

# Reversible process algebras and self-organisation.

Jean Krivine<sup>a</sup>

February 14, 2006

---

<sup>a</sup>INRIA-Rocquencourt; Université Paris 6

## Motivation

Using process algebra as a model to describe bio molecular interactions.

- Modeling with  $\pi$ -calculus:
  - Modular way of implementing distributed systems
  - Well studied techniques to analyse them
- Correspondence with biology?
  - Name sharing = binding, processes = proteins ...
  - Not really. In fact we are more *implementing* than *modelling*
- Correspondence would be useful
  - No arbitrary representation.
  - If we have a model and we cannot represent something, then one has to search for a new mechanism and transcribe it into the model...

## Outline.

1. Transactional systems.
2. Reversible process algebra.
3. Discussion.

## Transactional systems: a computer science example.

A transactional system is composed of independent processes trying to reach for an agreement. They may synchronise on communication channels.

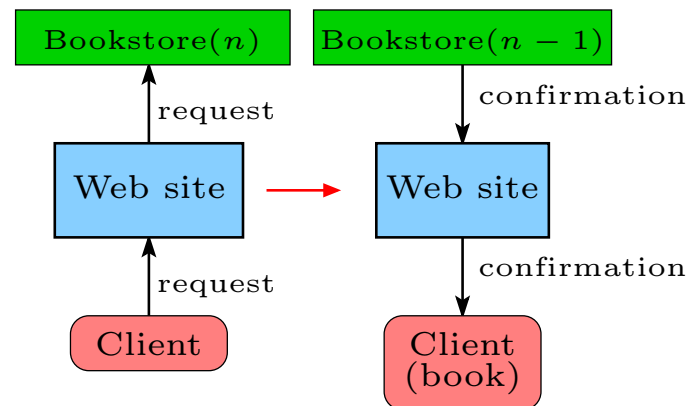


Figure 1: Bookstore query: a simple transactional system.

## Transactional systems: a biological example.

Biomolecular interactions may also be viewed as a transactional system.

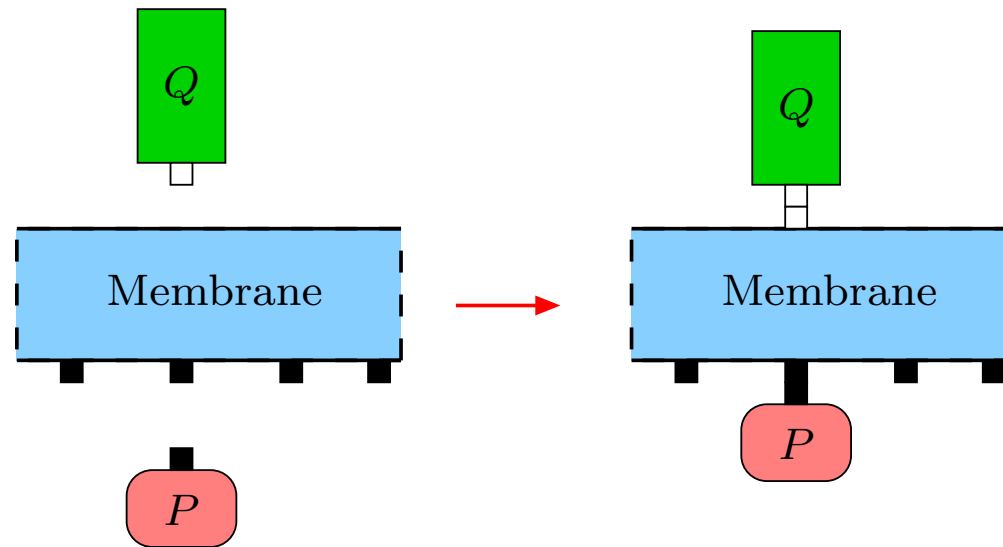


Figure 2: The membrane interface: another transactional system.

## Transactional systems: concurrency creates deadlocks

Both transactional systems we saw, contained the optimal number of participants to trigger a reaction. But concurrency entails deadlocks.

