Higher-order Chemical Programming Style

Jean-Pierre Banâtre[1], Pascal Fradet[2] and Yann Radenac[1]

[1] IRISA, Université de Rennes 1, France

[2] INRIA Rhône-Alpes, Grenoble, France

Plan

1. Chemical programming
2. Higher-order chemical programming: the γ-calculus
3. Chemical autonomic systems

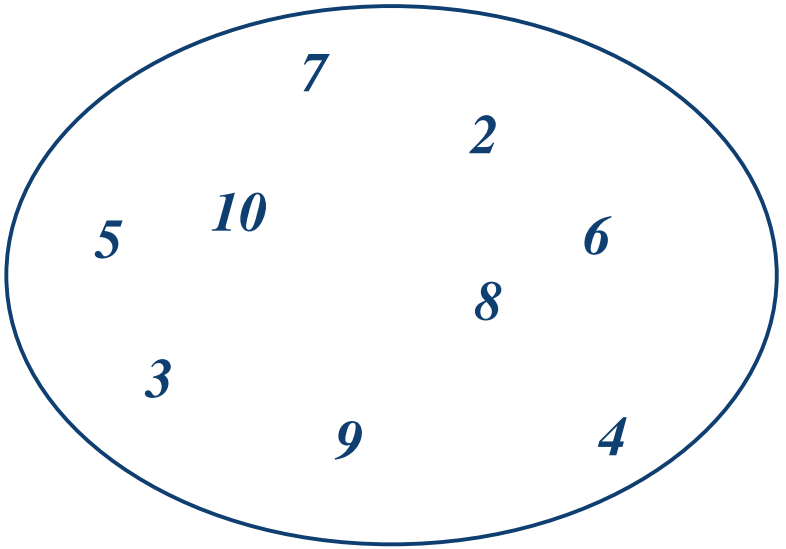
Chemical programming

- Gamma (Banâtre and Le Métayer, 1986)
- Chemical metaphor:
  - Computation is viewed as the global evolution of a collection of molecules interacting freely in a chemical solution
- Formally:
  - Computation is made of multiset rewritings

Gamma: an example

- Computing the prime numbers:

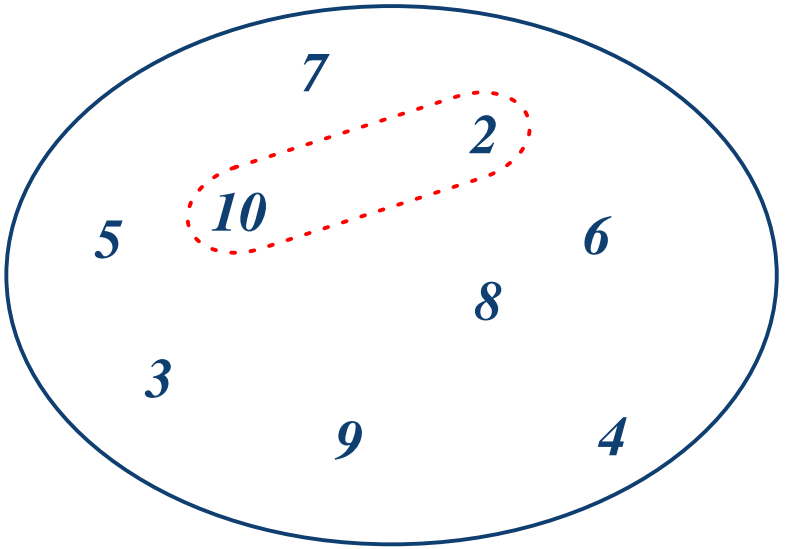
$$\text{prime} = \mathbf{replace}\ x, y \ \mathbf{by}\ x \ \mathbf{if}\ x\ div\ y$$



Gamma: an example

- Computing the prime numbers:

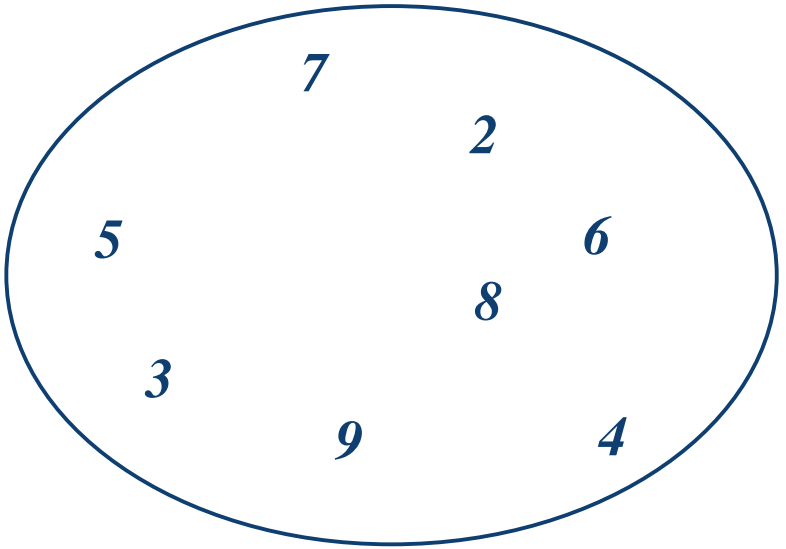
$$\text{prime} = \mathbf{replace}\ x, y \ \mathbf{by}\ x \ \mathbf{if}\ x\ div\ y$$



Gamma: an example

- Computing the prime numbers:

