

# *LMNtal* : a language model with links and membranes

---

Kazunori Ueda, Waseda Univ.  
(Joint work with Norio Kato)

June 2004

# $\mathcal{LMNtal}$ (pronounce: “*elemental*”)

---

$\mathcal{L}$  = logical links

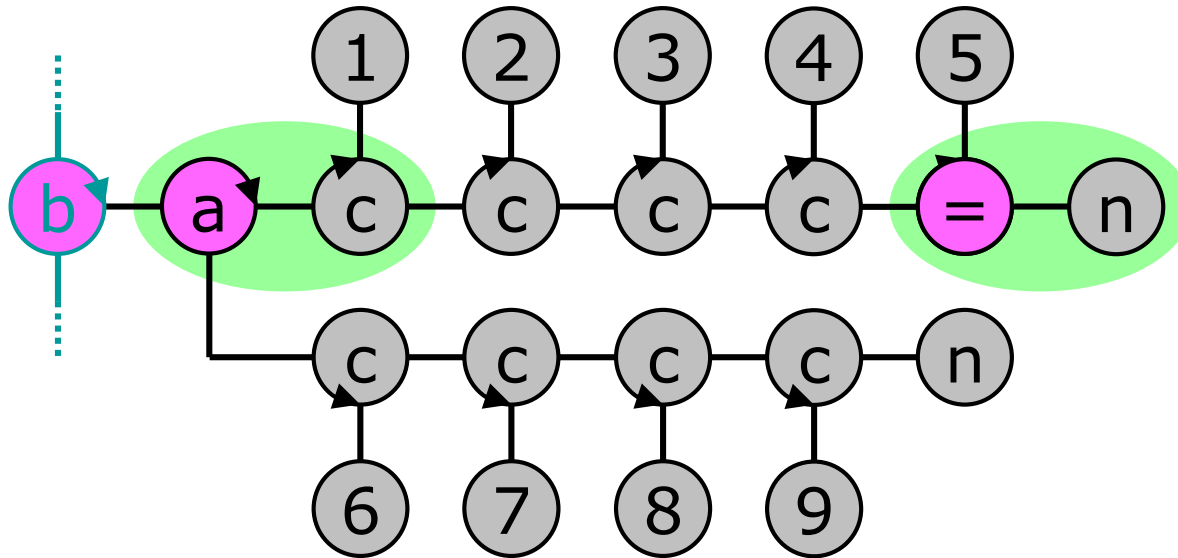
$\mathcal{M}$  = multisets/membranes

$\mathcal{N}$  = nested nodes

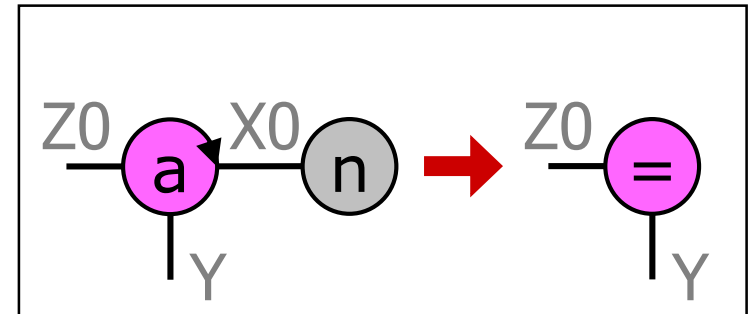
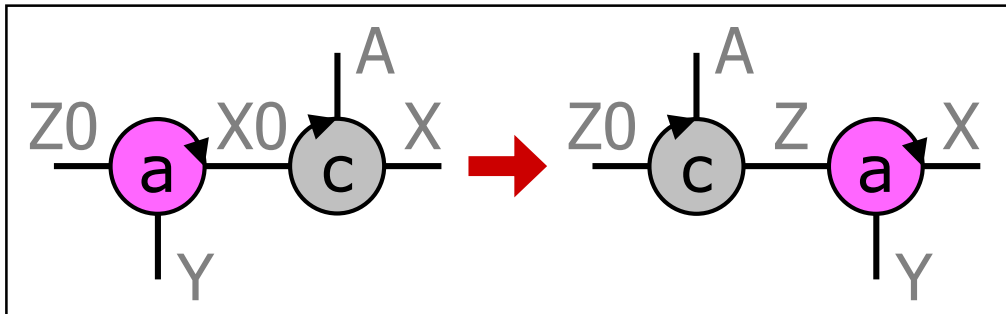
$ta$  = transformation

$\mathcal{I}$  = language

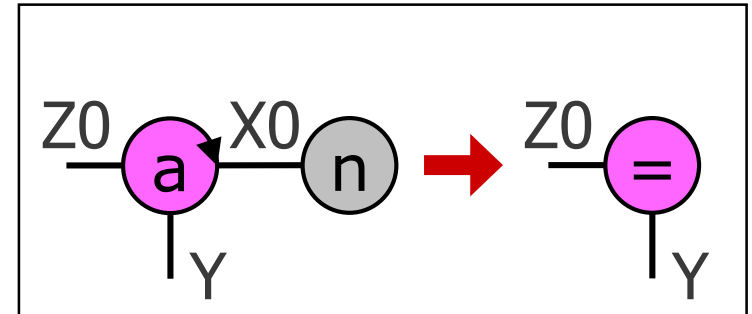
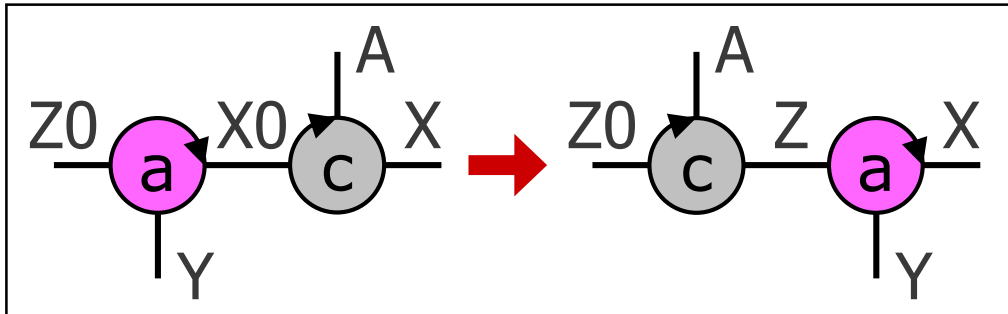
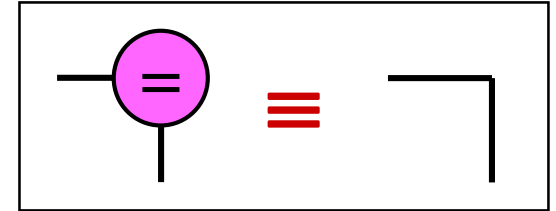
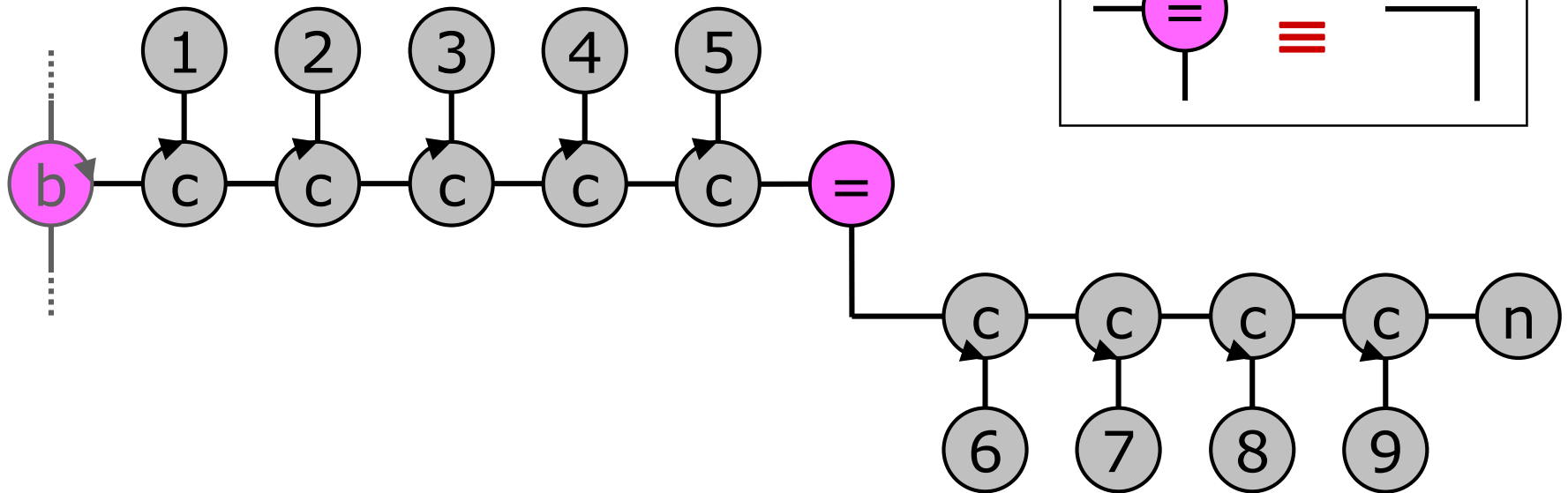
# Example 1: append