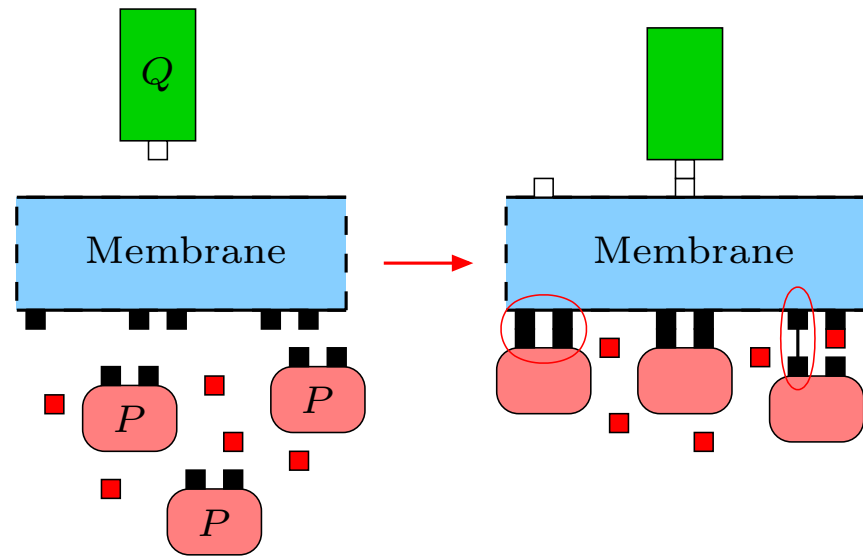
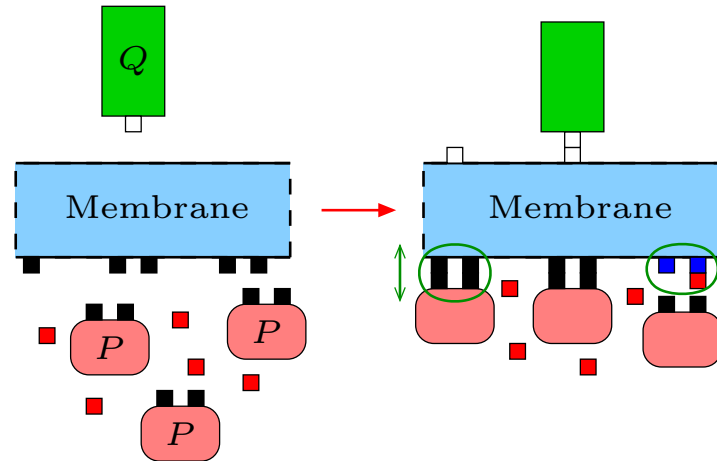


Figure 3: Transactional systems with concurrency.

## Transactional systems: $\pi$ -calculus representation is arbitrary.



$$\begin{aligned}
 P &:= (\nu b)(\nu unbind) \bar{a}\langle b, unbind \rangle.(\bar{b} \parallel \bar{b}); \overline{unbind}.P + \bar{c} \\
 \text{Membrane} &:= a(b, unbind).(b \parallel b); (unbind.\text{Membrane} + c.\bar{d}) \\
 &\parallel \dots \parallel a(b, unbind).(b \parallel b); (unbind.\text{Membrane} + c.\bar{d}) \\
 \text{Inhibitor} &:= (\nu x) \bar{a}\langle x, x \rangle.\bar{x} \\
 Q &:= d.Q'
 \end{aligned}$$

Hack: exchange private names to prevent interferences and use recursion to encode backtracking.

## Transactional systems: blind collision model

We use CCS to model blind interactions and see how we can find a “natural” way to treat deadlocks.