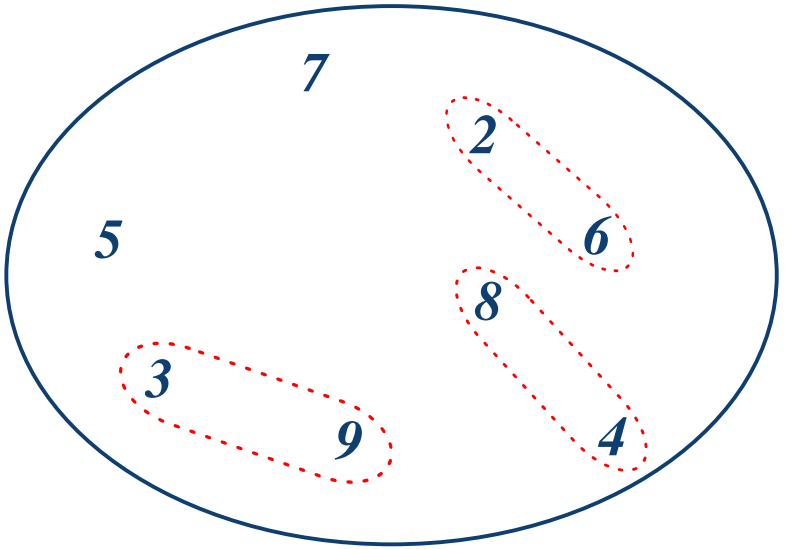
$$\text{prime} = \mathbf{replace}\ x, y \ \mathbf{by}\ x \ \mathbf{if}\ x\ div\ y$$



Gamma: an example

- Computing the prime numbers:

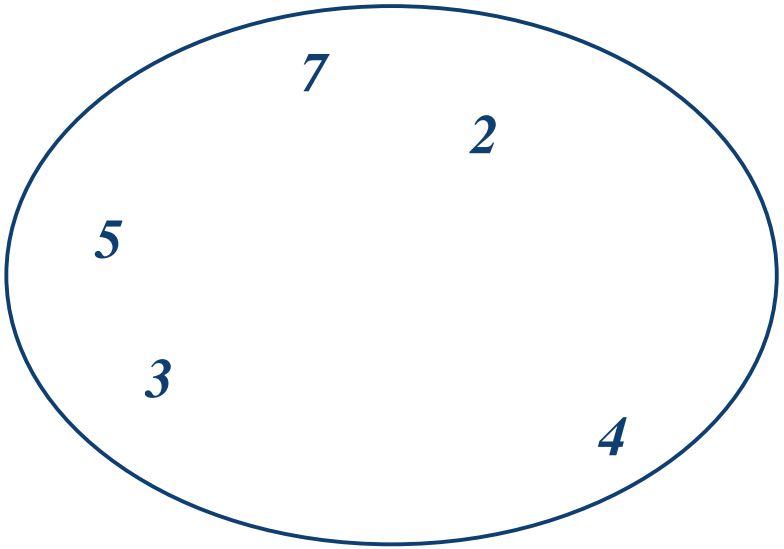
$$\text{prime} = \mathbf{replace}\ x, y \ \mathbf{by}\ x \ \mathbf{if}\ x\ div\ y$$



Gamma: an example

- Computing the prime numbers:

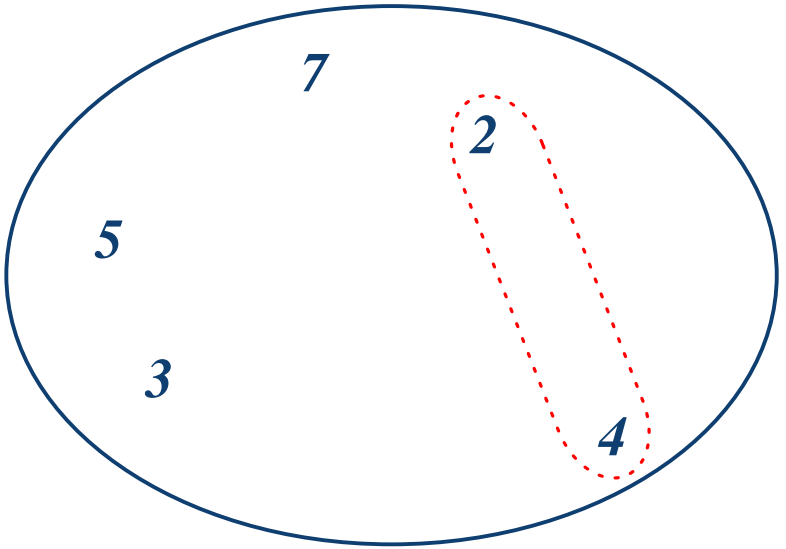
$$\text{prime} = \mathbf{replace}\ x, y \ \mathbf{by}\ x \ \mathbf{if}\ x\ div\ y$$



Gamma: an example

- Computing the prime numbers:

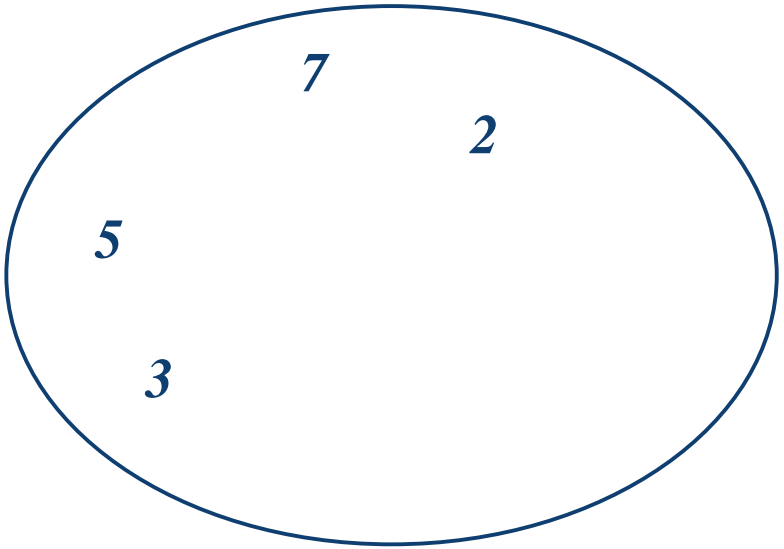
$$\text{prime} = \mathbf{replace}\ x, y \ \mathbf{by}\ x \ \mathbf{if}\ x\ div\ y$$



Gamma: an example

- Computing the prime numbers:

$$\text{prime} = \mathbf{replace}\ x, y \ \mathbf{by}\ x \ \mathbf{if}\ x\ div\ y$$