a : append  
 c : cons  
 n : nil



# Example 1: append



# append in $\mathcal{LMNtal}$

```
append(X0,Y,Z0), c(A,X,X0) :-  
    c(A,Z,Z0), append(X,Y,Z)  
append(X0,Y,Z0), n(X0) :- Y=Z0
```

★ cf. Guarded Horn Clauses (GHC) version

```
append(X0,Y,Z0) :- X0=[A|X] |  
    Z0=[A|Z], append(X,Y,Z).  
append(X0,Y,Z0) :- X0=[] | Y=Z0.
```

- No distinction between append and c(ons), though they could be implemented differently

# Design goals of $\mathcal{LMNtal}$

---

## ■ A simple language model

- ★ A “Turing Machine” for *universal* computing environments (from wide-area to nanoscale)
- ★ A practical programming language

## ■ Unifying and scalable

- ★ Unify various concepts about programming
- ★ Cover various computational platforms

## ■ Easy to program

- ★ Computation can be viewed as diagram transformation

# What do we mean by “Unifying” ?

---

- Treat processes, messages, and data uniformly
- Handle process structures and data structures uniformly and easily
  - ★ Most declarative languages are awkward in handling non-tree data structures
  - ★ Pointers in procedural languages are error-prone
- Treat synchronous communication and asynchronous communication uniformly

# Background

---

- **Concurrent Logic Programming** (early 1980's)
  - ★ Channel mobility using logical variables
  - ★ Various type systems (including linearity) and implementation experiences
- **Concurrent Constraint Programming** (late 1980's)
  - ★ Generalization of data domains (FD, multisets, . . . )
- **CHR (Constraint Handling Rules)** (early 1990's)
  - ★ Allows multisets of goals in rule heads
  - ★ An expressive multiset rewriting language
  - ★ Many applications (esp. constraint solvers)
  - ★ Lacks reaction control mechanisms such as termination detection and hierarchies



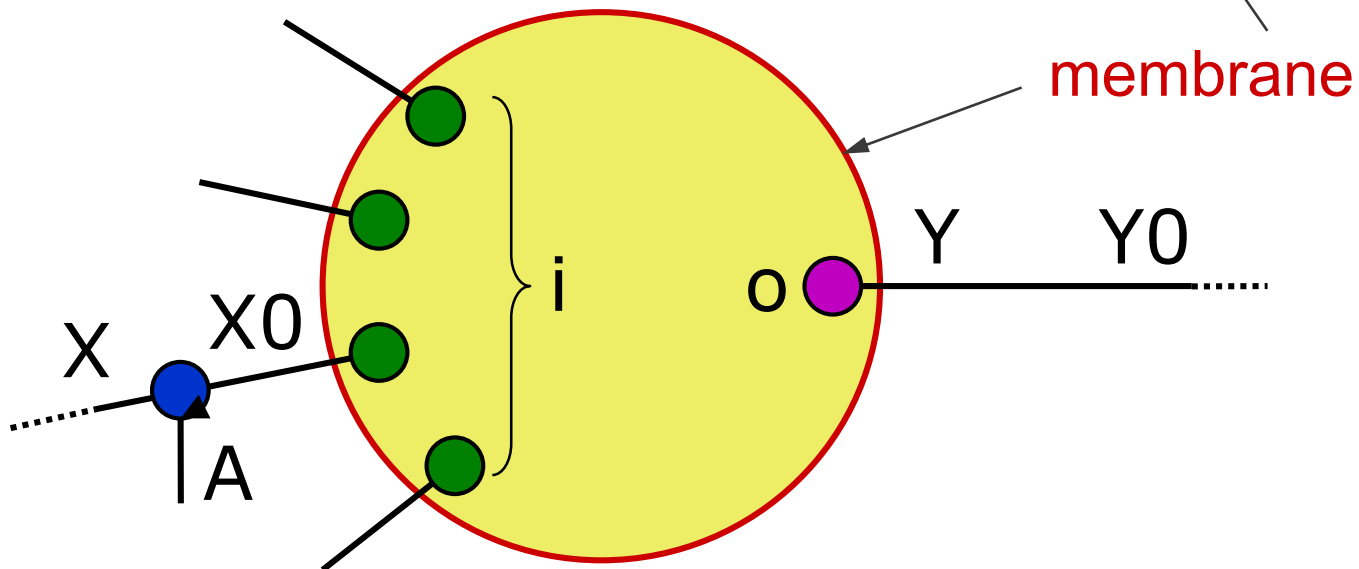
# Models and languages with multisets and **symmetric join**

---

- Petri Nets (1962)
- Production Systems and RETE match
- Graph transformation formalisms
- CCS, CSP
- Concurrent logic/constraint programming
- Linda
- Linear Logic languages
- Interaction Net
- Chemical Abstract Machine, reflexive CHAM, Join Calculus
- Gamma model
- Constraint Handling Rules
- Mobile ambients
- P-system, membrane computing
- Amorphous computing
- Bigraphical reactive system (2001)

