Sisteme cu membrane (2)

Prof Emeritus Marian Gheorghe

Department of Computer Science
University of Bradford
United Kingdom

Sumar

- kernel P sisteme
- Medii de simulare pentru sistemele cu membrane
- kP-Lingua si kPWorkbench
- Simulare si verificare folosind kPWorkbench
- Exemple in kP-Lingua si simulare in kPWorkbench

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

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

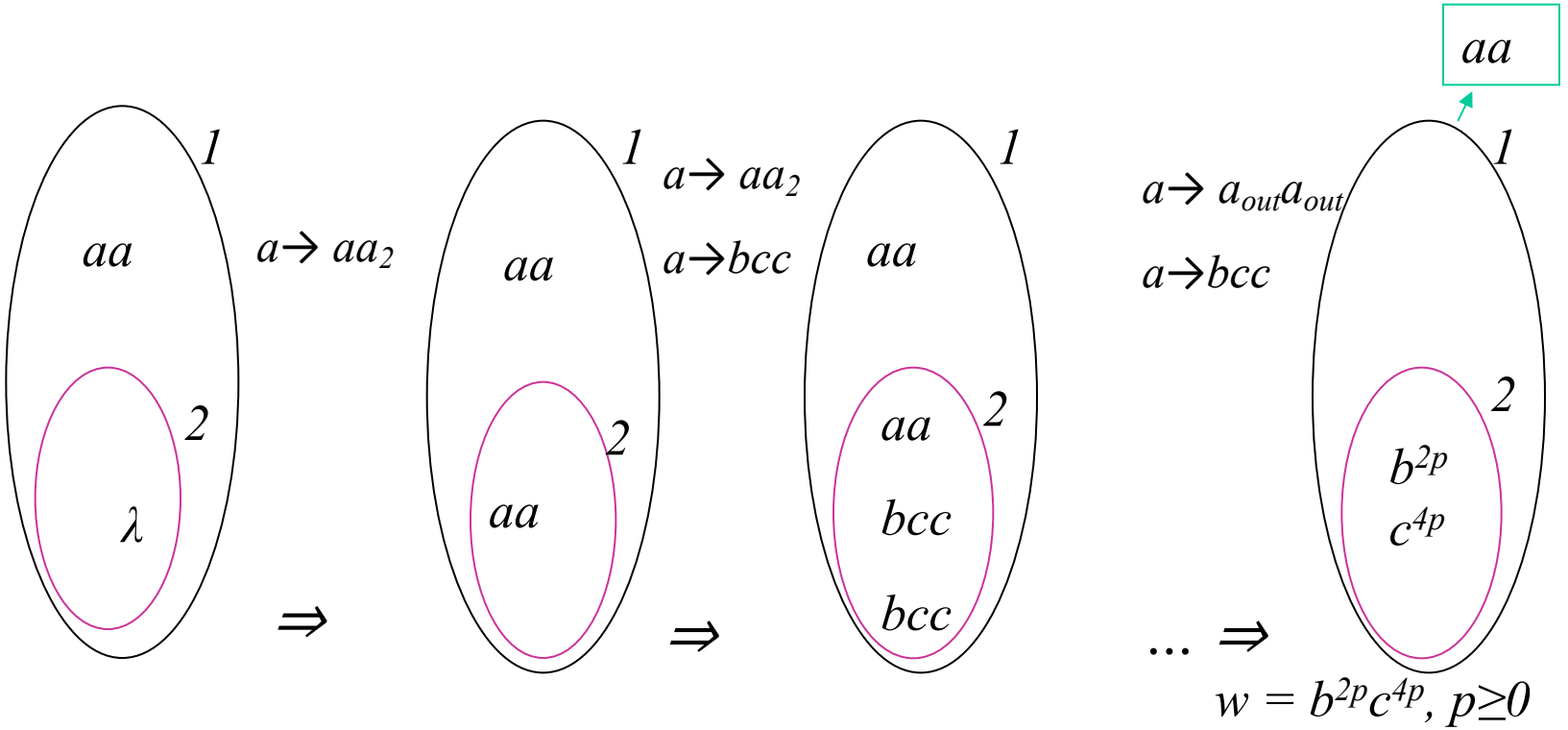
$R_1, \ldots, R_n$ – sets of rules

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

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



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:

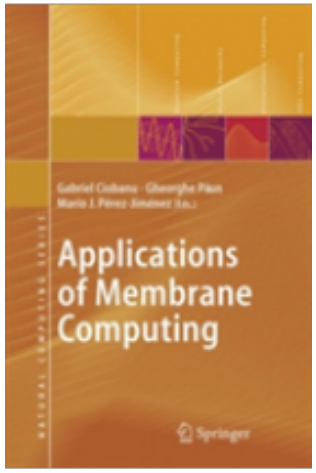
- *rewriting and communication* rules
- *structure changing* rules: membrane division, membrane dissolution, link creation and link destruction rules.