Properties of chemical programming

- Simple, intuitive, parallel and non deterministic model
- Bridging the gap between specifications and programs:
  - beyond specification: we express how to compute
  - needs further refinement: non optimal complexity
- A vision of programming:
  - a set of interacting molecules which evolves “chaotically” towards a result

Higher-order chemical calculus: the γ-calculus

- A higher-order extension
  - expressing mobility, adaptivity, ...
- A minimal and formal basis for the family of chemical languages
- A rewrite system where terms, called molecules, have the following syntax:

$$
\begin{array}{rcll}
M & ::= & a \mid b \mid \ldots & \textit{; variables} \\
 & \mid & \gamma\langle x\rangle.M & \textit{; abstractions} \\
 & \mid & M_1, M_2 & \textit{; multiset (AC)} \\
 & \mid & \langle M\rangle & \textit{; solution}
\end{array}
$$

Higher-order chemical calculus: the γ-calculus

- The chemical reaction (γ-reduction):

$$\langle N \rangle, \gamma\langle x \rangle.M \longrightarrow M[x := N] \qquad \text{if } (N \text{ is inert})$$

- Principles of chemical models:
  - Brownian motion (AC operator)
  - Local reaction (rule-based rewritings)
  - Minimum of control (access only to the content of inert solutions)

Higher-order chemical programming

- Extension of the minimal γ-calculus with reaction conditions, atomicity and a richer pattern matching:

$$
\begin{array}{rcll}
M & ::= & a \mid b \mid \ldots & \textit{; variables} \\
 & \mid & 0 \mid 1 \mid \ldots & \textit{; constants} \\
 & \mid & \gamma(P)\lfloor C \rfloor.M & \textit{; one-shot abstraction} \\
 & \mid & \mathbf{replace}\ P\ \mathbf{by}\ M\ \mathbf{if}\ C & \textit{; n-shot abstraction (catalyst)} \\
 & \mid & M_1, M_2 & \textit{; multiset (AC)} \\
 & \mid & \langle M \rangle & \textit{; solution}
\end{array}
$$

- The chemical reaction (n-shot γ-reduction):

$$
(\mathbf{replace}\ P\ \mathbf{by}\ M\ \mathbf{if}\ C), N \longrightarrow (\mathbf{replace}\ P\ \mathbf{by}\ M\ \mathbf{if}\ C), M[P := N]
$$

if (N matches P) and ($C[P := N]$)

Higher-order chemical programming

- Largest prime number lower than 10:

$$\langle\langle \mathsf{prime}, 2, 3, \ldots, 10\rangle, \mathsf{maxprime}\rangle$$
$$\downarrow^*$$
$$\langle\langle \mathsf{prime}, 2, 3, 5, 7\rangle, \mathsf{maxprime}\rangle$$
$$\downarrow$$
$$\langle 2, 3, 5, 7, \mathsf{max}\rangle$$
$$\downarrow^*$$
$$\langle \mathsf{max}, 7\rangle$$

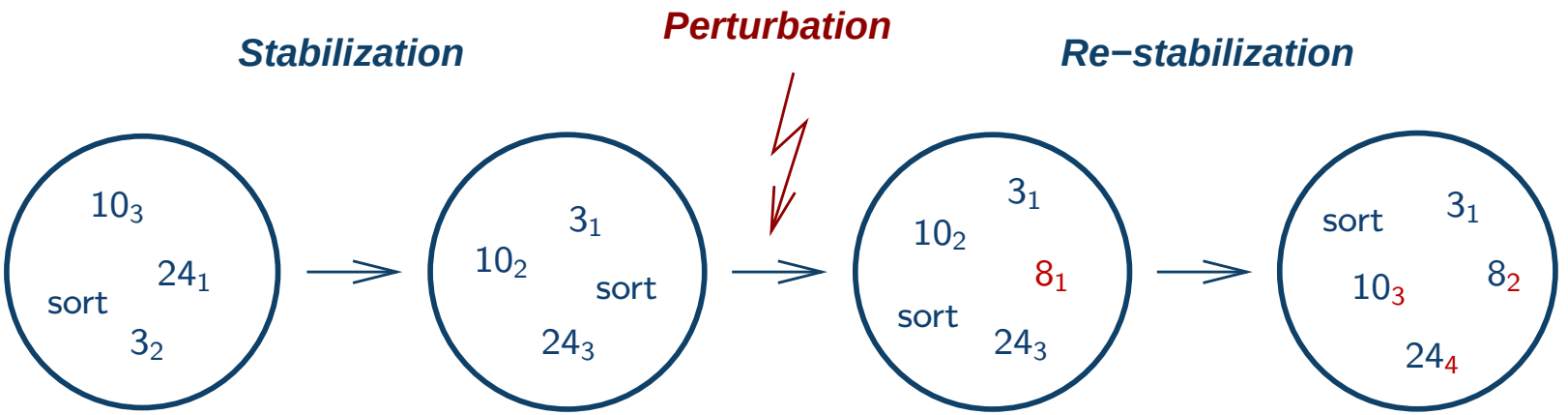
where

$$\mathsf{prime} = \mathbf{replace}\, x, y\, \mathbf{by}\, x\, \mathbf{if}\, x\ div\ y$$
$$\mathsf{maxprime} = \gamma\langle \mathsf{prime}, \omega\rangle.\omega, \mathsf{max}$$
$$\mathsf{max} = \mathbf{replace}\, x, y\, \mathbf{by}\, x\, \mathbf{if}\, x > y$$

Autonomic systems

- Autonomic systems manage themselves
- Different non functional properties:
  - self-organization
  - self-healing
  - self-optimization
  - self-protection
  - self-configuration
- Easily expressed as (higher-order) chemical reaction rules

