Sisteme cu membrane (2)

Prof Emeritus Marian Gheorghe

Department of Computer Science
University of Bradford
United Kingdom

Sumar

- Modelare si verificare kP sisteme
- Specificatii formale bazate pe kP sisteme si simulare (kPWorkbench)
- Mapare P sisteme neuronale in kP sisteme
- Verificare kP sisteme folosind model checking

Sisteme cu Membrane

Un model de calcul inspirat (bazat) pe structura si functionarea celulei vii (**Membrane or P systems**)

Trei caracteristici esentiale ale Sistemelor cu Membrane:

- ***structura*** (ierarhica, dar nu numai) de **membrane** delimitand **regiuni/compartimente** (*arbore*)
- ***multiseturi de obiecte*** si
- ***seturi finite de reguli*** asociate compartimentelor

Un P sistem **calculeaza**, evoluand de la o configuratie (stare) la alta, aplicand regulile, in compartimente, utilizand strategia paralelismului maxim

Regulile pot rescrie sau muta obiecte dintr-un compartiment in altul, dar pot modifica structura (introducand/stergand compartimente)

Sistem cu Membrane – Definitie

$$P = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, i_0) \text{ - P system}$$

where

O – an alphabet (finite set)

μ – a membrane structure with n membranes (regions)

$w_1, \ldots, w_n$ – multisets over O; w_i – initial values

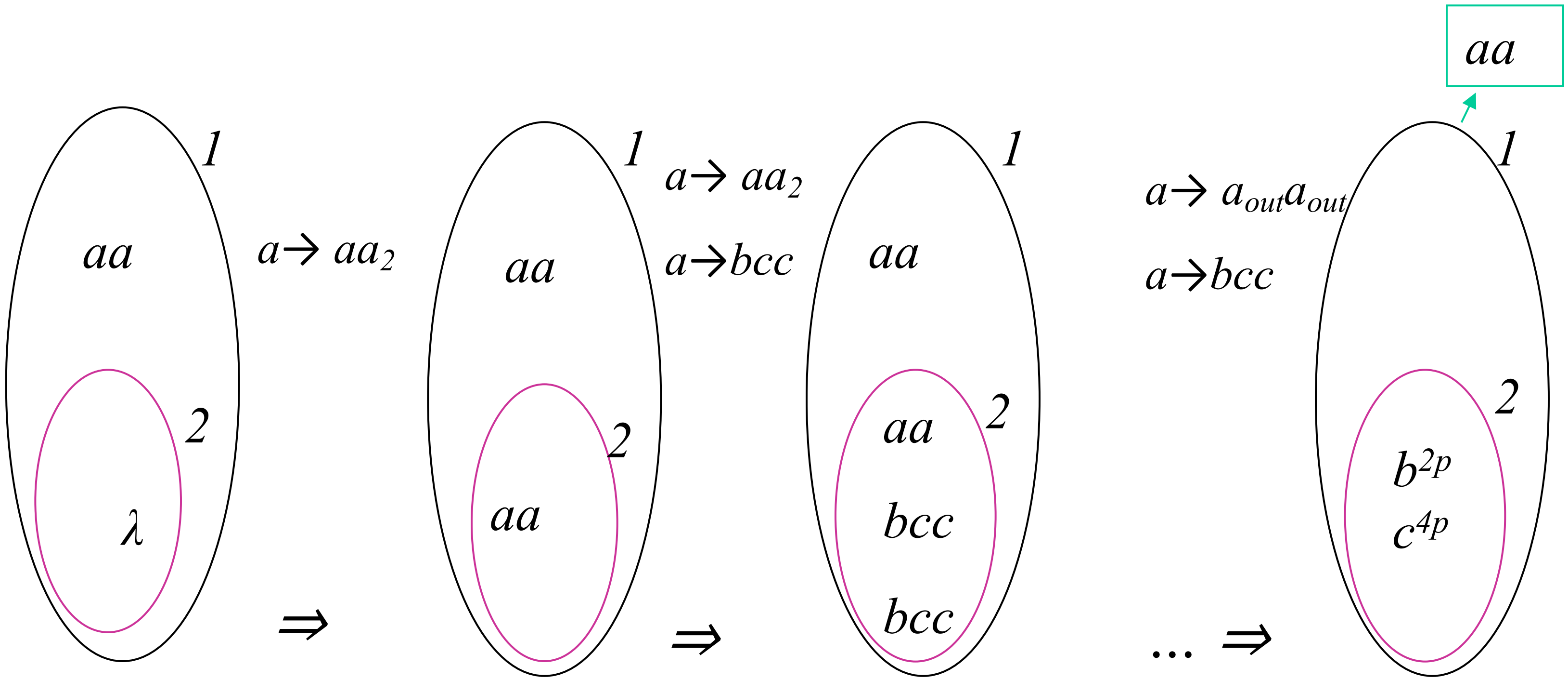
$R_1, \ldots, R_n$ – sets of rules

i_0 – the output cell (or environment)

R_i – evolution and communication rules: $v \rightarrow w$; v, w – strings over O + some indications of target regions in w

Exemplu – Doua compartimente

$P = (\{a,b,c\}, [_1[_2]_2]_1, aa, \lambda, \{a \to aa_2, aa \to a_{out}a_{out}\}, \{a \to bcc\}, 2)$