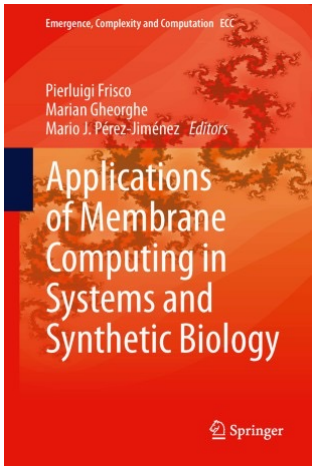
Complex *guards* can be incorporated, using multisets over A with relational and Boolean operators.

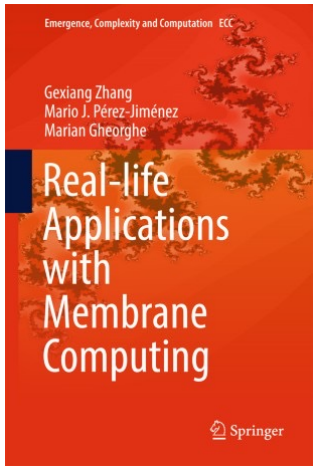
Execution strategies: for each compartment type / block

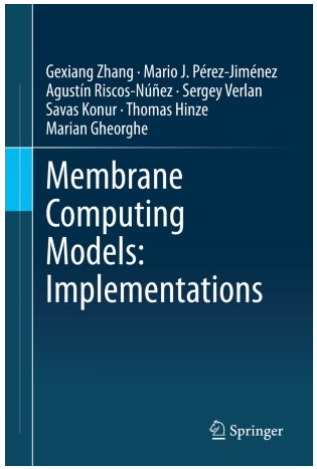
- *sequential*
- *choice*
- *arbitrary*
- *maximal parallel*

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

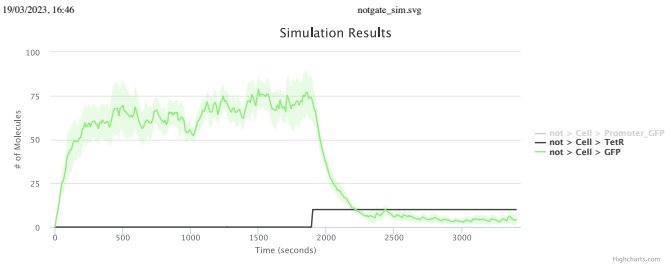
1. Infobiotics – aplicatii in biologie: https://infobiotics.org/
2. Meta PLab – aplicatii scrise in metabolic P systems: http://mplab.sci.univr.it/index.html
3. kPWorkbench - kP sisteme specificare, simulare, verificare: https://github.com/Kernel-P-Systems/kPWorkbench

P-Lingua. Exemplu: transition P system.

```
@model<transition>
def main()
{
  @mu = [[[]'3 []'4]'2]’1;
  @ms(3) = a,f;

  [a --> a,bp]’3;
  [a --> bp,@d]’3;
  [f --> f*2]’3;

  [bp --> b]’2;
  [b []'4 --> b [c]'4]’2;
  [f*2 --> f ]’2;
  [f --> a,@d]’2;
}
```

IBL. Exemplul – NOT Gate.

```
define Cell typeof CELL() {

  // Species
  Promoter_GFP = MOLECULE(concentration=2.0 molecules)
  TetR = MOLECULE(concentration=0.0 molecules)
  GFP = MOLECULE(concentration=0.0 molecules)

  // Rules
  RULE Degradation_GFP : GFP ->
  Degradation_GFP.forwardRate = '0.0075*[GFP]'

  RULE R_abstracted_production_Promoter_GFP : -> GFP + GFP + GFP + GFP + GFP + GFP + GFP + GFP + GFP + GFP
  R_abstracted_production_Promoter_GFP.forwardRate =
  '0.05*2.0*0.033*30.0/((1+0.033*30.0)+(0.5*[TetR])^2.0)'

  EVENT [TetR = 10 molecules] WHEN [TIME >= 1900 s]
}

define not typeof REGION () {
  CELL Cell = new Cell()
}
```