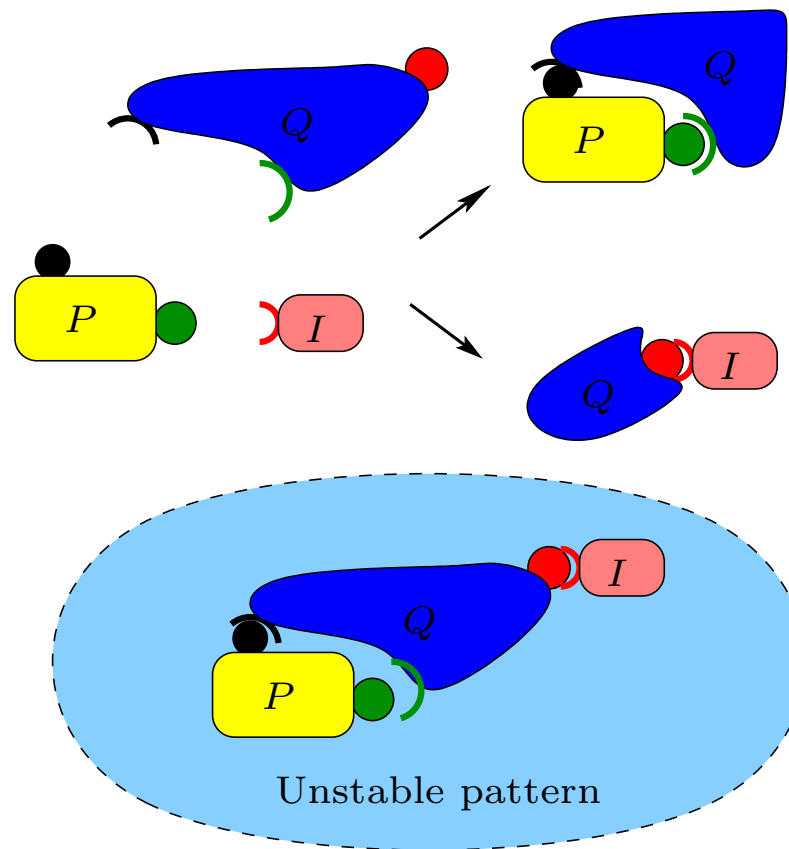
$$\textit{Processes} := a.P \mid P + P \mid (P \parallel P) \mid \nu x(P)$$

$$\textit{Complexation} : (\bar{a}.P + Q \parallel a.P' + Q') \rightarrow (P \parallel P')$$

In this calculus proteins engage in blind interactions (no artefact transition is permitted). Action prefixing processes expresses bindings and/or shape change of reactants.

## Example

$$P := (\bar{a} \parallel \bar{b}); \overline{fold} \quad I := \overline{inhibit}.0 \quad Q := a \parallel b; (fold + inhibit)$$



## Transactional systems: the programmer's point of view.

One can classify the programmer's job to implement a transaction into two parts:

- **Forward programming:** How to proceed with each micro transaction given an optimal number of participants.
- **Backward programming:** Program a way out of all possible deadlocks or unwanted behaviours entailed by the interleaving of micro transactions.

The first point corresponds to an intuitive way of programming because it involves local reasoning about interaction capabilities but the second does not!

## Transactional systems: what is Nature's point of view?

The correspondence with bio-molecular interactions could be the following:

- **Forward programming:** specify what are the possible complexations involving the reactants (which ligands are compatible?).
- **Backward programming:** each useful complex is stable (irreversible), whereas partial construction steps are unstable (reversible). It could explain *why* nature needs reversible interactions.

It can be viewed as a game between proteins (trying to bind with each others) and the system's dynamics (trying to break complexes).

## Outline.

1. Transactional systems.
2. Reversible process algebra.
3. Discussion.

## Reversible process algebra: specifying reactant interfaces.

We keep with the Calculus of Communicating Systems (CCS) because it gives conspicuous means to represent reactant's binding capabilities (forward behaviour).

<b>Protein/Process</b>	$P$	$:=$	$a.P$	<b>Binding/Communication</b> on $a$
			$  (P \parallel P)$	Parallel composition
			$  a.P + b.P$	Choice of interaction
			$  \nu a(P)$	Private <b>ligand/channel</b>

Channels  $\{a, b, \dots\}$  represent ligands and we suppose that each channel is equipped with a compatibility relation

$$\bar{a} := \{a' \mid (a, a') \in \text{bounds}\}.$$

## Reversible process algebra: (some) technical details.

$$\begin{array}{l}
 \text{Unstable bindings} \quad \left\{ \begin{array}{l} m \triangleright a.P + Q \quad \xrightarrow{a}_{\beta} \langle \beta, a, Q \rangle \cdot m \triangleright P \\ \langle \beta, a, Q \rangle \cdot m \triangleright P \quad \xrightarrow{a^{-}}_{\beta} m \triangleright a.P + Q \end{array} \right. \\
 \text{Stable binding} \quad m \triangleright \underline{a}.P + Q \xrightarrow{\underline{a}} \langle \underline{a} \rangle \cdot m \triangleright P
 \end{array}$$

$$\frac{A \xleftrightarrow{a^{-} a}_{\beta} A' \quad B \xleftrightarrow{a'^{-} a'}_{\beta} B' \quad a' \in \bar{a}}{A \parallel B \xleftrightarrow{\tau^{-} \tau}_{\beta} A' \parallel B'} \text{(Complex/decomplex)}$$

$$\text{(stable-complex)} \quad \frac{A \xrightarrow{\underline{a}} A' \quad B \xrightarrow{a'} B' \quad a' \in \bar{a}}{A \parallel B \xrightarrow{\underline{\tau}} A' \parallel B'} \quad \frac{A \xleftrightarrow{a^{-} a} A'}{A \parallel B \xleftrightarrow{a^{-} a} A' \parallel B} \text{(par)}$$