# Example 2: N-to-1 stream communication

$\{ i(X0), o(Y0), \$p[|*Z]) \}, c(A, X, X0) :-$   
 $c(A, Y, Y0), \{ i(X), o(Y), \$p[|*Z]) \}$



★ The number of free links in  $\{ \}$  remain unchanged

# Elements of $\mathcal{LMNtal}$ (1)

---

## 1. Nodes (= atomic processes) with links

- ★ Links are linear, zero-assignment logical variables
  - linear = occurring twice (1-to-1 comm.)
  - logical = **link identity changes after message sending** ( $\leftrightarrow$   $\pi$ -calculus)
  - zero-assignment = **not** instantiated ( $\leftrightarrow$  logic programming)
  - private
  - not directed (cf. chemical bonds)
- ★ Links of a node are ordered

# Elements of $\mathcal{LMNtal}$ (1)

---

- ★ Links are used
  - (a) to represent (private) communication channels
  - (b) to represent data structures (= graphs)
  - (c) to find partners in multiset rewriting
    - ◆  $O(1)$  if linked
    - ◆ can be  $O(n)$  if not linked
  - (d) to represent hyperlinks (using membranes; see next slide)

# Elements of $\mathcal{LMNtal}$ (2)

---

## 2. First-class multisets (using membranes)

- ★ Not many languages feature multisets as *first-class* citizens
- ★ Used for :
  - representing records (feature structures)
  - localization and logical management of computation
    - ◆ cf. ambients, join calculus, Unix processes

# Elements of $\mathcal{LMNtal}$ (3)

---

## 3. Rewrite rules

- ★ Can be put in a membrane to realize
  - local reaction
  - process mobility
- ★ Design issue: proper handling of free links
  - ◆ cf. graph grammars and transformation, logic programming

# Syntax: preliminaries

---

- Two presupposed syntactic categories:
  - ★  $X$  : links (or link variables)
    - In concrete syntax, start with capital letters
  - ★  $p$  : names (including numbers)
    - In concrete syntax, use identifiers different from links
- The name “=” (called a **connector**) is the only reserved symbol in  $\mathcal{LMNtal}$

# Syntax of $\mathcal{LMNtal}$ processes

- $P ::= \mathbf{0}$  (null)
  - |  $p(X_1, \dots, X_m)$  ( $m \geq 0$ ) (atom)
  - |  $P, P$  (molecule)
  - |  $\{P\}$  (cell)
  - |  $T :- T$  (rule)
- Not in Flat LMNtal
- **Link condition:** Each link in  $P$  (except those in reaction rules) occurs **at most twice**.
    - ★ **Free link of  $P$**  = link occurring only once
    - ★  $P$  is **closed** = has no free links



# Syntax of $\mathcal{LMNtal}$ process templates

- $T ::= \mathbf{0}$  (null)
- |  $p(X_1, \dots, X_m)$  ( $m \geq 0$ ) (atom)
- |  $T, T$  (molecule)
- |  $\{T\}$  (cell)
- |  $T :- T$  (rule)
- |  $@p$  (rule context)
- |  $\$p[X_1, \dots, X_m | A]$  ( $m \geq 0$ ) (process context)
- |  $p(*X_1, \dots, *X_m)$  ( $m > 0$ ) (aggregate)
- (residual args)  $A ::= []$  (empty)
- |  $*X$  (bundle)

Not in  
Flat LMNtal

# LHS conditions on the rule $T \text{ :- } T$

---

1. No rules
  - ★ so they won't be examined dynamically.
2. No aggregates
3. No rule contexts or process contexts outside cells
  - ★ they are to match “contexts” within the innermost cell

# Occurrence conditions on $T :- T$

---

1. Links and bundles must occur exactly twice.
2. Links occurring within a process context on the LHS must be distinct.
3. Bundles on the LHS must be distinct.
4. Names of rule contexts and process contexts must occur exactly once on the LHS and not within any other inner rule.
5. Each cell can have at most one rule context and at most one process context in its toplevel.

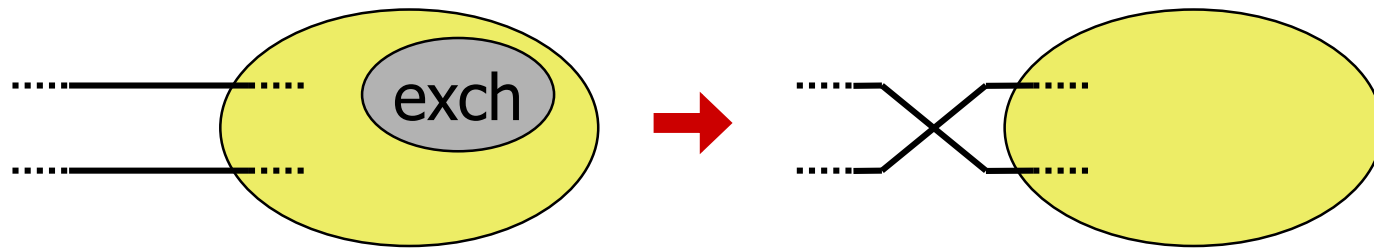
# Consistency conditions on $T:-T$

---

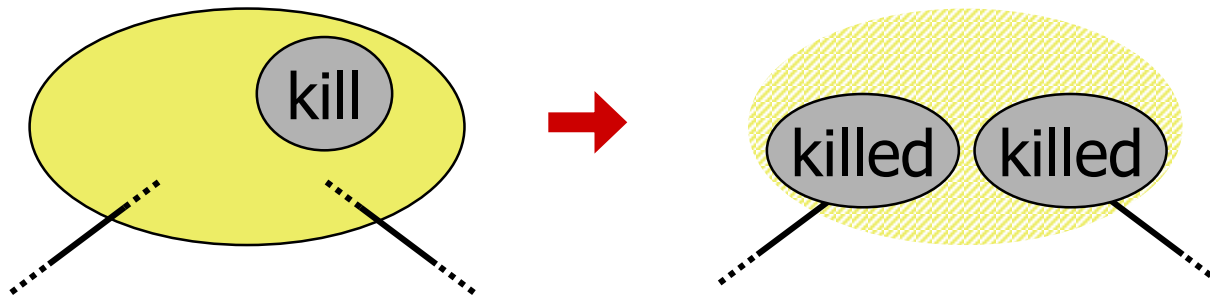
1. All process contexts having the same name must have the same arity and “consistent” residual args.
2. For each aggregate  $p(*X_1, \dots, *X_m)$ , there must be a single process context name  $\$q$  and all the  $*X_i$ ’s occur in process contexts with the name  $\$q$ .