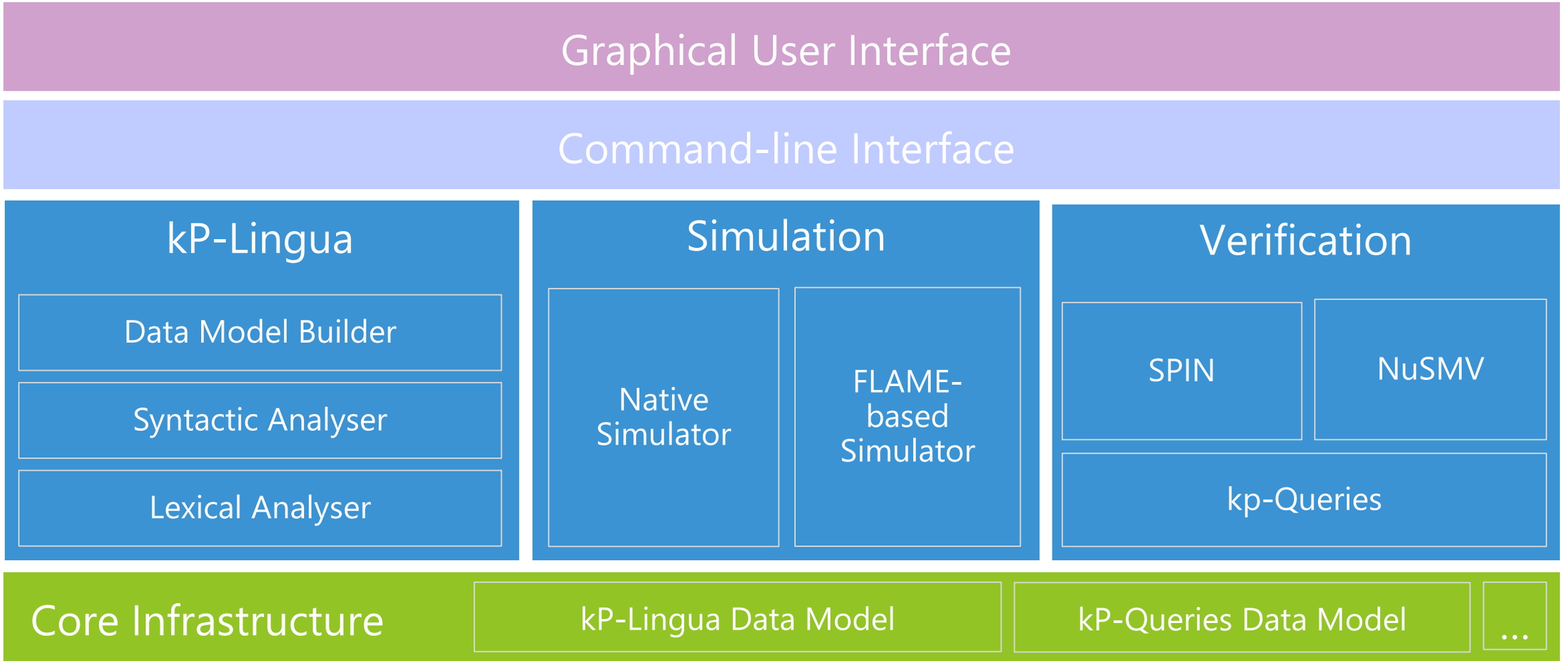


kP-Lingua

- Limbaj pentru specificarea kP sistemelor

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

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) .
```

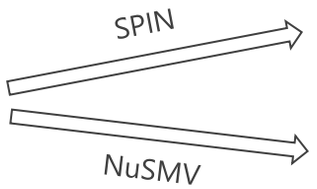
- Membrane instantiations

```
m1 - m2 .
```

- Link between compartments

kP-Queries

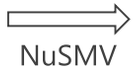
```
ltl: eventually m1.a > 0;
```




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

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

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




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

Pattern	kP-Query	LTL	CTL
Next	next p	X *p*	EX *p*
Existence	eventually p	F *p*	EF *p*
Absence	never p	¬(F *p*)	¬(EF *p*)
Universality	always p	G *p*	AG *p*
Recurrence	infinitely-often p	G F *p*	AG EF *p*
Steady-state	steady-state p	F G *p*	AF AG *p*
Until	p until q	*p* U *q*	A (*p* U *q*)
Response	p followed-by q	G (*p* → F *q*)	AG (*p* → EF *q*)
Precedence	p preceded-by q	¬(¬*p* U (¬*p* ∧ *q*))	¬(E (¬*p* U (¬*p* ∧ *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
$\{ab \rightarrow \varepsilon\}^{T}\{b \rightarrow (am,t_4)\}^{C}\{a \rightarrow (a,t_4),\ b \rightarrow (a,t_4)\}^{T}$; 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).
```

Concluzii

- Kernel P sisteme
- kP-Lingua si kPWorkbench
- Exemple

- **Proiecte** bazate pe kP sisteme (modelare, simulare)
- **Directii de cercetare** (master, doctorat): simulare eficienta (arhitecturi paralele), verificare, testare