

Reversing Algebraic Process Calculi

Irek Ulidowski

University of Leicester

joint work with Iain Phillips from Imperial College London

2nd March 2006

Introduction

Usually computation goes forward only $P \rightarrow Q$

But sometimes reversible computation $P \leftrightarrow Q$ can be helpful

- Landauer (1961): irreversibility generates heat.
- Desirability of reversible computational models

Abramsky (2001): Reversible automata; compilation of functional programs

Zuliani (2001): program transformation from irreversible to reversible

Mu, Hu and Takeichi (2004): bi-directional functional language

Bergstra, Ponse and van Wamel (1993): process algebra with backtracking

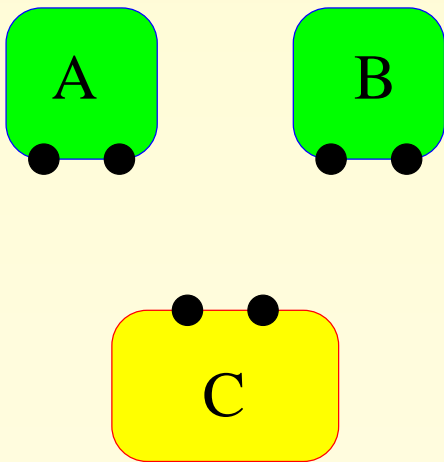
- Danos and Krivine (2003, 2004): reversible CCS; biological systems

A simple example—proteins and complex

Inspired by an example of Danos and Krivine (2003).

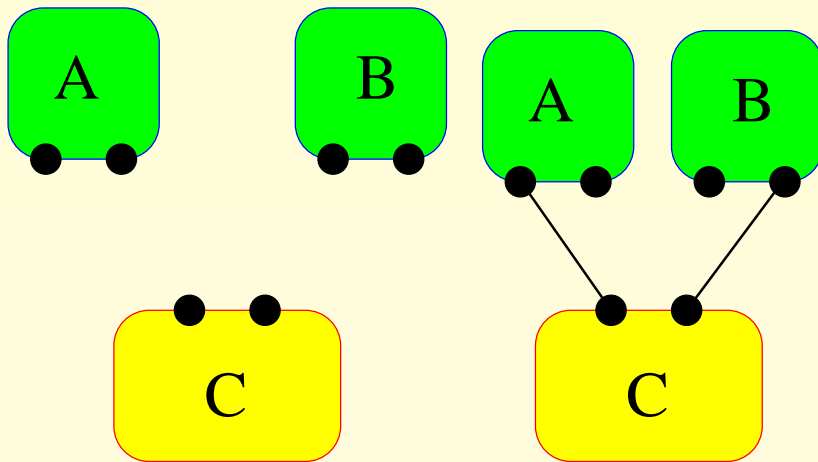
A simple example—proteins and complex

Inspired by an example of Danos and Krivine (2003).



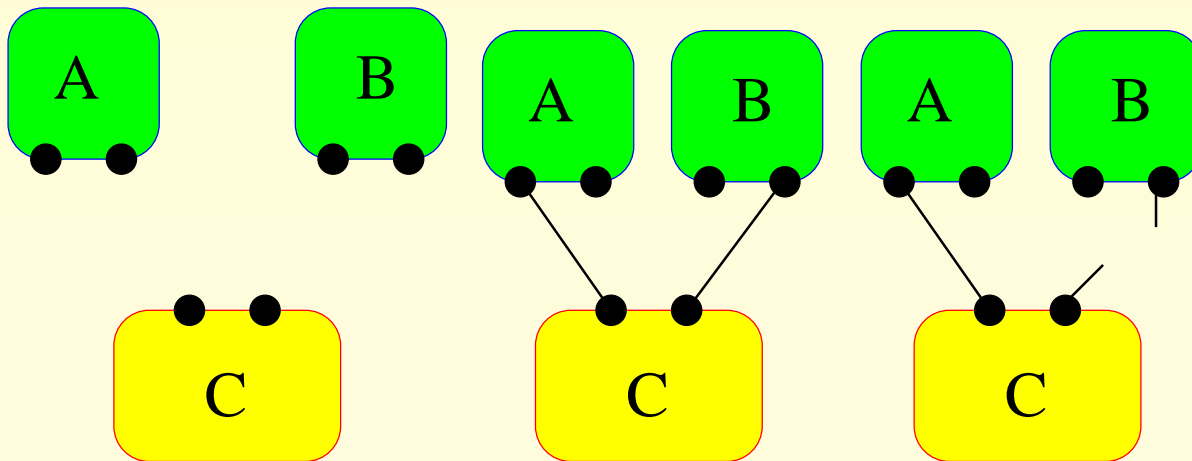
A simple example—proteins and complex

Inspired by an example of Danos and Krivine (2003).