# Process contexts, examples

- $\{\text{exch}, \$a[X,Y|[]]\} :- \{\$a[Y,X|[]]\}$



- $\{\text{kill}, \$a[|*X[]]\} :- \text{killed}(*X)$



# Syntactic sugar

---

$c(A1, X1, X0), c(A2, X2, X1), c(A3, X3, X2), n(X3)$

$\equiv c(A1, c(A2, c(A3, n)), X0)$

$\equiv X0=Y, c(A1, c(A2, c(A3, n)), Y)$

$\equiv X0=c(A1, c(A2, c(A3, n)))$

# Structural congruence

$$(E1) \quad \mathbf{0}, P \equiv P$$

$$(E2) \quad P, Q \equiv Q, P$$

$$(E3) \quad P, (Q, R) \equiv (P, Q), R$$

$$(E4) \quad P \equiv P[Y/X] \quad \text{if } X \text{ is a local link of } P$$

$$(E5) \quad P \equiv P' \Leftrightarrow P, Q \equiv P', Q$$

(Link conditions should hold on both sides of  $\equiv$ )

$$(E6) \quad P \equiv P' \Leftrightarrow \{P\} \equiv \{P'\}$$

$$(E7) \quad X = X \equiv \mathbf{0}$$

$$(E8) \quad X = Y \equiv Y = X$$

$$(E9) \quad X = Y, P \equiv P[Y/X] \quad \text{if } P \text{ is an atom and } X \text{ is a free link of } P$$

# Reduction semantics

$$(R1) \quad \frac{P \rightarrow P'}{P, Q \rightarrow P', Q}$$

$$(R2) \quad \frac{P \rightarrow P'}{\{P\} \rightarrow \{P'\}}$$

$$(R3) \quad \frac{Q \equiv P \quad P \rightarrow P' \quad P' \equiv Q'}{Q \rightarrow Q'}$$

$$(R4) \quad \{X=Y, P\} \rightarrow X=Y, \{P\}$$

if  $X$  and  $Y$  don't occur in  $P$ , and  $X$  and  $Y$  are distinct

$$(R5) \quad X=Y, \{P\} \rightarrow \{X=Y, P\}$$

if  $X$  occurs in  $P$ 's non-rule part

$$(R6) \quad T\theta, (T:-U) \rightarrow U\theta, (T:-U)$$

if  $T\theta$ 's non-rule part doesn't contain =

$\theta$  is to instantiate process & rule variables. Links are matched using  $\alpha$ -conversion.



# Reduction semantics

---

- Can  $p(A,A)$  be reduced using  $p(X,Y) \text{ :- } q(Y,X)$  ?

- ★ The rule can't be  $\alpha$ -converted to the form

$p(A,A) \text{ :- } \dots$

- ★ However, because  $p(A,A)$  is equivalent to  $p(A,B), A=B$  (B a fresh link), it can be reduced as:

$p(A,A)$

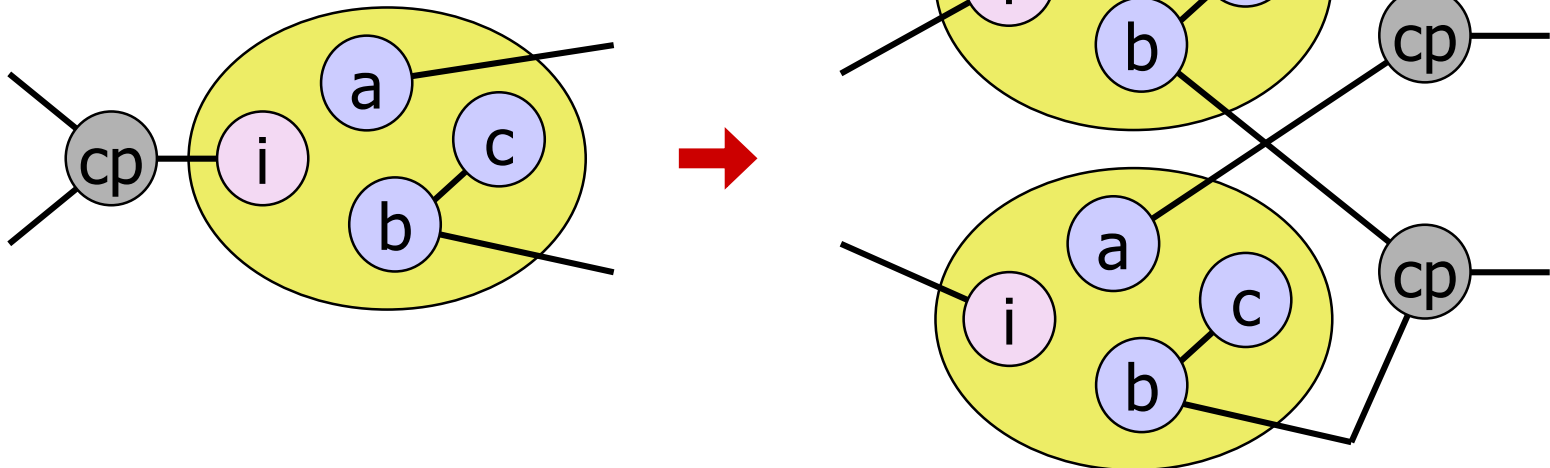
$\equiv p(A,B), A=B$

$\rightarrow q(B,A), A=B$

$\equiv q(A,A)$

# Process contexts, examples

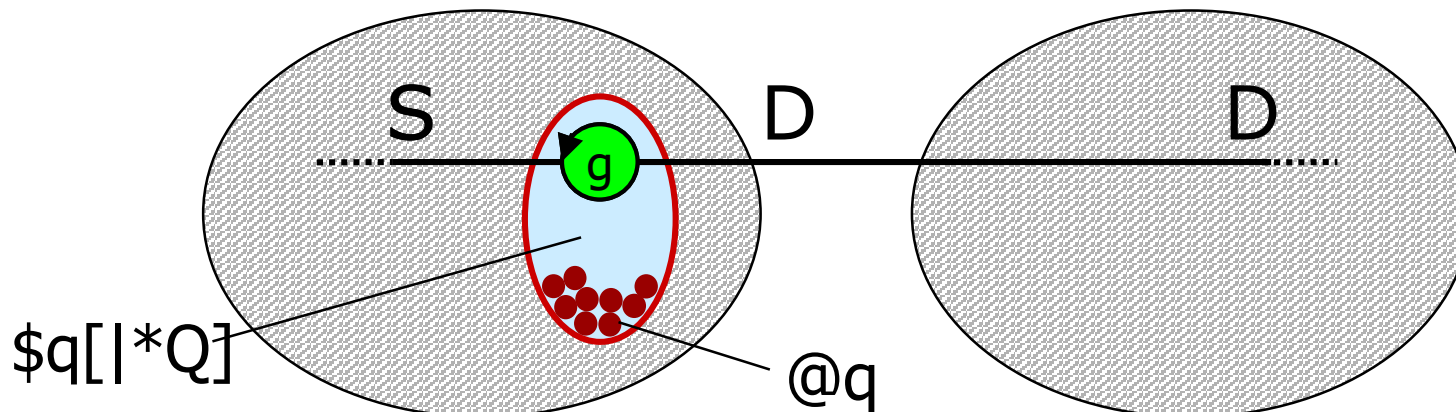
- $\text{cp}(S, S1, S2), \{i(S), \$p[|*P]\} :-$   
 $\{i(S1), \$p[|*P1]\},$   
 $\{i(S2), \$p[|*P2]\},$   
 $\text{cp}(*P, *P1, *P2)$