## Reversible process algebra: global behaviour.

**Theorem 1 (Informal version)** *If every micro transaction (with no concurrency) is correct then the global transaction (with concurrency) in the reversible algebra is correct.<sup>a</sup>*

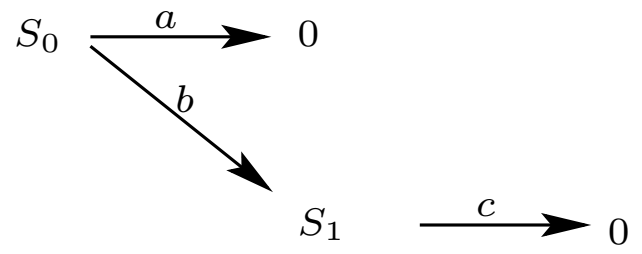
Technically,  $\text{CTS}(P) \approx (\langle \rangle \triangleright P)$  where  $\text{CTS}(P)$  is the transition system of  $P$  restricted to causal traces ending by an irreversible action.

---

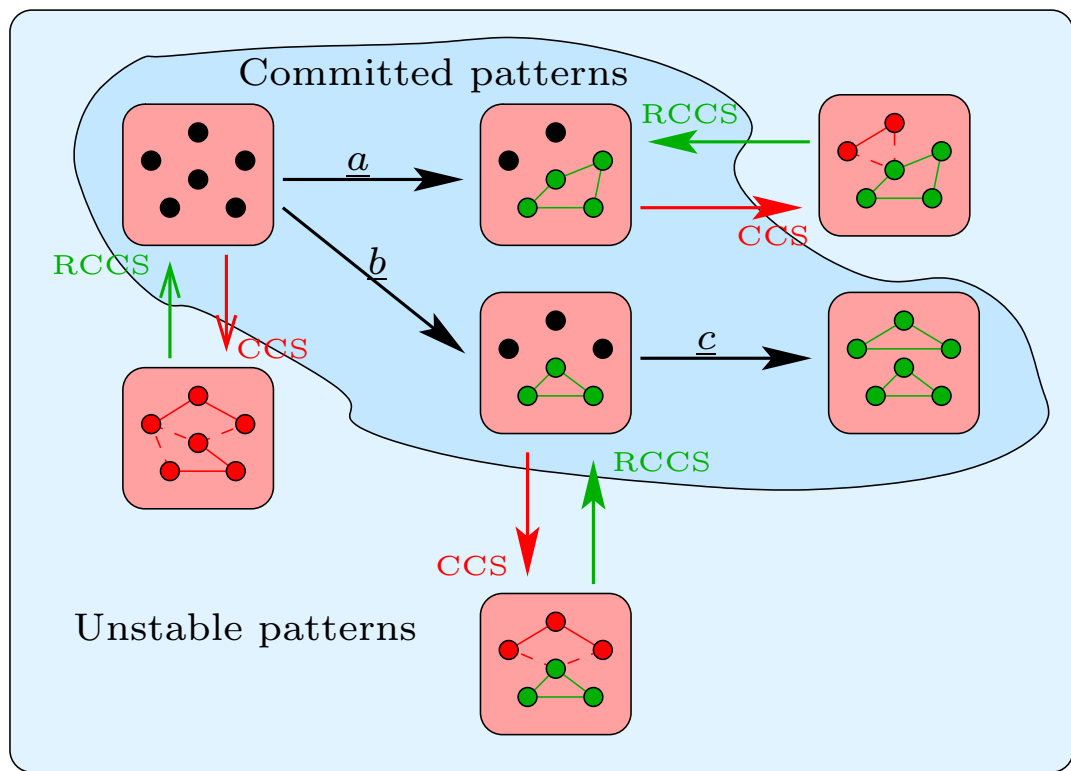
<sup>a</sup>Danos and Krivine 2005.