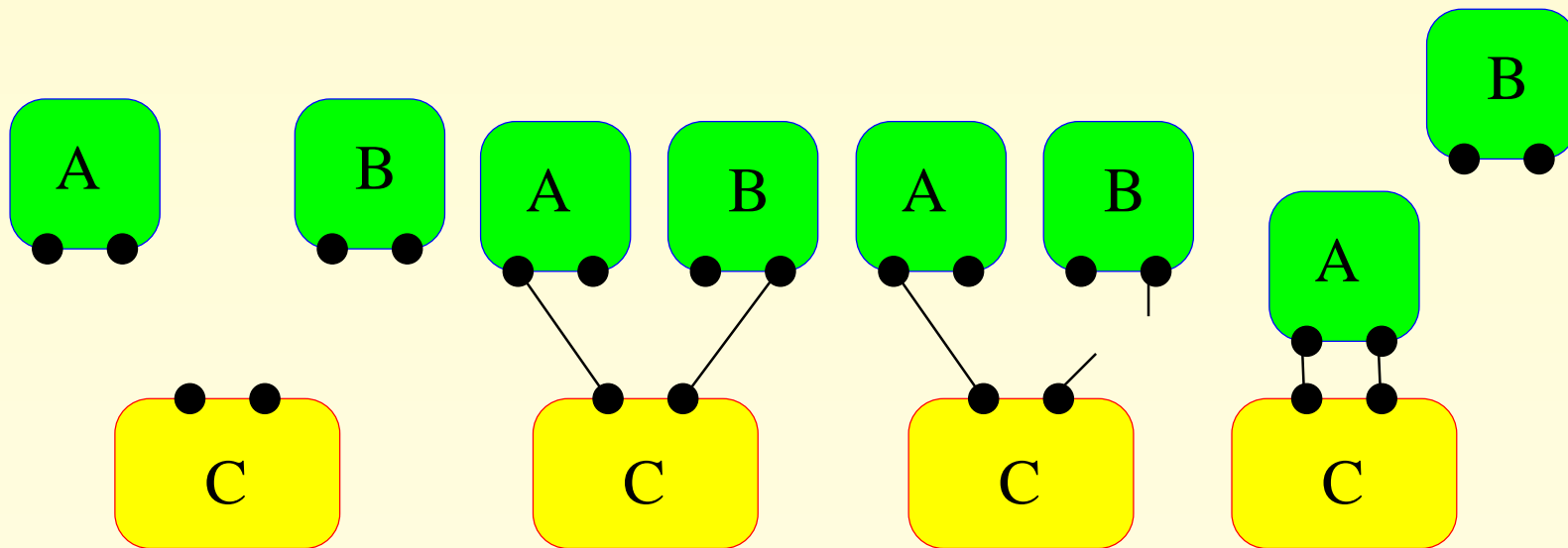
A simple example—proteins and complex

Inspired by an example of Danos and Krivine (2003).



A simple example—proteins and complex

Inspired by an example of Danos and Krivine (2003).



Reversing transitions

$a.P \xrightarrow{a} P$ can be reversed: $P \overset{a}{\rightsquigarrow} a.P$ for any a .

But a has been forgotten, and so highly nondeterministic.

Danos and Krivine (2003): introduce separate **memories** to keep track of past behaviour.

A more algebraic alternative is to **keep the past in the terms**:

$$a.P \xrightarrow{a} \underline{a}.P \quad \underline{a}.P \overset{a}{\rightsquigarrow} a.P$$

Here $\underline{a}.$ is a new unary **nonstandard** operator denoting **past action**.

Predicates: $\text{nstd}(\underline{a}.P)$ $\text{std}(a.P)$ (if $\text{std}(P)$)

CCS choice

$$a.b + c \xrightarrow{a} b$$

Keep the structure; retain the discarded alternatives: $a.b + c \xrightarrow{a} \underline{a}.b + c$
(cf. Boudol and Castellani 1994)

Can tell which alternative was taken: $\text{nstd}(\underline{a}.b) \quad \text{std}(c)$

There are meaningless processes like $\underline{a} + \underline{b}$

—not reachable from standard processes

Our Approach

Our Approach

- Past behaviour is recorded in the **syntax of terms** and not on external devices. Achieved by converting standard operators to **static** versions

Our Approach

- Past behaviour is recorded in the **syntax of terms** and not on external devices. Achieved by converting standard operators to **static** versions
- **Predicates** are used to control execution of converted terms

Our Approach

- Past behaviour is recorded in the **syntax of terms** and not on external devices. Achieved by converting standard operators to **static** versions
- **Predicates** are used to control execution of converted terms
- **One-time keys** allow to distinguish individual action occurrences

Our Approach

- **Past behaviour** is recorded in the **syntax of terms** and not on external devices. Achieved by converting standard operators to **static** versions
- **Predicates** are used to control execution of converted terms
- **One-time keys** allow to distinguish individual action occurrences
- **Symmetry** between forward and reverse SOS rules

Reversing Operators

Consider n -ary operator f with arguments in $N = \{1, \dots, n\}$.