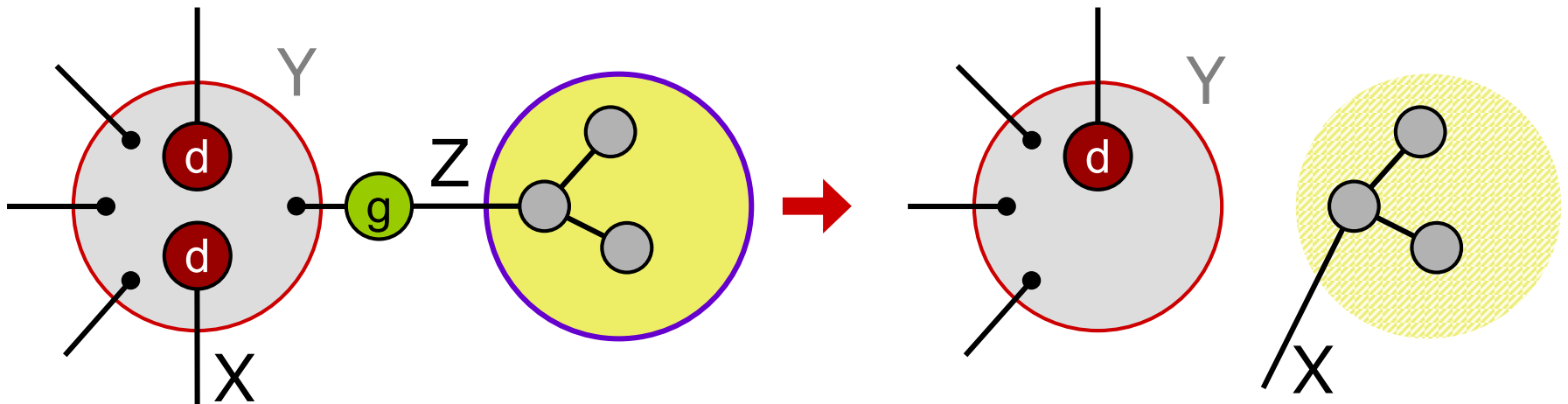
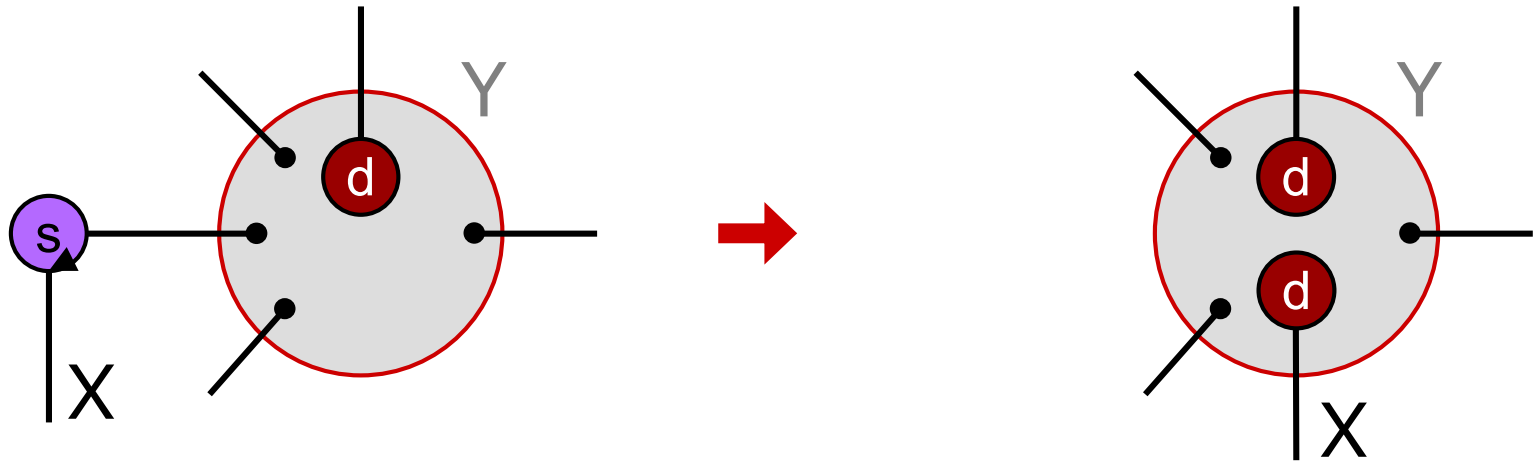
# Example 3: process migration

$\{\$p[S|*P]), \{ @q, \$q[|*Q], go(S,D) \}, \{ \$r[D|*R] \} :-$   
 $\{ \$p[S|*P] \}, \{ \{ @q, \$q[|*Q] \}, arrived(S,D) \}, \$r[D|*R] \}$

- ★ Placed outside the membranes
- ★ Innermost  $\{ \}$  is to specify what should migrate



# Example 4: asynchronous $\pi$ -calculus



# Example 5: circular data structures

## ■ Bidirectional circular buffer:

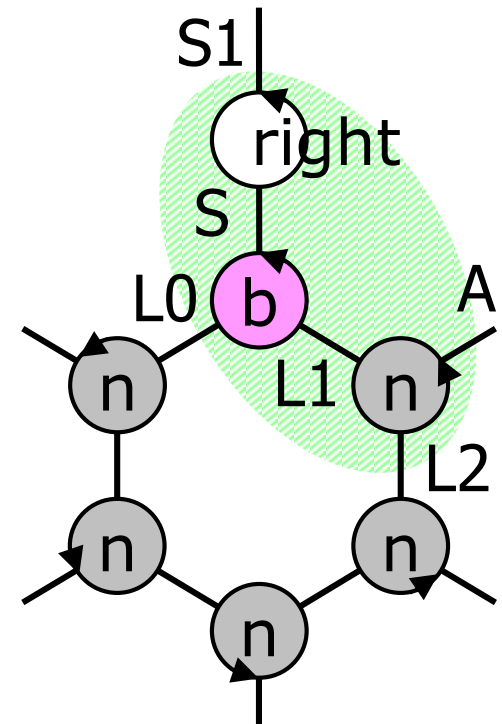
$$b(S, L_n, L_0), n(A_1, L_0, L_1), \dots, n(A_n, L_{n-1}, L_n)$$