Self-organization

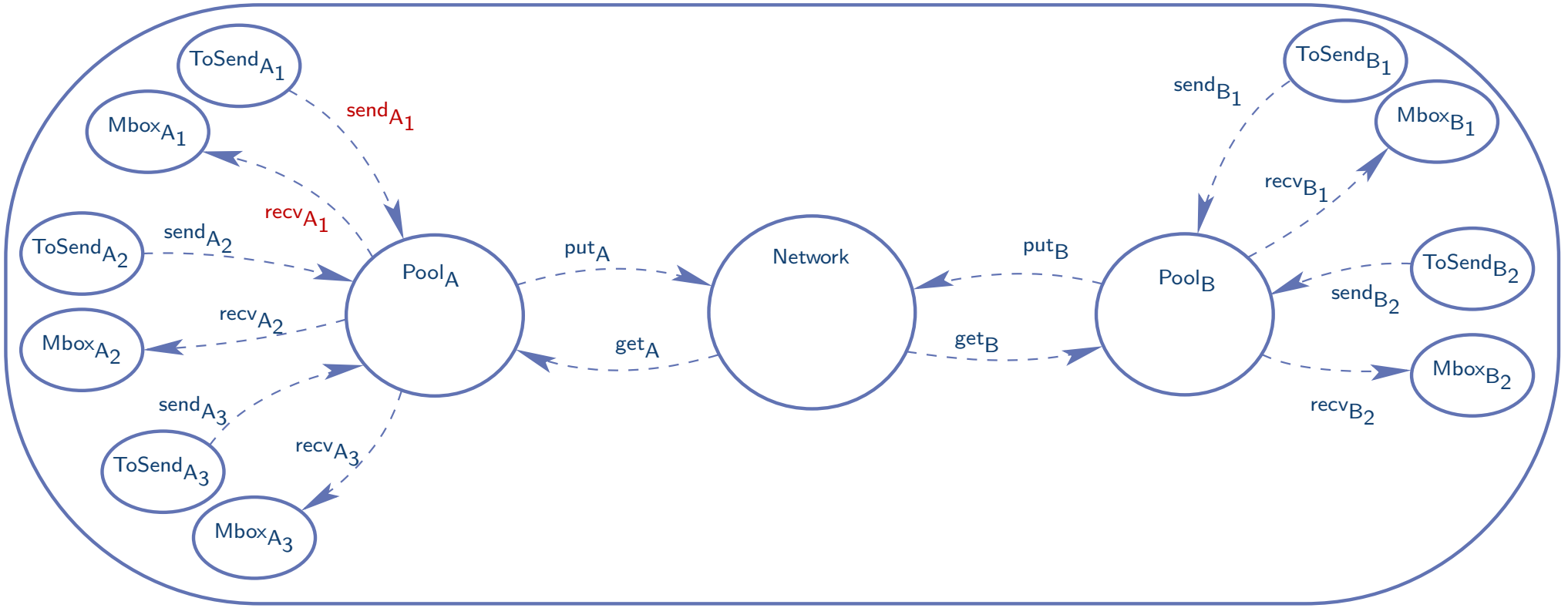


$\Rightarrow$ Catalysts maintain an invariant

An autonomic mail system

- Architecture:
  - mail servers manage domains
  - clients are connected to domains
  - clients send messages to their domain where the server forwards them to the network if needed
  - servers get messages addressed to their domain
  - mail servers may crash

An autonomic mail system

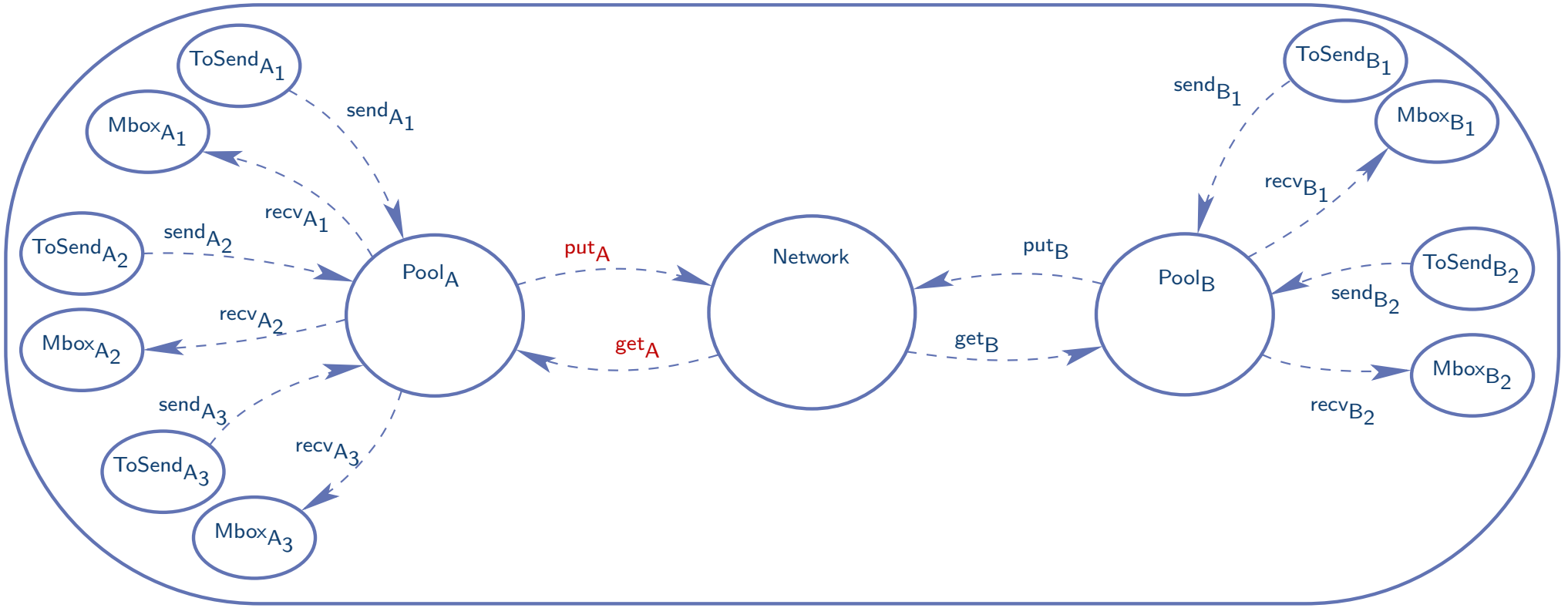


- *Self-organization:* clients/domains

$$\text{send}_{d_i} = \textbf{replace } \text{ToSend}_{d_i} = \langle msg, \omega_t \rangle,\ \text{Pool}_d = \langle \omega_p \rangle$$
$$\textbf{by } \text{ToSend}_{d_i} = \langle \omega_t \rangle,\ \text{Pool}_d = \langle msg, \omega_p \rangle$$

$$\text{recv}_{d_i} = \textbf{replace } \text{Pool}_d = \langle msg, \omega_p \rangle, \text{Mbox}_{d_i} = \langle \omega_b \rangle$$
$$\textbf{by } \text{Pool}_d = \langle \omega_p \rangle, \text{Mbox}_{d_i} = \langle msg, \omega_b \rangle$$
$$\textbf{if } recipient(msg) = i$$