$w = b^{2p}c^{4p}, p \geq 0$

Caracteristici:
structura ierarhica (arbore); reguli de rescriere si comunicare; nedeterminism; paralelism maxim. Rezultatul: multiset (*b: 2p & c: 4p*)

Kernel P System – Introducere Informala

- 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 (compartimente); creare/stergere legaturi intre compartimente)

- Fiecare compartiment are un tip din care deriva, iar tipul compartimentului defineste setul de reguli. Fiecare instanta are propriul multiset initial.

Definition (Kernel P systems)

A kP system of degree n is a tuple $k\Pi = (A, \mu, C_1, \ldots, C_n, i_0)$, where

- A is a finite set of elements called objects;
- μ defines the membrane structure, which is a graph, (V, L), where V is a set of vertices representing components (compartments), and L is a set of edges, i. e., links between components;
- $C_i = (t_i, w_{i,0})$, $1 \leq i \leq n$, is a compartment of the system consisting of a *compartment type*, t_i, from a set T and an initial multiset, $w_{i,0}$ over A; the type $t_i = (R_i, \delta_i)$ consists of a set of evolution rules, R_i, and an execution strategy, δ_i;
- i_0 is the output compartment where the result is obtained.

Rule types

Within the kP systems framework, the following types of evolution rules have been considered so far:

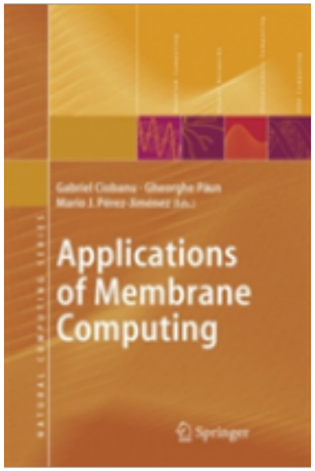
- *rewriting and communication* rules
- *structure changing* rules: membrane division, membrane dissolution, link creation and link destruction rules.

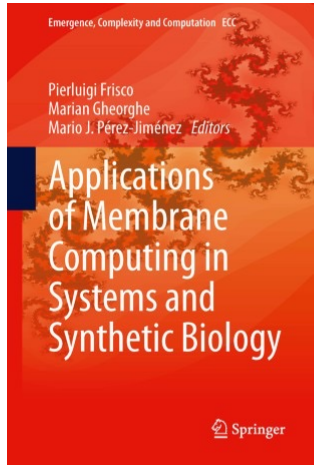
Complex *guards* can be incorporated, using multisets over A with relational and Boolean operators.

