, v4 -> v .
        }
        max {
                x -> y .
                >=v1 | >=v2 | >=v3 | >=v4 | !=10v : y -> f (T1) .
                <v1 & <v2 & <v3 & <v4 & =10v : y -> t (T1) .
        }
}
t0 {} (T0) .
t1 {s} (T1) .
t2 {a1, v1, 5v2, 10v3, 4v4} (T2) .
t0 - t1 - t2 .
```

kP Sisteme: Simulare si Verificare

- S-a ilustrat modelarea si simularea kP sistemelor (kP-Lingua si mediul kPWorkbench)
- Corectitudinea modelor – model checking
- Model checking – verificarea formala a unui model utilizand asertiuni (proprietati) exprimate intr-o logica (temporala). Larga utilizare in inginerie (safety critical systems). Ex: Spin, NuSMV
- Specificare de tip model checking: model formal si un set de asertiuni logice care sa fie verificate pentru model.
- kPW: (1) simulare; (2) verificare model checking – translatarea kP-Lingua & proprietati (limbaj natural) in model si asertiuni (proprietati) exprimate ca model checker

Unary encoding

- Another form of encoding natural numbers.
- The unary numeral system is the simplest numeral system to represent natural numbers[2].
- For $X \in \mathbb{N}$, and x a symbol representing the natural number 1, then the unary encoding of X is 1 is repeated X times [1].
- i.e., for $X = 7$, let x be the symbol for 1,
- then x^7 is the unary representation of $X = 7$.
- Note: x^7 is a multiset over set $\mathcal{X} = \{x\}$.

[2] Wikipedia (Unary numeral system)

Sum & Diff

- Sum($a\prime$, $b\prime$), Diff($a\prime$, $b\prime$), where $a\prime = 7, \quad b\prime = 3$:
- Objects a, b represents the unary encoding for $a\prime$ and $b\prime$.
- Sum($a\prime$, $b\prime$) $\equiv$ Sum(a^7, b^3) = s^{10}.
- Diff($a\prime$, $b\prime$) $\equiv$ Diff(a^7, b^3) = a^4.
- for $a\prime = 4, \quad b\prime = 9$:
- Diff($a\prime$, $b\prime$) $\equiv$ Diff(a^4, b^9) = b^5.

Concluzii

- Modelarea si relatia cu Ingineria Software
- Modele cu rescriere. Sisteme cu Membrane
- Cum se modeleaza cu Sistemele cu Membrane (exemple)
- Aplicatii: probleme NP-complete (partition problem); spiking neural P systems translatate in kernel P systems
- Alte topici: (1) verificarea si testarea folosind sistemele cu membrane; (2) instrumente de modelare, verificare si testare

Q & A

Thank you for your attention!

References

1. kPworkbench, https://github.com/Kernel-P-Systems/kPWorkbench
2. Gheorghe, M., Ipate, F., Dragomir, C., Mierla, L. and Valencia-Cabrera, L., 2012. *Kernel P systems*. Membrane Computing, Tenth Brainstorming Week, BWMC, pp.153-170.