An autonomic mail system

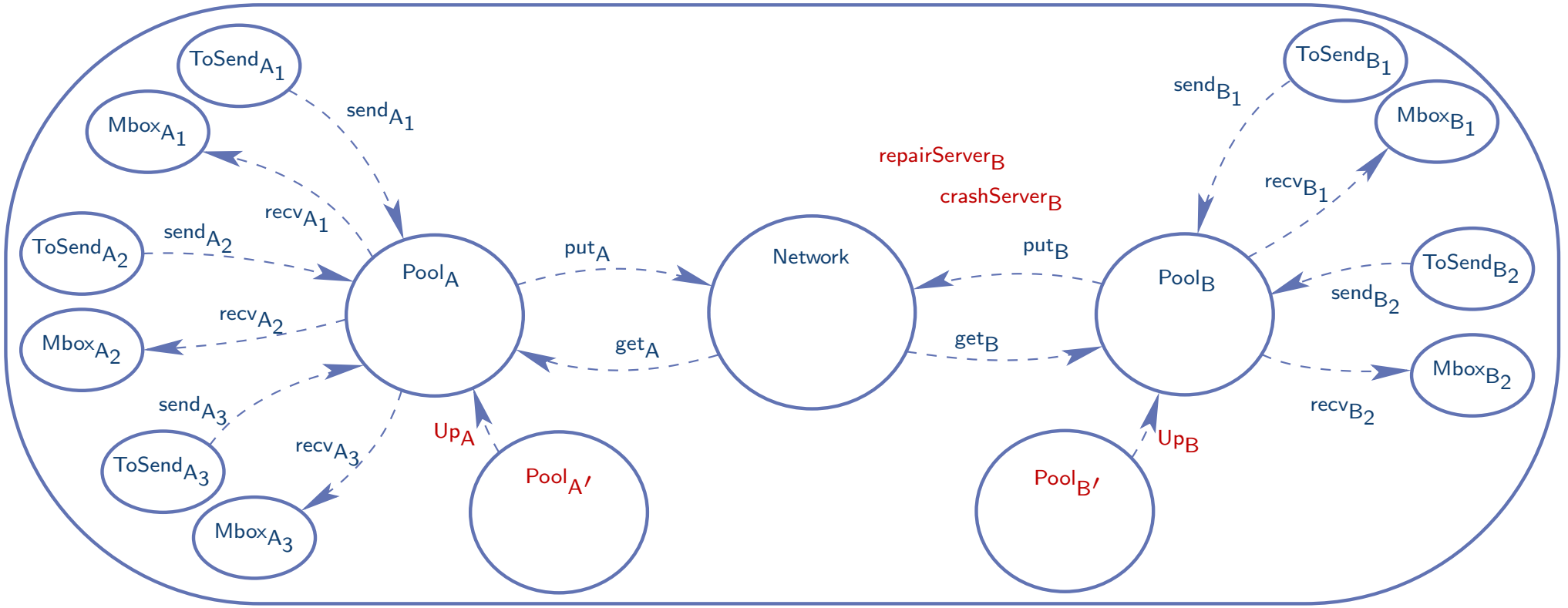


- *Self-organization:* domains/network

$$\text{put}_d = \textbf{replace } \text{Pool}_d = \langle msg, \omega_p \rangle, \text{Network} = \langle \omega_n \rangle$$
$$\textbf{by } \text{Pool}_d = \langle \omega_p \rangle, \text{Network} = \langle msg, \omega_n \rangle$$
$$\textbf{if } recipientDomain(msg) \neq d$$

$$\text{get}_d = \textbf{replace } \text{Network} = \langle msg, \omega_n \rangle, \text{Pool}_d = \langle \omega_p \rangle$$
$$\textbf{by } \text{Network} = \langle \omega_n \rangle, \text{Pool}_d = \langle msg, \omega_p \rangle$$
$$\textbf{if } recipientDomain(msg) = d$$