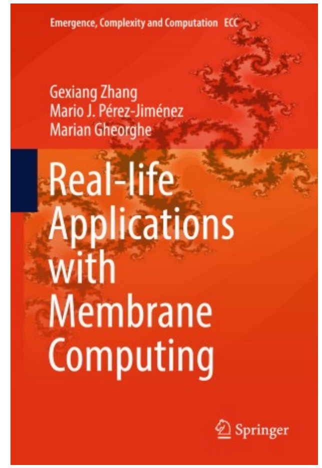
Execution strategies: for each compartment type / block

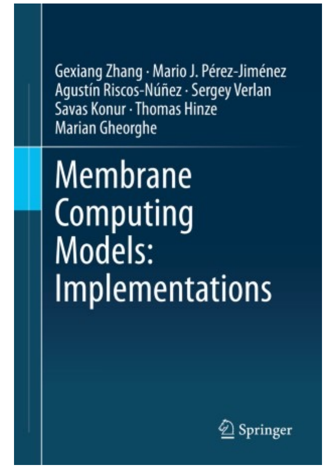
- *sequential*
- *choice*
- *arbitrary*
- *maximal parallel*

Aplicatii ale P sistemelor









- Diferite arii si domenii
- Implica specificare (modelare) si simulare, dar si analiza (verificare), testare

Medii de Specificare/Simulare

- Generale

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

- Specifice

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

kP-Lingua

- Limbaj pentru specificarea kP sistemelor

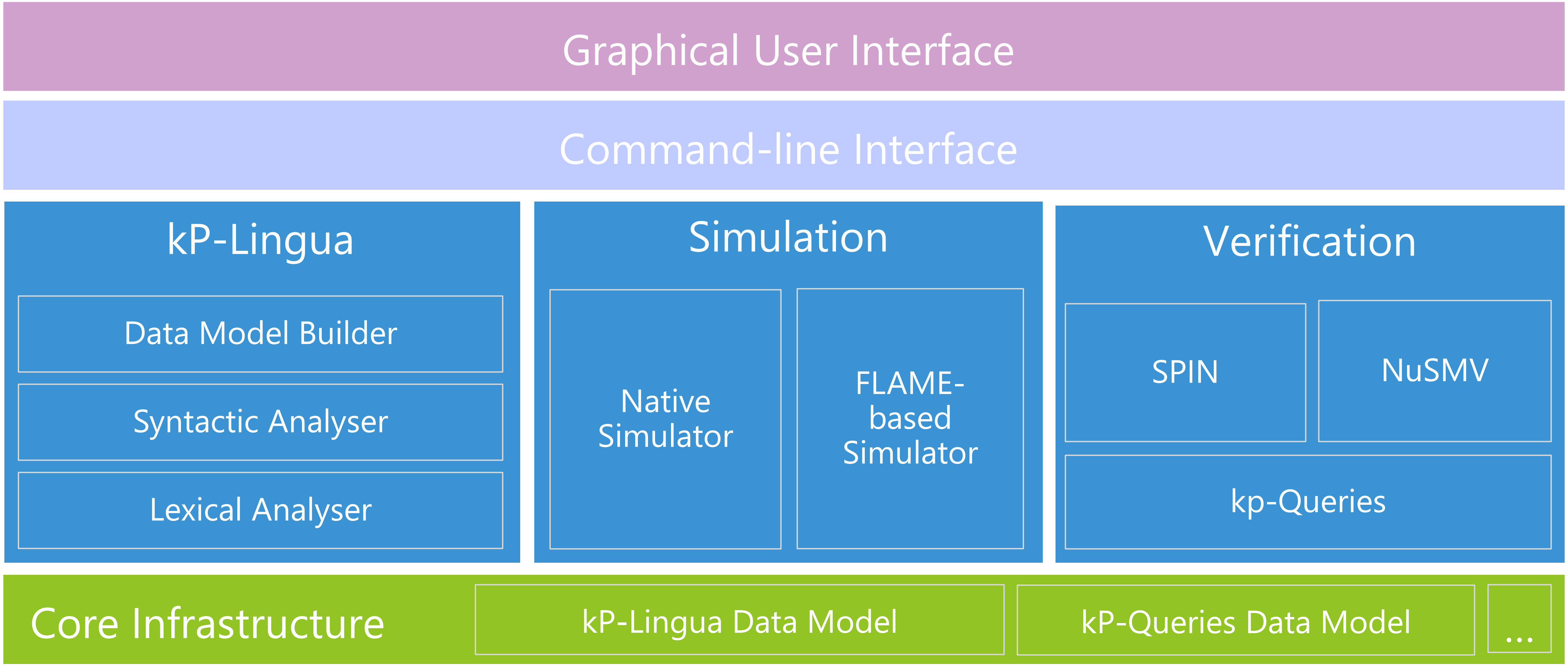
- inclus in kP Workbench – specificarea modelor, simulare si verificare