Operators in process calculi can be classified according to their SOS rules:

Reversing Operators

Consider n -ary operator f with arguments in $N = \{1, \dots, n\}$.

Operators in process calculi can be classified according to their SOS rules:

- **Dynamic**: consumed, e.g. choice, action prefixing; arguments in $D \subseteq N$

Reversing Operators

Consider n -ary operator f with arguments in $N = \{1, \dots, n\}$.

Operators in process calculi can be classified according to their SOS rules:

- **Dynamic**: consumed, e.g. choice, action prefixing; arguments in $D \subseteq N$
- **Static**: preserved, e.g. parallel composition; arguments in $S \subseteq N$

Reversing Operators

Consider n -ary operator f with arguments in $N = \{1, \dots, n\}$.

Operators in process calculi can be classified according to their SOS rules:

- **Dynamic**: consumed, e.g. choice, action prefixing; arguments in $D \subseteq N$
- **Static**: preserved, e.g. parallel composition; arguments in $S \subseteq N$
- **Mixed**: both static and dynamic rules, e.g. sequential composition

Reversing Operators

Consider n -ary operator f with arguments in $N = \{1, \dots, n\}$.

Operators in process calculi can be classified according to their SOS rules:

- **Dynamic**: consumed, e.g. choice, action prefixing; arguments in $D \subseteq N$
- **Static**: preserved, e.g. parallel composition; arguments in $S \subseteq N$
- **Mixed**: both static and dynamic rules, e.g. sequential composition

We discuss how to reverse each category.

We convert standard \rightarrow_S into **equivalent** \rightarrow with **reverse ltr** \rightsquigarrow .

Dynamic operators

Rules either have premises or are **axioms**.

Axioms: e.g. $\frac{}{a.X \xrightarrow{a}_S X}$

A generalisation: $r \frac{}{f(\vec{X}) \xrightarrow{a}_S X_j}$ for $j \in E = N \setminus S$.

First attempt: introduce new **auxiliary** operator g_r with rules

$$\frac{}{f(\vec{X}) \xrightarrow{a} g_r(\vec{X})} \quad \frac{X_j \xrightarrow{b} X'_j}{g_r(\vec{X}) \xrightarrow{b} g_r(\vec{X}')} \text{ for all } b \text{ (sieve rule)}$$

Problem: ambiguous when reversed (premises “overlap”)

Solution: use predicates (Baeten and Verhoef 1993) as well:

$$\frac{\{\text{std}(X_k)\}_{k \in E}}{f(\vec{X}) \xrightarrow{a} g_r(\vec{X})} \quad \frac{X_j \xrightarrow{b} X'_j \quad \{\text{std}(X_k)\}_{k \in E \setminus \{j\}}}{g_r(\vec{X}) \xrightarrow{b} g_r(\vec{X}')} \text{ for all } b$$

Solution: **use predicates** (Baeten and Verhoef 1993) as well:

$$\frac{\{\text{std}(X_k)\}_{k \in E}}{f(\vec{X}) \xrightarrow{a} g_r(\vec{X})} \quad \frac{X_j \xrightarrow{b} X'_j \quad \{\text{std}(X_k)\}_{k \in E \setminus \{j\}}}{g_r(\vec{X}) \xrightarrow{b} g_r(\vec{X}')} \text{ for all } b$$

$$\frac{\{\text{std}(X_k)\}_{k \in E}}{g_r(\vec{X}) \xrightarrow{a} f(\vec{X})} \quad \frac{X_j \xrightarrow{b} X'_j \quad \{\text{std}(X_k)\}_{k \in E \setminus \{j\}}}{g_r(\vec{X}) \xrightarrow{b} g_r(\vec{X}')} \text{ for all } b$$

Solution: **use predicates** (Baeten and Verhoef 1993) as well:

$$\frac{\{\text{std}(X_k)\}_{k \in E}}{f(\vec{X}) \xrightarrow{a} g_r(\vec{X})} \quad \frac{X_j \xrightarrow{b} X'_j \quad \{\text{std}(X_k)\}_{k \in E \setminus \{j\}}}{g_r(\vec{X}) \xrightarrow{b} g_r(\vec{X}')} \text{ for all } b$$

$$\frac{\{\text{std}(X_k)\}_{k \in E}}{g_r(\vec{X}) \xrightarrow{a} f(\vec{X})} \quad \frac{X_j \xrightarrow{b} X'_j \quad \{\text{std}(X_k)\}_{k \in E \setminus \{j\}}}{g_r(\vec{X}) \xrightarrow{b} g_r(\vec{X}')} \text{ for all } b$$