An autonomic mail system

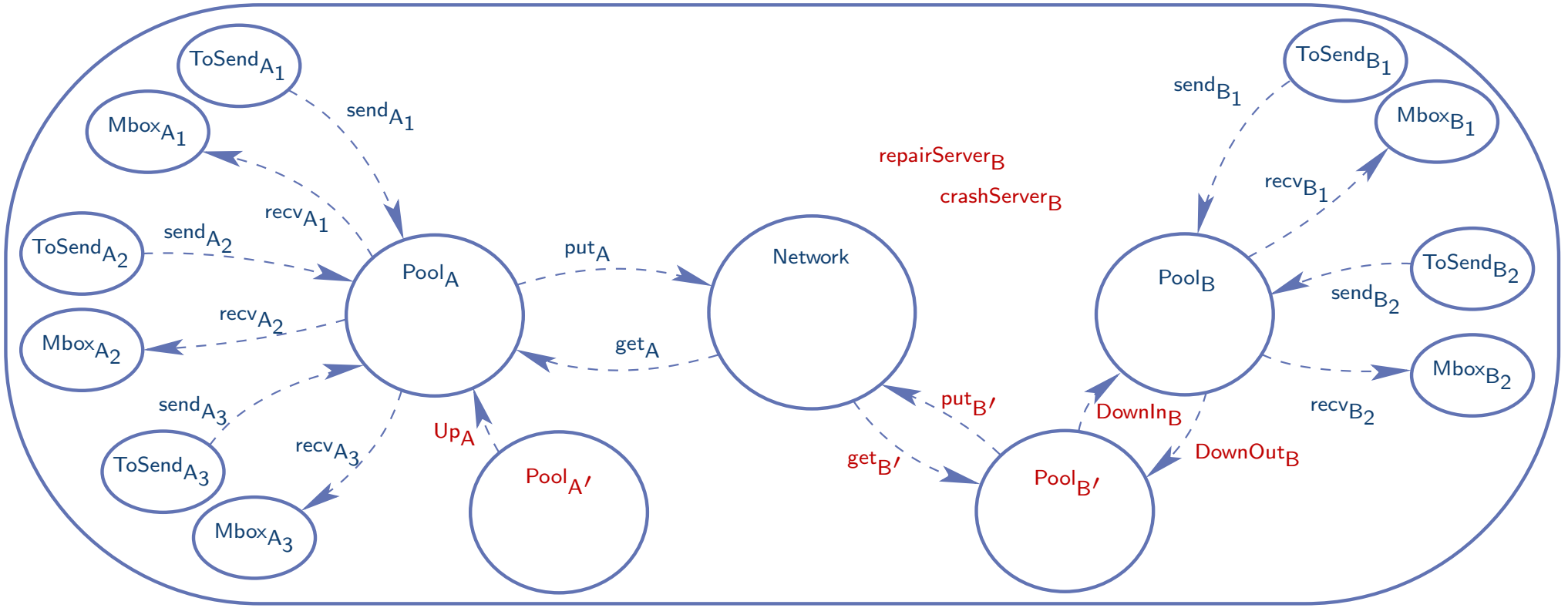


- *Self-healing:* emergency server

$$\mathrm{Up}_d = \mathbf{replace}\mathrm{Pool}_{d'} = \langle msg, \omega_p \rangle, \mathrm{Pool}_d = \langle \omega_n \rangle$$
$$\mathbf{by}\mathrm{Pool}_{d'} = \langle \omega_p \rangle, \mathrm{Pool}_d = \langle msg, \omega_n \rangle$$

$$\mathrm{crashServer}_d = \mathbf{replace}\mathrm{put}_d,\ \mathrm{get}_d,\ \mathrm{Up}_d$$
$$\mathbf{by}\mathrm{put}_{d'},\ \mathrm{get}_{d'},\ \mathrm{DownIn}_d,\ \mathrm{DownOut}_d$$
$$\mathbf{if}\, failure(d)$$

An autonomic mail system

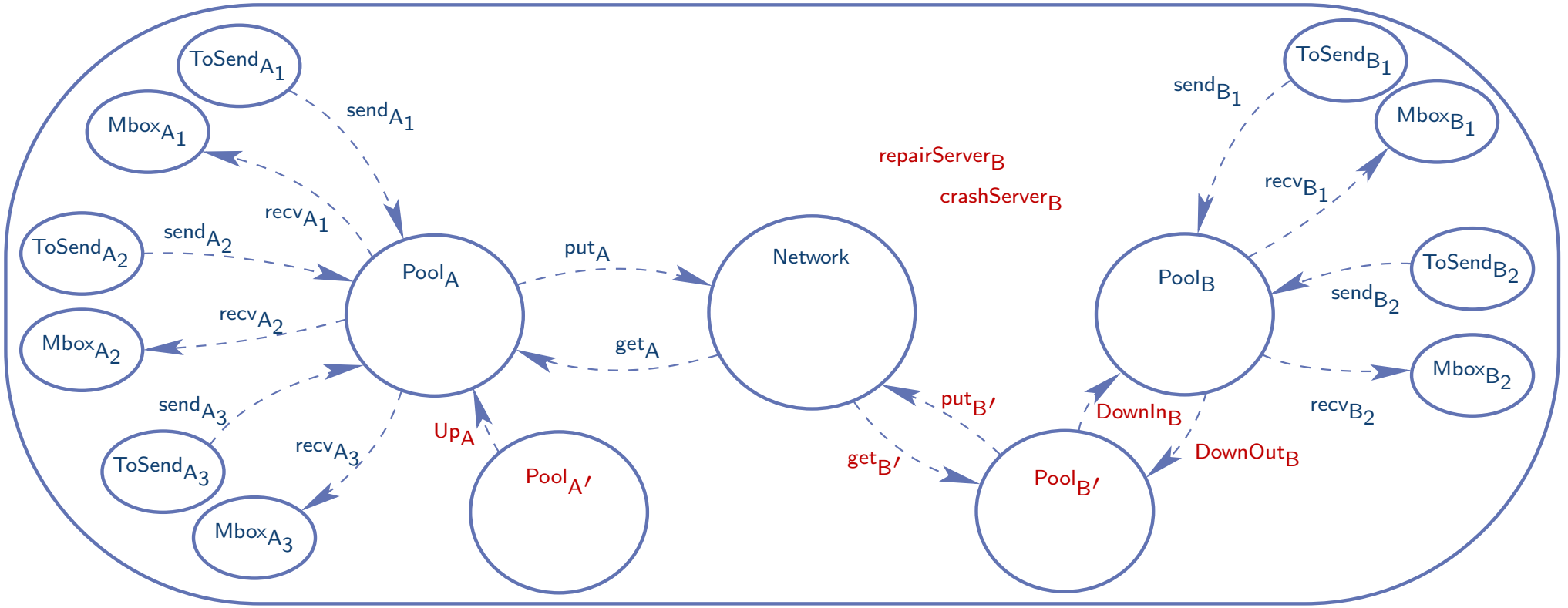


- *Self-healing:* emergency server

$$\mathrm{Up}_d = \mathbf{replace}\mathrm{Pool}_{d'} = \langle msg, \omega_p \rangle, \mathrm{Pool}_d = \langle \omega_n \rangle$$
$$\mathbf{by}\mathrm{Pool}_{d'} = \langle \omega_p \rangle, \mathrm{Pool}_d = \langle msg, \omega_n \rangle$$

$$\mathrm{crashServer}_d = \mathbf{replace}\mathrm{put}_d,\ \mathrm{get}_d,\ \mathrm{Up}_d$$
$$\mathbf{by}\mathrm{put}_{d'},\ \mathrm{get}_{d'},\ \mathrm{DownIn}_d,\ \mathrm{DownOut}_d$$
$$\mathbf{if}\, failure(d)$$