# Reversible process algebra: unstable and stable states

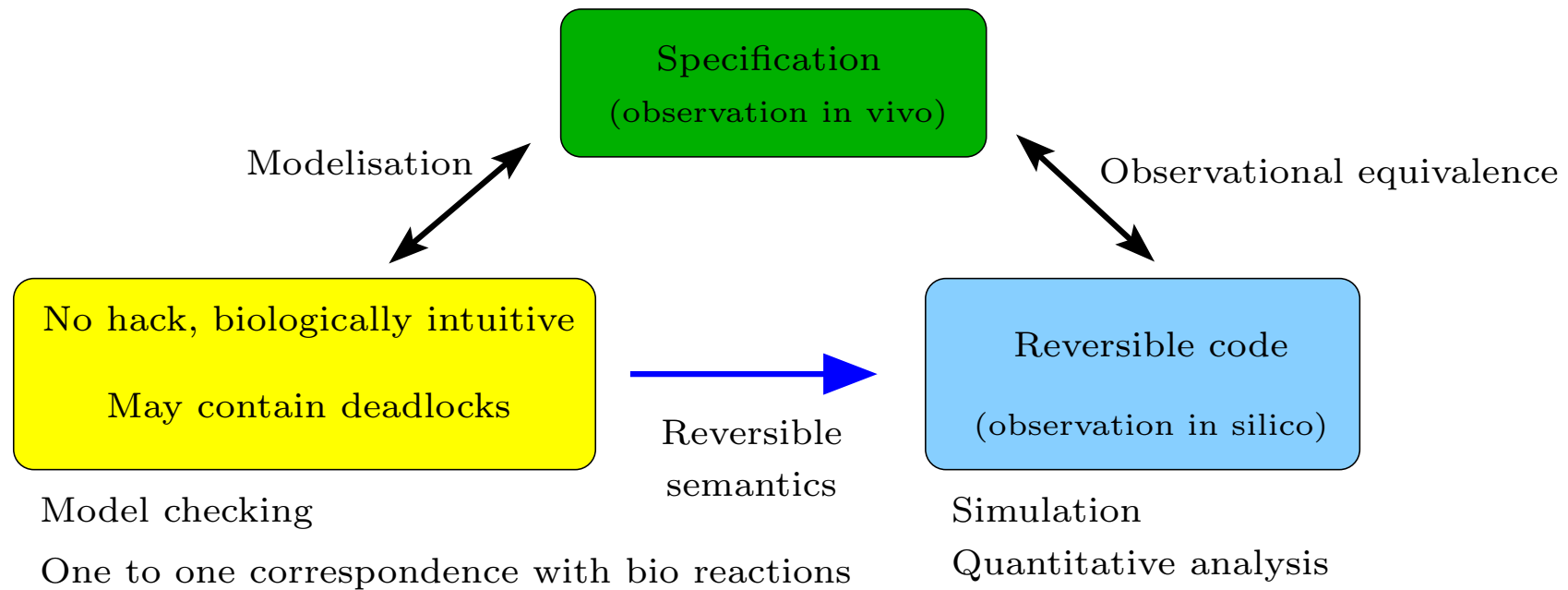
In vivo



In silico



## Reversible process algebra: sketch



## Reversible process algebra: back to the example.

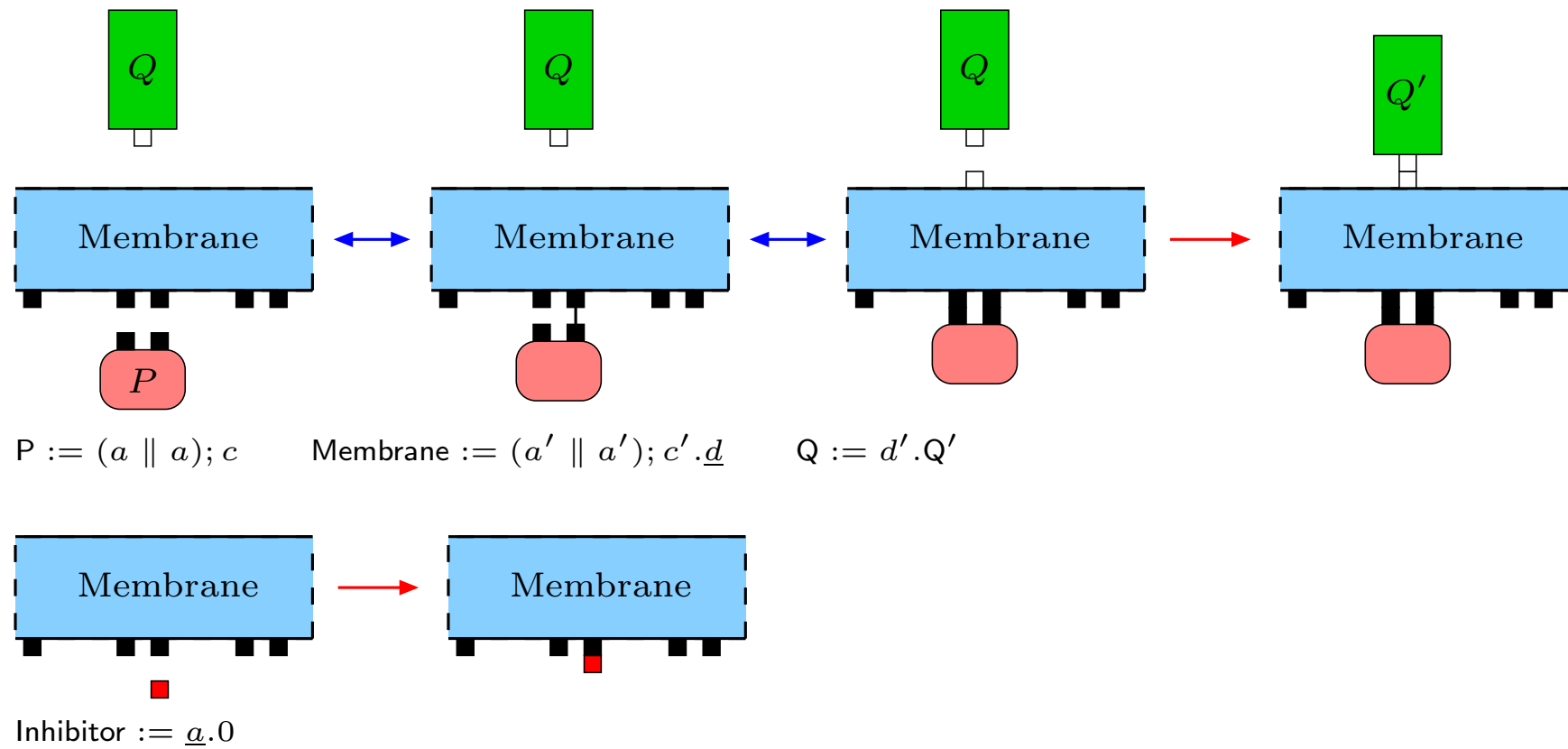
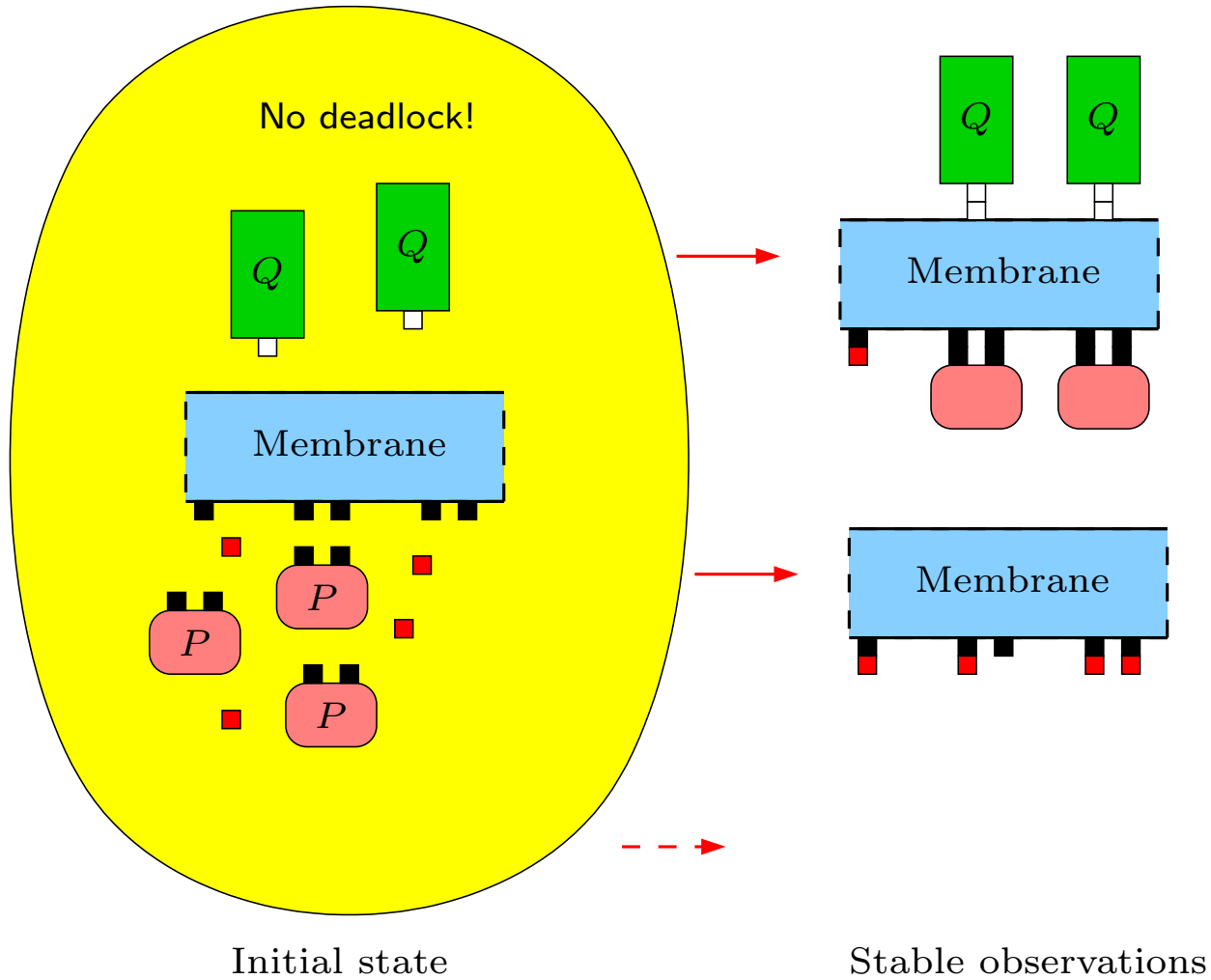