Redefined **action prefixing** uses **past actions** \underline{a} :

$$\frac{\text{std}(X)}{a.X \xrightarrow{a} \underline{a}.X} \quad \frac{X \xrightarrow{b} X'}{\underline{a}.X \xrightarrow{b} \underline{a}.X'} \text{ for all } b \quad \frac{\text{std}(X)}{\underline{a}.X \xrightarrow{a} a.X} \quad \frac{X \xrightarrow{b} X'}{\underline{a}.X \xrightarrow{b} \underline{a}.X'} \text{ for all } b$$

Dynamic rules with premises

e.g. CCS choice: $\frac{X \xrightarrow{a}_S X'}{X + Y \xrightarrow{a}_S X'} \quad \frac{Y \xrightarrow{a}_S Y'}{X + Y \xrightarrow{a}_S Y'}$ for all actions a .

A generalisation, called a **simple choice rule**, is: $\frac{X_d \xrightarrow{a}_S X'_d}{f(\vec{X}) \xrightarrow{a}_S X'_d} \quad (d \in D)$

The idea is to **make the above rule static**: $\frac{X_d \xrightarrow{a} X'_d}{f(\vec{X}) \xrightarrow{a} f(\vec{X}')}$

Dynamic rules with premises

e.g. CCS choice: $\frac{X \xrightarrow{a}_S X'}{X + Y \xrightarrow{a}_S X'} \quad \frac{Y \xrightarrow{a}_S Y'}{X + Y \xrightarrow{a}_S Y'}$ for all actions a .

A generalisation, called a **simple choice rule**, is: $\frac{X_d \xrightarrow{a}_S X'_d}{f(\vec{X}) \xrightarrow{a}_S X'_d} \quad (d \in D)$

The idea is to **make the above rule static**: $\frac{X_d \xrightarrow{a} X'_d}{f(\vec{X}) \xrightarrow{a} f(\vec{X}')}$

We also need to allow X'_d to proceed: $\frac{X_d \xrightarrow{b} X'_d}{f(\vec{X}) \xrightarrow{b} f(\vec{X}')}$ for all b (sieve rule)

And we need to stop all other arguments from proceeding:

Again use predicates:

$$\frac{X_d \xrightarrow{a} X'_d \quad \{\text{std}(X_k)\}_{k \in E \setminus \{d\}}}{f(\vec{X}) \xrightarrow{a} f(\vec{X}')}$$

$$\frac{X_d \overset{a}{\rightsquigarrow} X'_d \quad \{\text{std}(X'_k)\}_{k \in E \setminus \{d\}}}{f(\vec{X}) \overset{a}{\rightsquigarrow} f(\vec{X}')}$$

Again use predicates:

$$\frac{X_d \xrightarrow{a} X'_d \quad \{\text{std}(X_k)\}_{k \in E \setminus \{d\}}}{f(\vec{X}) \xrightarrow{a} f(\vec{X}')}$$

$$\frac{X_d \rightsquigarrow^a X'_d \quad \{\text{std}(X'_k)\}_{k \in E \setminus \{d\}}}{f(\vec{X}) \rightsquigarrow^a f(\vec{X}')}$$

The reformulated and reverse rules for **CCS choice** (of course, for all a):

$$\frac{X \xrightarrow{a} X' \quad \text{std}(Y)}{X + Y \xrightarrow{a} X' + Y}$$

$$\frac{Y \xrightarrow{a} Y' \quad \text{std}(X)}{X + Y \xrightarrow{a} X + Y'}$$

$$\frac{X \rightsquigarrow^a X' \quad \text{std}(Y)}{X + Y \rightsquigarrow^a X' + Y}$$

$$\frac{Y \rightsquigarrow^a Y' \quad \text{std}(X)}{X + Y \rightsquigarrow^a X + Y'}$$

Unambiguous reversibility

A dynamic argument of f is **permissive** if for each $d \in D$ and each action a there is a simple choice rule for f .

If all operators are dynamic and satisfy requirements given, then if P, Q, R are reachable terms such that $P \xrightarrow{a} Q$ and $P \xrightarrow{b} R$ then $a = b$ and $Q = R$.

Static operators

Rules of the form $\frac{\{X_i \xrightarrow{a_i}_S X'_i\}_{i \in I}}{f(\vec{X}) \xrightarrow{a}_S f(\vec{X}')} (I \subseteq S)$ are called **static** rules.