★  $S$  acts as an interface link

```

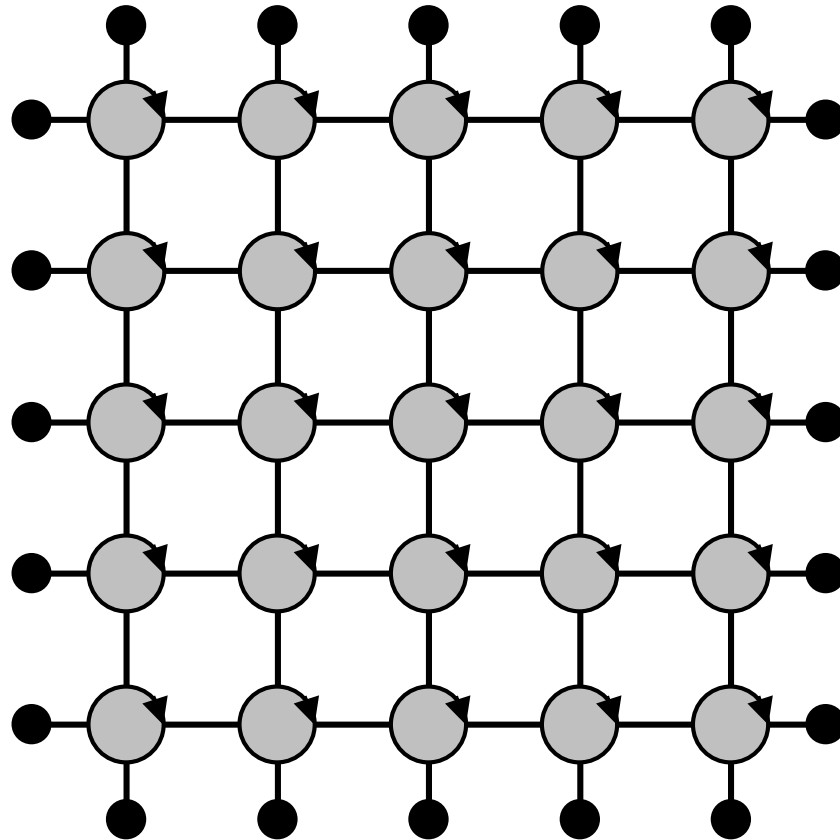
left(S1, S), n(A, L0, L1), b(S, L1, L2) :-
    b(S1, L0, L1), n(A, L1, L2)
right(S1, S), b(S, L0, L1), n(A, L1, L2) :-
    n(A, L0, L1), b(S1, L1, L2)
put(A, S1, S), b(S, L0, L2) :-
    n(A, L0, L1), b(S1, L1, L2)
  