Figure 4: Encoding with a reversible algebra

# Reversible process algebra: Correctness.



## Outline.

1. Transactional systems.
2. Reversible process algebra.
3. Discussion.

## Discussion: model or implementation?

$$P \quad := \quad (\nu b)(\nu unbind) \bar{a}\langle b, unbind \rangle.(\bar{b} \parallel \bar{b}); \overline{unbind}.P + \bar{c}$$

$$\begin{aligned} \text{Membrane} &:= a(b, unbind).(b \parallel b); (unbind.\text{Membrane} + c.d) \\ &\parallel \dots \parallel a(b, unbind).(b \parallel b); (unbind.\text{Membrane} + c.d) \end{aligned}$$

$$\text{Inhibitor} \quad := \quad (\nu x) \bar{a}\langle x, x \rangle$$

$$Q \quad := \quad \bar{d}.Q'$$

---

---

$$P \quad := \quad (\bar{b} \parallel \bar{b}); \bar{c}$$

$$\text{Membrane} \quad := \quad (b \parallel b); c.\underline{d} \parallel \dots \parallel (b \parallel b); c.\underline{d}$$

$$\text{Inhibitor} \quad := \quad \underline{\bar{b}}$$

$$Q \quad := \quad \bar{d}.Q'$$

## Discussion: benefits

Abstracting away from backtracking may be useful for both quantitative and qualitative analysis:

- Forward code is smaller than codes containing explicit handling of deadlocks. It is easier to perform model checking on it.
- Rates of biological interactions  $R \rightleftharpoons R'$  have a direct correspondence in the reversible algebra: they correspond to giving probabilities to forward and backward transitions.

## Discussion: which algebra?

This approach can be adapted to different kind of process algebra<sup>a,b</sup>. The general philosophy we advocate does not depend on the choice of algebra:

$$m \triangleright P$$

- Start from a minimal language that models only biological interactions (no artefact transitions).
- Build an abstract machine that implements backtracking to undo unstable complexes.
- Perform simulation by giving parameters to the abstract machine.
- Perform model checking (such as reachability analysis) on forward code.

---

<sup>a</sup>I. Phillips and I. Ulidowski: Reversing Algebraic Process Calculi

<sup>b</sup>J. Krivine: Reversible Process Algebra (PhD Thesis)