An autonomic mail system

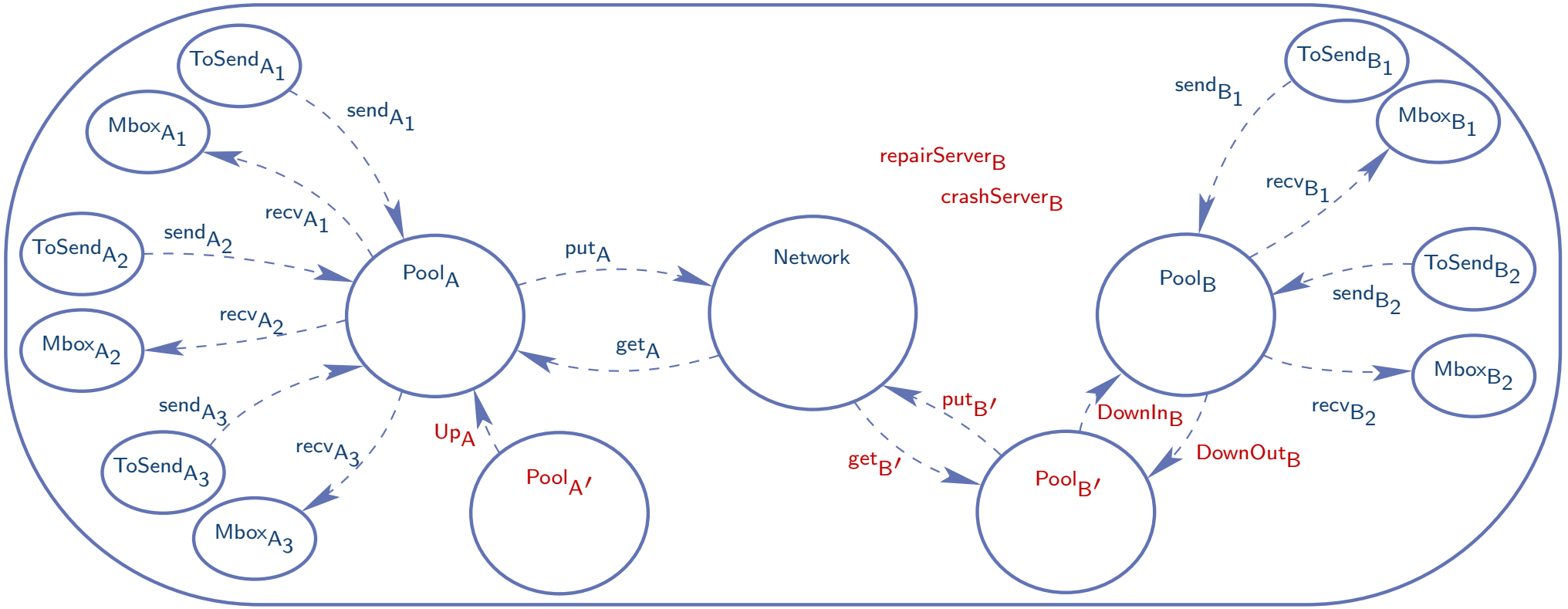


- *Self-healing:* emergency server

$$\mathrm{DownOut}_d = \mathbf{replace}\,\mathsf{Pool}_d = \langle msg, \omega_p \rangle, \mathsf{Pool}_{d'} = \langle \omega_n \rangle$$
$$\mathbf{by}\,\mathsf{Pool}_d = \langle \omega_p \rangle, \mathsf{Pool}_{d'} = \langle msg, \omega_n \rangle$$
$$\mathbf{if}\, domain(msg) \neq d$$

$$\mathrm{DownIn}_d = \mathbf{replace}\,\mathsf{Pool}_d = \langle \omega_p \rangle, \mathsf{Pool}_{d'} = \langle msg, \omega_n \rangle$$
$$\mathbf{by}\,\mathsf{Pool}_d = \langle msg, \omega_p \rangle, \mathsf{Pool}_{d'} = \langle \omega_n \rangle$$
$$\mathbf{if}\, domain(msg) = d$$