```

★ cf. Shape Types



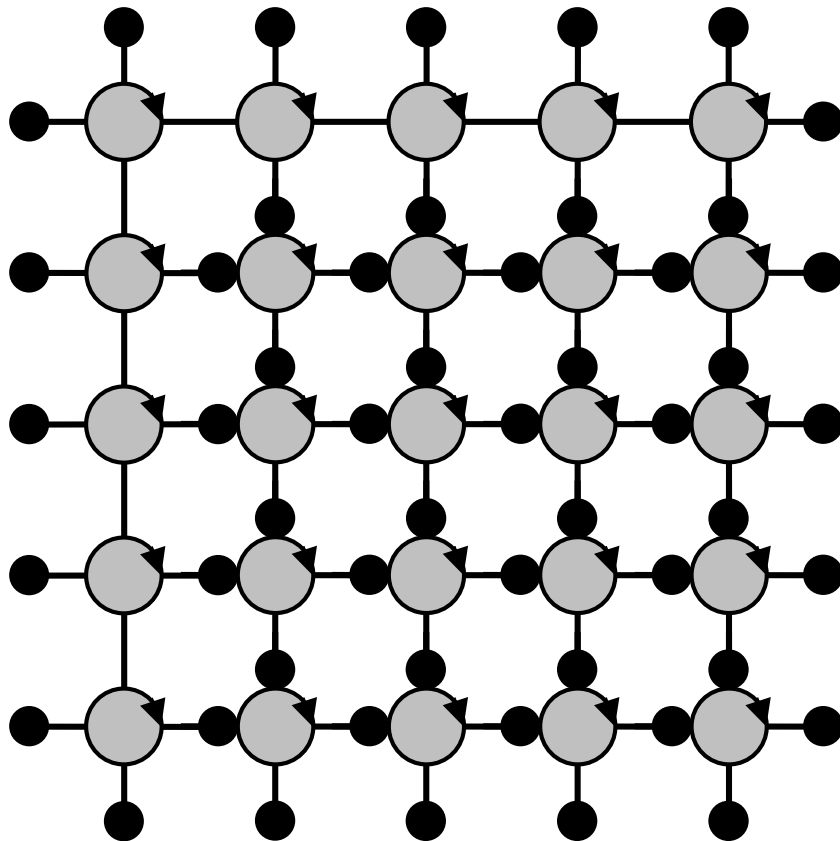
# Example 6: self-organizing 2-D grid

---

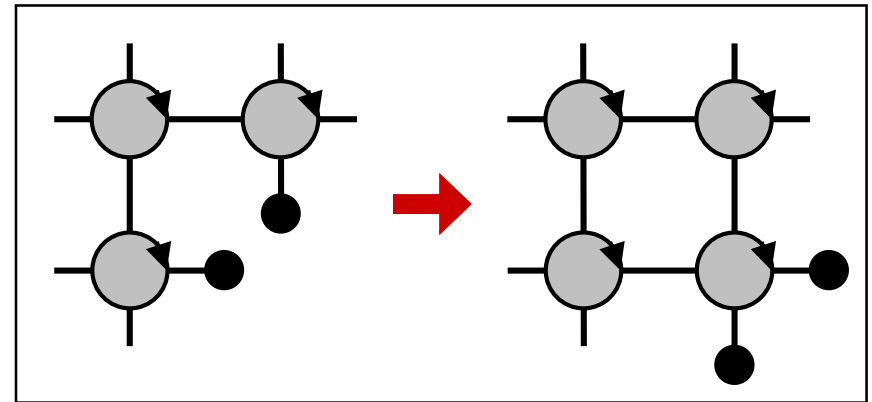


Final State

# Example 6: self-organizing 2-D grid

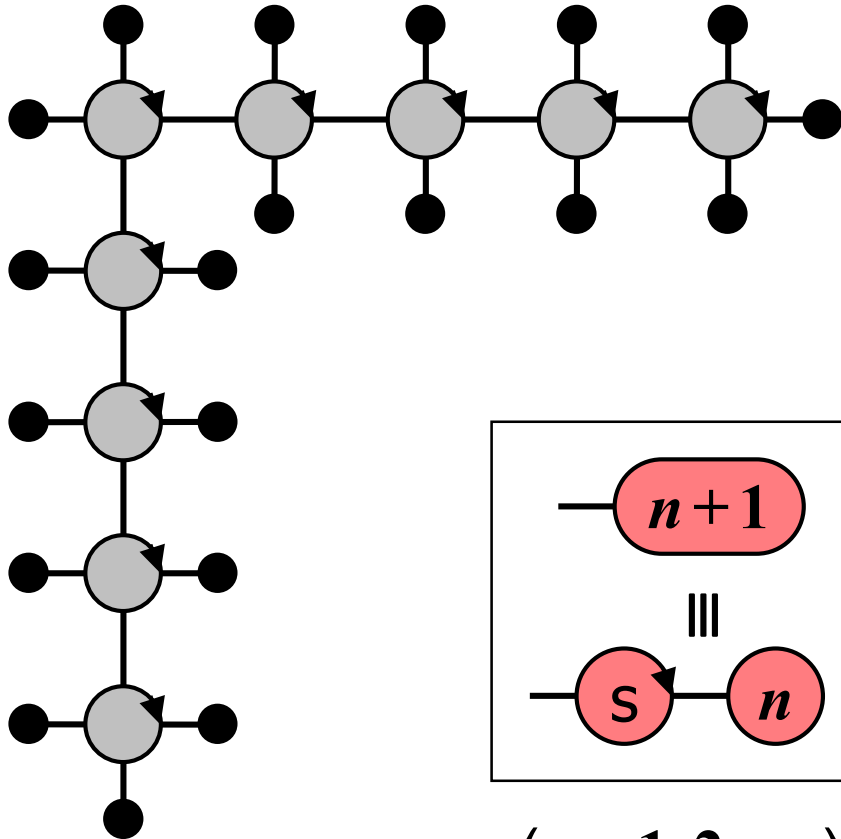


Initial State

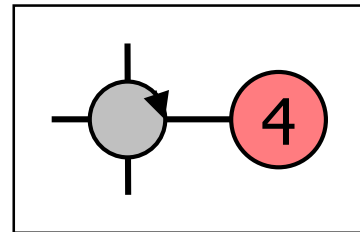


Reaction rule

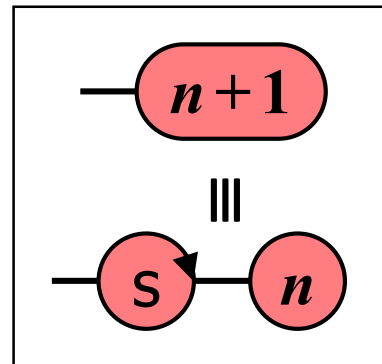
# Example 6: self-organizing 2-D grid



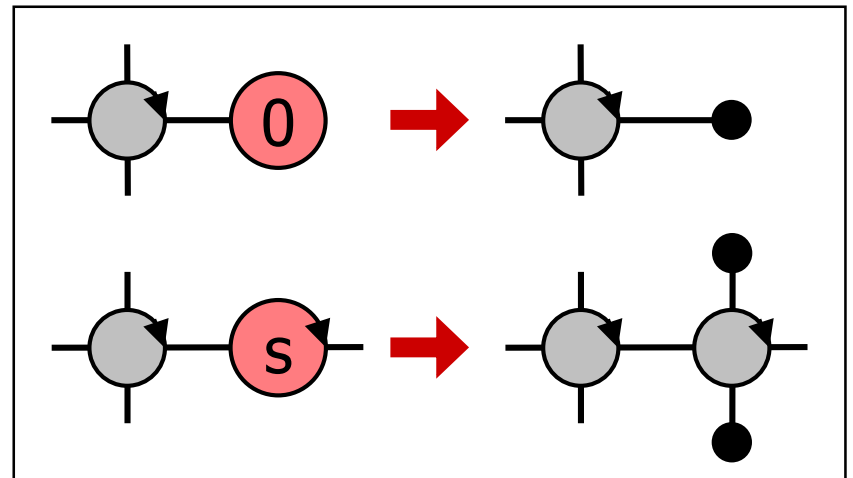
Building the uppermost row



Initial State



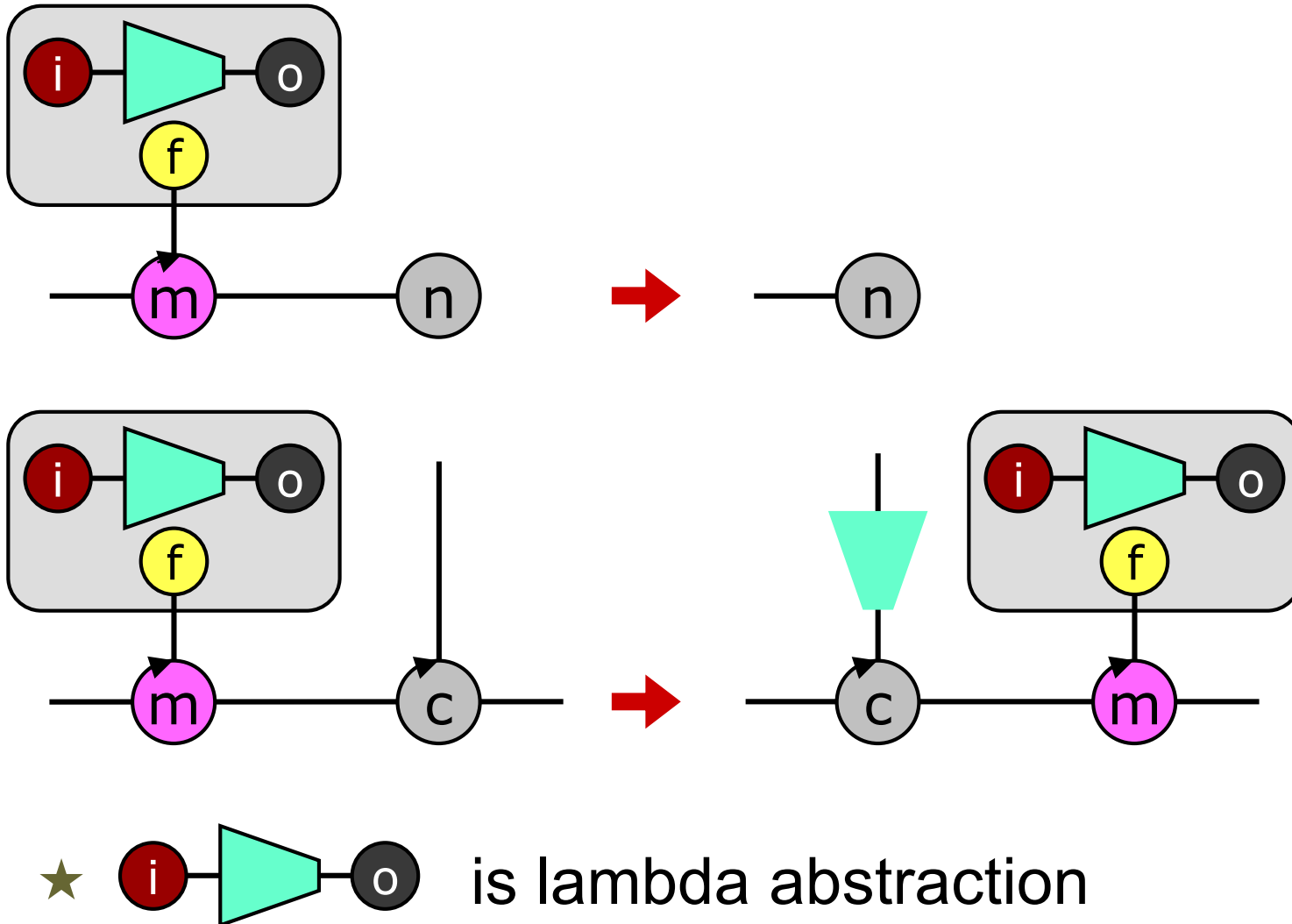
$(n = 1, 2, \dots)$



Reaction Rules



# Example 7: map function



# Extension: termination detection

---

- Syntax, expanded

$P ::= \dots \mid \{P\}/$  (quiet cell)

$T ::= \dots \mid \{T\}/$  (quiet cell)