Require that if two static rules r and r' have the same premises then they have the same conclusion, i.e. if $I = I'$ and all $a_i = a'_i$, then $a = a'$.

$$r \frac{\{X_i \xrightarrow{a_i}_S X'_i\}_{i \in I}}{f(\vec{X}) \xrightarrow{a}_S f(\vec{X}')} \quad r' \frac{\{X_i \xrightarrow{a'_i}_S X'_i\}_{i \in I'}}{f(\vec{X}) \xrightarrow{a'}_S f(\vec{X}')}$$

All standard operators including parallel operators, hiding and restriction satisfy this requirement.

Parallel operators

CSP-type parallel (A a set of actions, $a \in A$, b is any action $\notin A$):

$$\frac{X \xrightarrow{a}_S X' \quad Y \xrightarrow{a}_S Y'}{X \parallel_A Y \xrightarrow{a}_S X' \parallel_A Y'} \quad \frac{X \xrightarrow{b}_S X'}{X \parallel_A Y \xrightarrow{b}_S X' \parallel_A Y} \quad \frac{Y \xrightarrow{b}_S Y'}{X \parallel_A Y \xrightarrow{b}_S X \parallel_A Y'}$$

If $A = \emptyset$ we get interleaving parallel \parallel .

CCS parallel composition: for any actions a and \bar{a}

$$\frac{X \xrightarrow{a}_S X' \quad Y \xrightarrow{\bar{a}}_S Y'}{X | Y \xrightarrow{\tau}_S X' | Y'} \quad \frac{X \xrightarrow{a}_S X'}{X | Y \xrightarrow{a}_S X' | Y} \quad \frac{Y \xrightarrow{a}_S Y'}{X | Y \xrightarrow{a}_S X | Y'}$$

Reversibility versus Backtracking

$$a \parallel b \xrightarrow{a} \underline{a} \parallel b \xrightarrow{b} \underline{a} \parallel \underline{b}$$

Reversibility versus Backtracking

$$a \parallel b \xrightarrow{a} \underline{a} \parallel b \xrightarrow{b} \underline{a} \parallel \underline{b}$$

Simple backtracking: can only reverse on b : $\underline{a} \parallel \underline{b} \overset{b}{\rightsquigarrow} \underline{a} \parallel b$

Reversibility versus Backtracking

$$a \parallel b \xrightarrow{a} \underline{a} \parallel b \xrightarrow{b} \underline{a} \parallel \underline{b}$$

Simple backtracking: can only reverse on b : $\underline{a} \parallel \underline{b} \overset{b}{\rightsquigarrow} \underline{a} \parallel b$

But in applications should allow to reverse on a as well:

$$\underline{a} \parallel \underline{b} \overset{a}{\rightsquigarrow} a \parallel \underline{b}$$

Distinction between backtracking and reversing.

Reversing static rules

We reformulate static rules as

$$\frac{\{X_i \xrightarrow{a_i} X'_i\}_{i \in I} \quad \{\text{std}(X_e)\}_{e \in E}}{f(\vec{X}) \xrightarrow{a} f(\vec{X}')}$$

and then just reverse them:

$$\frac{\{X_i \overset{a_i}{\rightsquigarrow} X'_i\}_{i \in I} \quad \{\text{std}(X_e)\}_{e \in E}}{f(\vec{X}) \overset{a}{\rightsquigarrow} f(\vec{X}')}$$

Mixed operators: ACP sequential composition

Static rules phase is followed by application of single dynamic rule:

$$\frac{X \xrightarrow{a}_S X'}{X \cdot Y \xrightarrow{a}_S X' \cdot Y} \quad \frac{Y \xrightarrow{a}_S Y' \quad \text{sterm}(X)}{X \cdot Y \xrightarrow{a}_S Y'}$$

Mixed operators: ACP sequential composition

Static rules phase is followed by application of single dynamic rule:

$$\frac{X \xrightarrow{a}_S X'}{X \cdot Y \xrightarrow{a}_S X' \cdot Y} \qquad \frac{Y \xrightarrow{a}_S Y' \quad \text{sterm}(X)}{X \cdot Y \xrightarrow{a}_S Y'}$$

We use [predicates approach](#):

$$\frac{X \xrightarrow{a} X' \quad \text{std}(Y)}{X \cdot Y \xrightarrow{a} X' \cdot Y} \qquad \frac{Y \xrightarrow{a} Y' \quad \text{sterm}(X)}{X \cdot Y \xrightarrow{a} X \cdot Y'}$$

with reverse rules

$$\frac{X \overset{a}{\rightsquigarrow} X' \quad \text{std}(Y)}{X \cdot Y \overset{a}{\rightsquigarrow} X' \cdot Y} \qquad \frac{Y \overset{a}{\rightsquigarrow} Y' \quad \text{sterm}(X)}{X \cdot Y \overset{a}{\rightsquigarrow} X \cdot Y'}$$

Occurrences and Keys

The approach so far works for many standard operators.