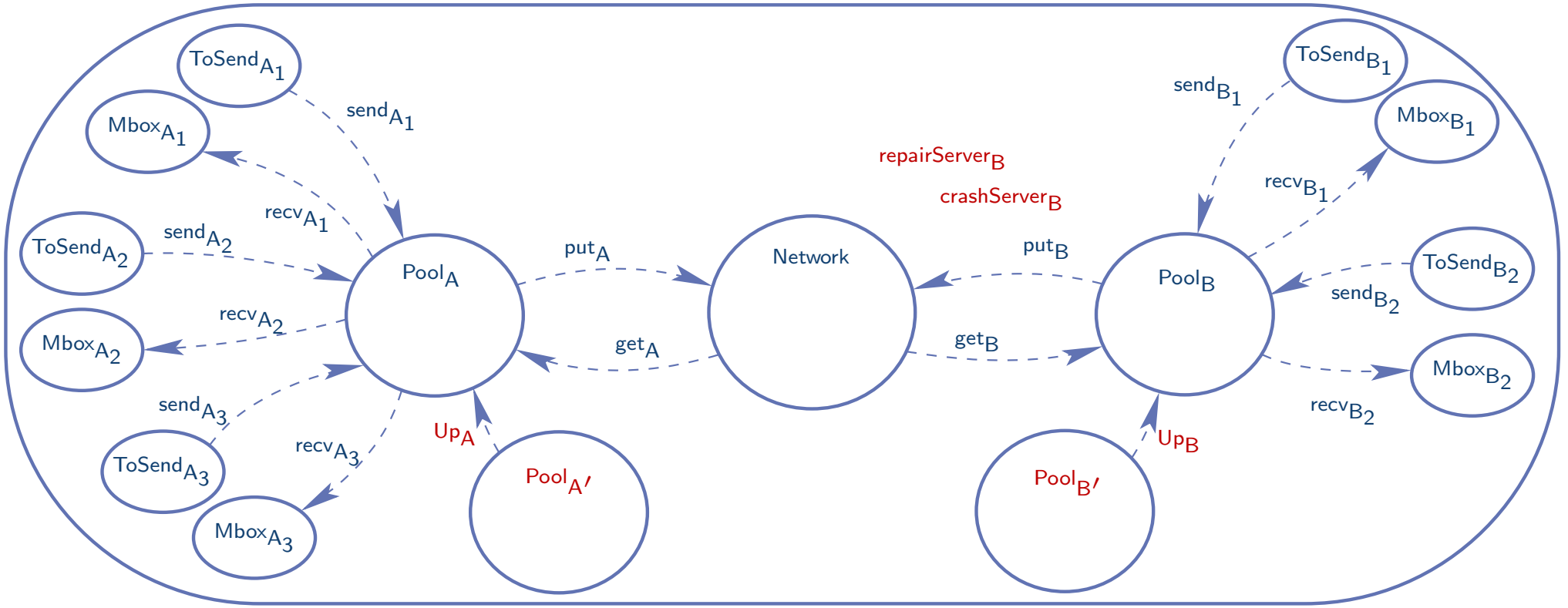
An autonomic mail system



- *Self-healing:* emergency server

$$\mathrm{repairServer}_d = \mathbf{replace} \mathrm{put}_{d'},\ \mathrm{get}_{d'},\ \mathrm{DownIn}_d,\ \mathrm{DownOut}_d$$
$$\mathbf{by} \mathrm{put}_d,\ \mathrm{get}_d,\ \mathrm{Up}_d$$
$$\mathbf{if} recover(d)$$

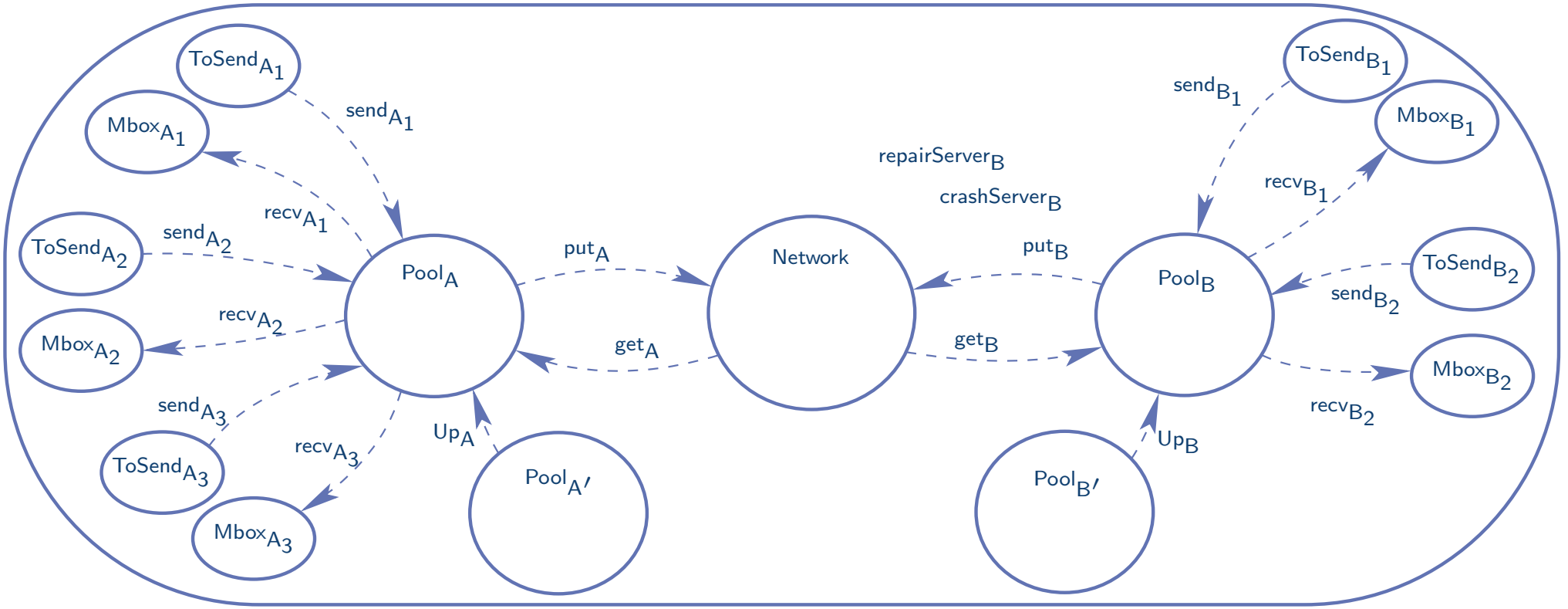
An autonomic mail system



- *Self-healing:* emergency server

$$\mathrm{repairServer}_d = \mathbf{replace} \mathrm{put}_{d'},\ \mathrm{get}_{d'},\ \mathrm{DownIn}_d,\ \mathrm{DownOut}_d$$
$$\mathbf{by} \mathrm{put}_d,\ \mathrm{get}_d,\ \mathrm{Up}_d$$
$$\mathbf{if} recover(d)$$

An autonomic mail system



- *Self-optimization:* load balancing between main servers and their emergency servers
- *Self-protection:* remove spam/viruses
- *Self-configuration:* clients may move from domains to domains

Conclusion

- A new vision:
  - reactions are molecules (active molecules): higher-order property
  - computations are triggered by adding active molecules
  - properties are ensured by adding appropriate reactive molecules
- Chemical metaphor is well suited for the specification of autonomic systems:
  - reactive and adaptive system (“autonomy”)
  - reactions keep local properties satisfied
  - properties (healing, optimization, protection, etc.) can be expressed independently as molecules