- Structural congruence, expanded

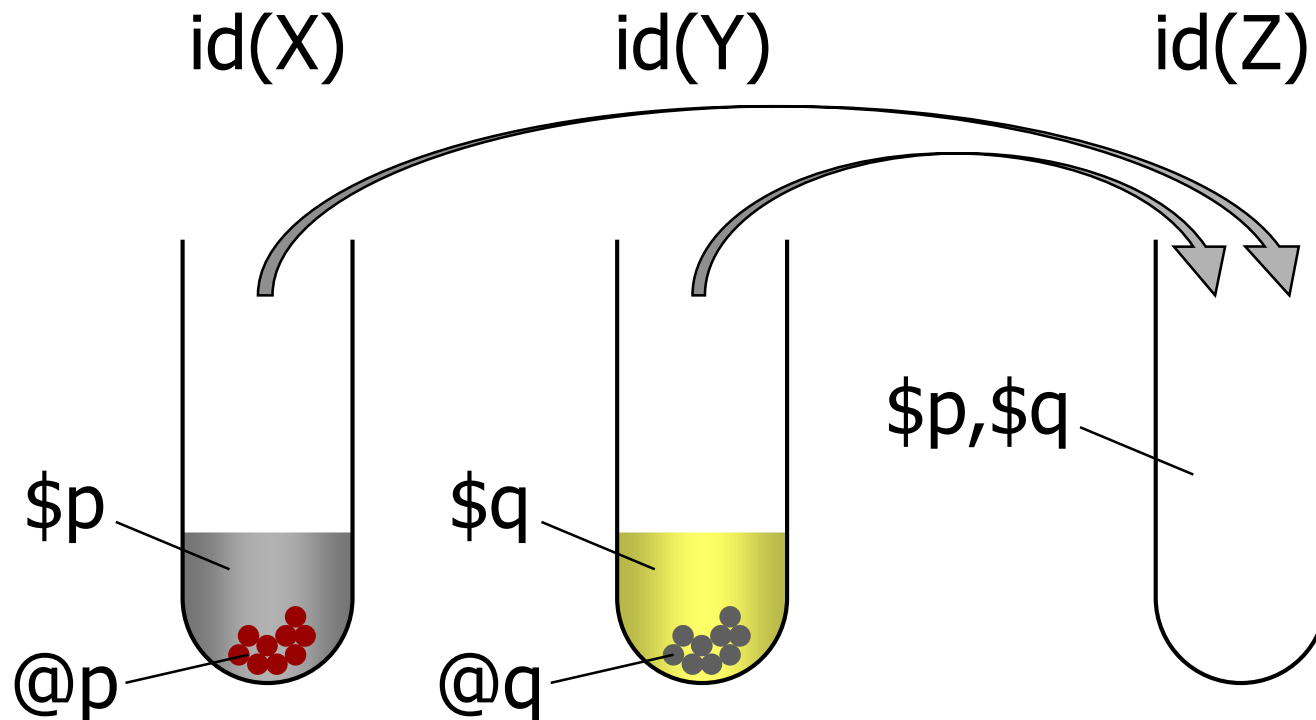
$\{P\} \equiv \{P\}/$  if  $P \nrightarrow$

- ★ An irreducible cell can show the flag “/”

- ★ Irreducibility of a cell can only be checked from outside the cell

# Example 8: termination detection

```
{id(X), $p, @p} /, {id(Y), $q, @q} /, {id(Z)} :-  
    p(X, Y, Z) | % test tube manager  
    {id(X), @p}, {id(Y), @q}, {id(Z), $p, $q}
```



# Conclusions

---

- The “four elements” of  $\mathcal{LMNtal}$  are:
  - ★ (logical) links,
  - ★ multisets/membranes,
  - ★ (nested) nodes, and
  - ★ transformation.
- Inspired by communication using logical variables, we have designed a simple language model for the unified treatment of processes, messages and data.

# References

---

- Andries, M. *et al.*, Graph Transformation for Specification and Programming. *Sci. Comput. Program.*, Vol.34, No.1 (1999), pp.1-54.
- Banâtre, J.-P. and Le Métayer, D., Programming by Multiset Transformation. *Commun. ACM*, Vol.35, No.1 (1993), pp.98-111.
- Drewes, F., Hoffmann, B. and Plump, D., Hierarchical Graph Transformation. *J. Comput. Syst. Sci.*, Vol.64, No.2 (2002), pp.249-283.
- Engels, G. and Schürr, A., Encapsulated Hierarchical Graphs, Graph Types, and Meta Types. *Electronic Notes in Theor. Comput. Sci.*, Vol.1 (1995), pp.75-84.
- Fradet, P. and Le Métayer, D., Shape Types. In *Proc. POPL'97*, ACM Press, 1997, pp.27-39.

# References

---

- Frühwirth, T., Theory and Practice of Constraint Handling Rules. *J. Logic Programming*, Vol.37, No.1-3 (1998), pp.95-138.
- Lafont, Y., Interaction Nets. In *Proc. POPL'90*, pp. 95-108.
- R. Milner, Bigraphical reactive systems: basic theory. <http://www.cl.cam.ac.uk/~rm135/bigraphs.pdf>, 2001.
- Paun, Gh., Computing with Membranes. *J. Comput. Syst. Sci.*, Vol.61, No.1 (2000), pp.108-143.
- Saraswat, V.A., Kahn, K. and Levy, J., Janus: A Step Towards Distributed Constraint Programming. In *Proc. 1990 North American Conf. on Logic Programming*, MIT Press, 1990, pp.431-446.
- Ueda, K., Resource-Passing Concurrent Programming. In *Proc. TACS 2001*, LNCS 2215, Springer, 2001, pp.95-126.

# Ongoing and future work (1)

---

## ■ Type systems supporting multisets

★ Purposes ... (let's skip; should be clear to us !)

★ Properties we want to guarantee:

- Process structures ... "do these rules form lists, trees or rings?"
- Neighboring atoms ... "should I look for my partners myself, or do they look for me?"
- Link directionality ... "can this link be implemented as a one-way pointer?"
- Cell links ... "what and how many free links emanate from this membrane?"

# Ongoing and future work (2)

---

## ■ Implementing *LMNtal*

- ★ process representation
- ★ rule compilation
- ★ parallel and distributed processing
- ★ Java implementation of a LMNtal subset by R. Sakuma, T. Nagata and S. Yajima (2003)
- ★ Ruby implementation by N. Kato (2003)
- ★ fuller Java implementation ongoing (2004)

## ■ Program verification

- ★ study of “human-oriented” proofs



# Ongoing and future work (3)

---

- Relating to and embedding other models featuring multisets and joins
  - ★  $\pi$ -calculus, ambient calculus, CHR
- Language design
  - ★ scoping and ordering constructs for names and rules
  - ★ exception handling
  - ★ resource-consciousness
  - ★ location-consciousness
  - ★ nondeterministic  $\mathcal{LMNtal}$

# Ongoing and future work (4)

---

## ■ Applications

- ★ Graph algorithms
- ★ Multi-agent systems (e.g., MOO)
- ★ Web services
- ★ Wide-area computation
- ★ Programming based on self-assembly
- ★ Amorphous and molecular computing
- ★ . . .