However sometimes it is desirable to have more control over event occurrences.

- semantic anomalies related to **auto-concurrency**
- ambiguities connected with reversing **CCS communication**

We use **one-time keys**.

Concurrency and causation

(Stirling; De Nicola, Montanari and Vaandrager 1990) In a reversible world:

$$a \mid b \neq a.b + b.a$$

We have $a \mid b \xrightarrow{a} \xrightarrow{b} \underline{a} \mid \underline{b} \overset{a}{\rightsquigarrow}$ but $a.b + b.a \xrightarrow{a} \xrightarrow{b} \underline{a}.\underline{b} + b.a \not\rightsquigarrow$.

Concurrency and causation

(Stirling; De Nicola, Montanari and Vaandrager 1990) In a reversible world:

$$a \mid b \neq a.b + b.a$$

We have $a \mid b \xrightarrow{a} \xrightarrow{b} \underline{a} \mid \underline{b} \overset{a}{\rightsquigarrow}$ but $a.b + b.a \xrightarrow{a} \xrightarrow{b} \underline{a}.\underline{b} + b.a \not\xrightarrow{a}$.

In $a \mid b \xrightarrow{a} \xrightarrow{b} \overset{a}{\rightsquigarrow}$ b does not **cause** a and a does not **cause** b .

We have **observed concurrency**.

Concurrency and causation

(Stirling; De Nicola, Montanari and Vaandrager 1990) In a reversible world:

$$a \mid b \neq a.b + b.a$$

We have $a \mid b \xrightarrow{a} \underline{a} \mid \underline{b} \xrightarrow{b} \underline{a} \mid \underline{b} \xrightarrow{a} \underline{a} \mid \underline{b}$ but $a.b + b.a \xrightarrow{a} \underline{a}.b + b.a \not\xrightarrow{b}$.

In $a \mid b \xrightarrow{a} \underline{a} \mid \underline{b} \xrightarrow{b} \underline{a} \mid \underline{b} \xrightarrow{a} \underline{a} \mid \underline{b}$ b does not **cause** a and a does not **cause** b .

We have **observed concurrency**.

Anomaly: what about $a \mid a$ (auto-concurrency) and $a.a$?

To distinguish, need to keep track of occurrences of a .

Keys

Dynamically choose a **fresh key** (m, n, \dots) when performing action:

$$a \mid b$$

Keys

Dynamically choose a **fresh key** (m, n, \dots) when performing action:

$$a \mid b \xrightarrow{a[m]} a[m] \mid b$$

Keys

Dynamically choose a **fresh key** (m, n, \dots) when performing action:

$$a \mid b \xrightarrow{a[m]} a[m] \mid b \xrightarrow{b[n]} a[m] \mid b[n]$$

Keys

Dynamically choose a **fresh key** (m, n, \dots) when performing action:

$$a \mid b \xrightarrow{a[m]} a[m] \mid b \xrightarrow{b[n]} a[m] \mid b[n] \xrightarrow{\sim} a \mid b[n] \quad (m \neq n)$$

Keys

Dynamically choose a **fresh key** (m, n, \dots) when performing action:

$$a \mid b \xrightarrow{a[m]} a[m] \mid b \xrightarrow{b[n]} a[m] \mid b[n] \xrightarrow{\sim} a \mid b[n] \quad (m \neq n)$$

Then we can distinguish $a \mid a$ from $a.a$:

$$a \mid a \xrightarrow{a[m]} a[m] \mid a \xrightarrow{a[n]} a[m] \mid a[n] \xrightarrow{\sim} a \mid a[n] \quad (m \neq n)$$

Keys

Dynamically choose a **fresh key** (m, n, \dots) when performing action:

$$a \mid b \xrightarrow{a[m]} a[m] \mid b \xrightarrow{b[n]} a[m] \mid b[n] \overset{a[m]}{\rightsquigarrow} a \mid b[n] \quad (m \neq n)$$

Then we can distinguish $a \mid a$ from $a.a$:

$$a \mid a \xrightarrow{a[m]} a[m] \mid a \xrightarrow{a[n]} a[m] \mid a[n] \overset{a[m]}{\rightsquigarrow} a \mid a[n] \quad (m \neq n)$$

By contrast

$$a.a \xrightarrow{a[m]} a[m].a \xrightarrow{a[n]} a[m].a[n] \not\overset{a[m]}{\rightsquigarrow} \quad (m \neq n)$$