- 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

Insereaza

- slideuri: 8, 10, 12, 18 – Laurentiu

KPWorkbench tool stack



kP-Lingua constructs

```
type C1 {
    2a, 3b -> c .
    arbitrary {
        >= 2c & > 2b : b, c -> a .
    }
    choice {
        b -> 2b .
        < 3b : b -> 3b .
    }
    max {
        a -> a, a(C2), {a, 2b}(C3) .
    }
    2c -> - (C2) .
    2b -> \- (C2) .
    = 5a : a -> [3a, 3b](C1) [3b](C2) .
}
```

- Compartment type definitions
- Rewriting and communication rules
- Rule guards
- Membrane division
- Link creation and destruction
- Sequential, arbitrary, choice and maximal parallelism execution strategies

```
m1 {2x, b} (C1) .
m2 {x} (C2) .
```

- Membrane instantiations

```
m1 - m2 .
```

- Link between compartments

kP-Queries

```
ltl: eventually m1.a > 0;
```

SPIN ⟹

```
ltl prop1 { <> (c[0].x[a_] > 0 && state == step_complete) }
```

NuSMV ⟹

```
LTLSPEC F (m1.a > 0)
```

```
ctl: m2.a = 1 followed-by m3.a = 1;
```

NuSMV ⟹

```
SPEC AG (m2.a = 1 -> EF m3.a = 1)
```

Pattern	**kP-Query**	**LTL**	**CTL**
Next	next p	$\mathrm{X}\ p$	$\mathrm{EX}\ p$
Existence	eventually p	$\mathrm{F}\ p$	$\mathrm{EF}\ p$
Absence	never p	$\neg(\mathrm{F}\ p)$	$\neg(\mathrm{EF}\ p)$
Universality	always p	$\mathrm{G}\ p$	$\mathrm{AG}\ p$
Recurrence	infinitely-often p	$\mathrm{G\ F}\ p$	$\mathrm{AG\ EF}\ p$
Steady-state	steady-state p	$\mathrm{F\ G}\ p$	$\mathrm{AF\ AG}\ p$
Until	p until q	$p\ \mathrm{U}\ q$	$\mathrm{A}\ (p\ \mathrm{U}\ q)$
Response	p followed-by q	$\mathrm{G}\ (p \rightarrow \mathrm{F}\ q)$	$\mathrm{AG}\ (p \rightarrow \mathrm{EF}\ q)$
Precedence	p preceded-by q	$\neg(\neg p\ \mathrm{U}\ (\neg p \wedge q))$	$\neg(\mathrm{E}\ (\neg p\ \mathrm{U}\ (\neg p \wedge q)))$

Square numbers generation

```
type main {
    max {
        = t: a -> {} .
        < t: a -> a, 2b, s, i .
        < t: a -> a, s, t, i .
        < t: b -> b, s .
    }
}

m {a} (main) .
```

```
/* LTL Properties */
ltl: always m.t <= 1;
ltl: steady-state (m.a = 0 implies m.t = 1);
ltl: never m.s = 15;

ltl: always m.s == m.i * m.i; // Invariant


/* CTL Properties */
ctl: eventually m.a = 0;
ctl: eventually m.t = 1;
ctl: m.a = 0 preceded-by m.t = 1;
ctl: m.a > 0 followed-by m.a = 0;
ctl: m.t = 1 followed-by m.a = 0;
ctl: always m.t <= 1;
ctl: never m.s = 15;
```

kP Sisteme. Exemplul 1.