Synchronisation and Keys

Without keys we may have strange behaviour:

$$a \mid \bar{a} \xrightarrow{\tau} \underline{a} \mid \underline{\bar{a}} \rightsquigarrow^a a \mid \underline{\bar{a}}$$

Desirable to link occurrences of a and \bar{a} together

Synchronisation and Keys

Without keys we may have strange behaviour:

$$a \mid \bar{a} \xrightarrow{\tau} \underline{a} \mid \underline{\bar{a}} \rightsquigarrow^a a \mid \underline{\bar{a}}$$

Desirable to link occurrences of a and \bar{a} together

Our solution: use keys to **lock** the two partners in a communication

$$a \mid \bar{a} \xrightarrow{\tau} a[m] \mid \bar{a}[m]$$

The partners agree on the key m . Can only reverse together

Can also proceed separately; can reverse in either order, but not together.

$$a \mid \bar{a} \xrightarrow{a[m]} a[m] \mid \bar{a} \xrightarrow{\bar{a}[n]} a[m] \mid \bar{a}[n] \quad (m \neq n)$$

SOS Rules with Keys

Action labels are now of the form $a[m]$.

For each rule r add predicates $\text{fresh}[m](X_i)$ to the premises of r for each **static** i that does not already appear in the premises of r .

SOS Rules with Keys

Action labels are now of the form $a[m]$.

For each rule r add predicates $\text{fresh}[m](X_i)$ to the premises of r for each **static** i that does not already appear in the premises of r .

CCS prefixing with keys: past actions \underline{a} are now $a[m]$:

$$\frac{\text{std}(X)}{a.X \xrightarrow{a[m]} a[m].X} \quad \frac{X \xrightarrow{b[n]} X'}{a[m].X \xrightarrow{b[n]} a[m].X'} \quad m \neq n$$

The reformulated **CCS parallel**:

$$\frac{X \xrightarrow{a[m]} X' \quad \text{fresh}[m](Y)}{X | Y \xrightarrow{a[m]} X' | Y} \quad \frac{Y \xrightarrow{a[m]} Y' \quad \text{fresh}[m](X)}{X | Y \xrightarrow{a[m]} X | Y'} \quad \frac{X \xrightarrow{a[m]} X' \quad Y \xrightarrow{\bar{a}[m]} Y'}{X | Y \xrightarrow{\tau[m]} X' | Y'}$$

SOS Rules for CCSK

$$\frac{\text{std}(X)}{a.X \xrightarrow{a[m]} a[m].X} \quad \frac{X \xrightarrow{b[n]} X'}{a[m].X \xrightarrow{b[n]} a[m].X'} \quad m \neq n$$

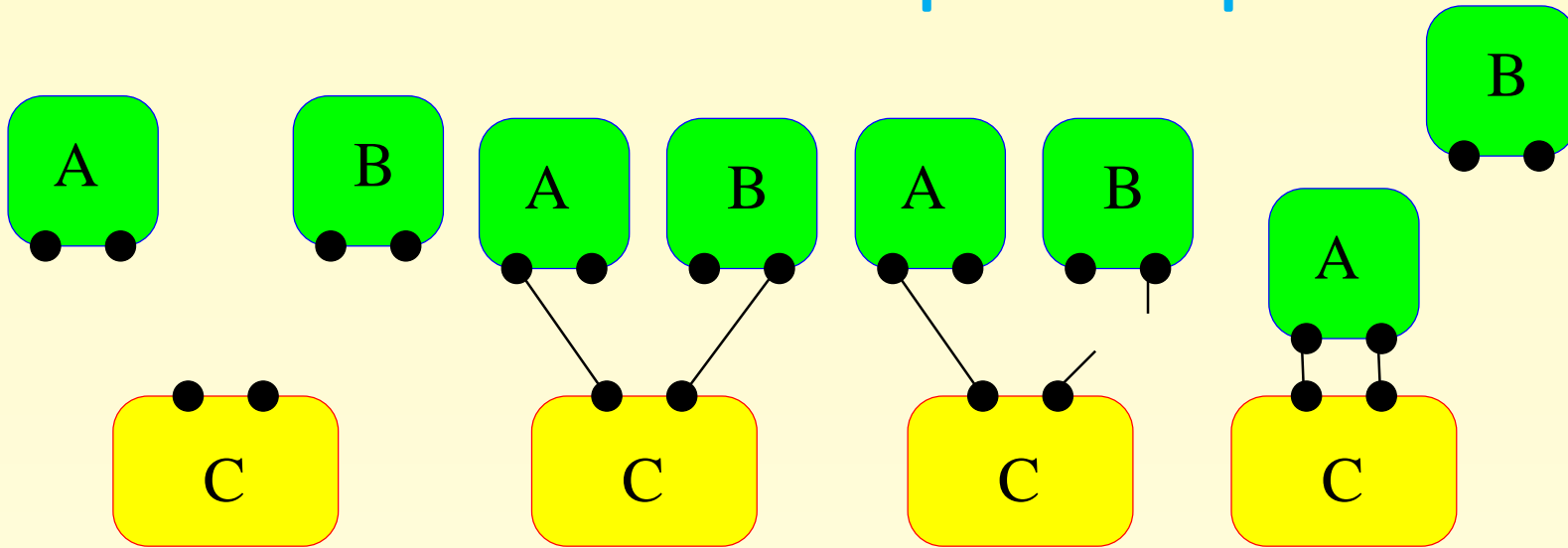
$$\frac{X \xrightarrow{a[m]} X' \quad \text{std}(Y)}{X + Y \xrightarrow{a[m]} X' + Y} \quad \frac{Y \xrightarrow{a[m]} Y' \quad \text{std}(X)}{X + Y \xrightarrow{a[m]} X + Y'}$$