Scaderea a doua numere intregi positive. kP sistemul are 4 componente: C_1, C_2, C_3, C_4. Tipurile associate sunt t_1, t_2, t_3, t_4.

t_1 cu regula $a \rightarrow (a,t_3)$ aplicata cf paralelismului maxim

t_2 cu regula $a \rightarrow (b,t_3)$ aplicata cf paralelismului maxim

t_3 cu regulile de mai jos aplicate

$\{ab \rightarrow \varepsilon\}^{\mathrm{T}}\{b \rightarrow (am,t_4)\}^{\mathrm{C}}\{a \rightarrow (a,t_4), b \rightarrow (a,t_4)\}^{\mathrm{T}}$; o secventa de 3 sub-executii, unde T este parallelism maxim, iar C este alegere (choice)

t_4 este tipul vid.

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

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 diferenta, cu semn, care trece in C_4

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).
```

Exemplul 1 - Simulare

- Rezultatele in fisierele Subtraction-sign-Minus.out si Subtraction-sign-Plus.out
- **Obs**. Notatiile in fisier sunt diferite de cele de mai sus

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

Spiking Neural P systems

- Class of P systems inspired by the way neurons interact by sending spikes
- Represent an AI paradigm also inspired by the same biological phenomena
- SNPs developed both on
  - *theoretical aspects*: many variants, with computational power and complexity issues studies
  - many *significant applications*
- Several survey papers
- In this research we are interested in the modelling aspects and connections with kP systems

Definition (Spiking neural P systems – SNP)

An SN P system of degree m, $m \geq 1$, is a tuple $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$, where

- $O = \{a\}$ is a singleton alphabet (a is called spike);
- $\sigma_i, 1 \leq i \leq m$, are neurons, $\sigma_i = (n_i, R_i), 1 \leq i \leq m$, where
  - $n_i \geq 0$ is the number of spikes in σ_i;
  - R_i is a finite set of rules of the following forms:
    - Type (1): spiking rules
      $E/a^c \rightarrow a^p$; where E is a regular expression over $\{a\}$, and $c \geq 1$, $p \geq 1$, such that $c \geq p$;
    - Type (2): forgetting rules
      $a^s \rightarrow \lambda$, for $s \geq 1$, such that for each rule $E/a^c \rightarrow a^p$ of type (1) from R_i, $a^s \notin L(E)$;
- $syn = \{(i,j) | 1 \leq i, j \leq m, i \neq j\}$ (synapses between neurons);
- $in, out \in \{1, \ldots .m\}$ indicate the input and output neurons respectively.

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
σ_1, σ_2: $a \to a$
σ_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.

kP Lingua. Exemplul 2 (translatat).

```
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.
```

Specificarea completa in Add.kpl; lipsesc componentele intrare si iesire
Rezultatele in Add-101add100.out.txt

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

Exemplul 3 – Pasii Implicati.

Broadcasting. Trimiterea unui mesaj la toate componentele unui sistem, exprimat ca un arbore de compartimente.

Simularea.
- Broadcasting.kpl – modelul
- Broadcasting-1.out, Broadcasting-2.out – executii nedeterministe

Verificarea.
- Broadcasting.kpl – modelul
- Broadcasting.kpq – asertiuni (proprietati) limbaj natural

Broadcasting_verification.smv – translatare model + asertiuni (LTL, CTL)
- Broadcasting_verification.txt – rezultate verificare (NuSMV)

Concluzii

- Simularea si verificarea kP sistemelor

- Aplicatii (complexe) in diferite domenii, medii de specificare si verificare

- Directii de cercetare si dezvoltare: simulare eficienta (arhitecturi paralele), testare, mapare in alte modele

- Oportunitati: proiecte bazate pe folosirea modelelor, simulare si verificare; cercetare (doctorat)