$$\frac{X \xrightarrow{a[m]} X' \quad \text{fresh}[m](Y)}{X | Y \xrightarrow{a[m]} X' | Y} \quad \frac{Y \xrightarrow{a[m]} Y' \quad \text{fresh}[m](X)}{X | Y \xrightarrow{a[m]} X | Y'} \quad \frac{X \xrightarrow{a[m]} X' \quad Y \xrightarrow{\bar{a}[m]} Y'}{X | Y \xrightarrow{\tau[m]} X' | Y'}$$

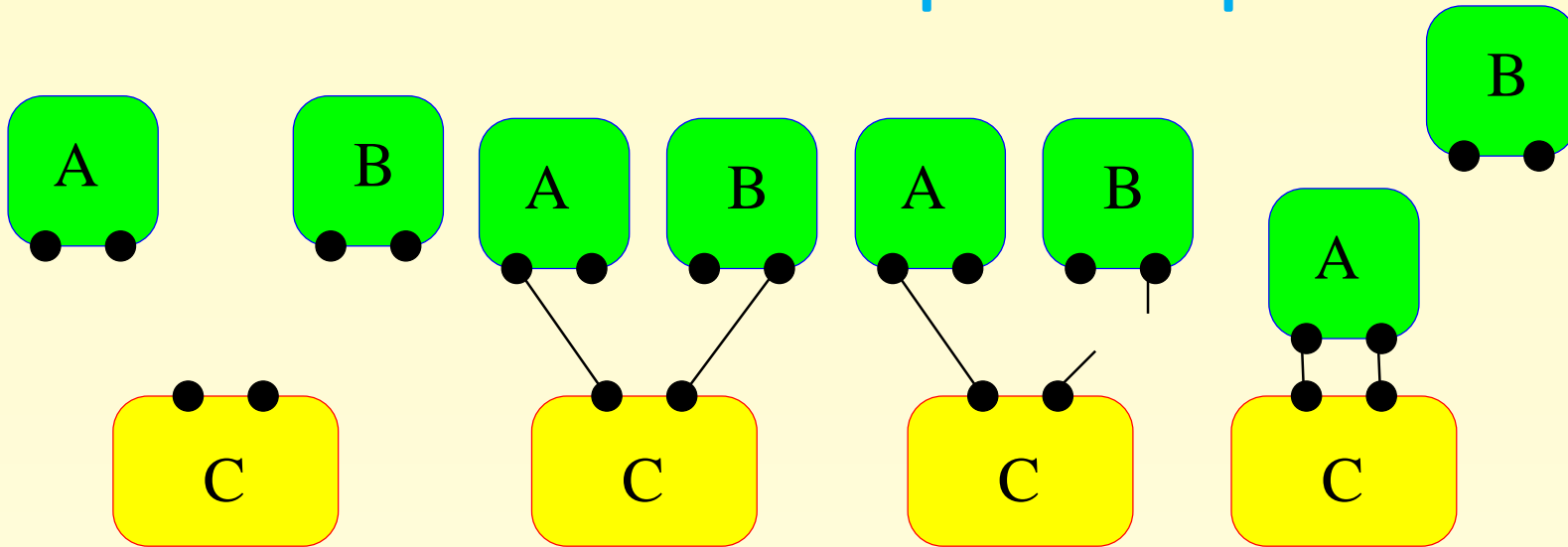
$$\frac{X \xrightarrow{a[m]} X'}{X \setminus A \xrightarrow{a[m]} X' \setminus A} \quad a \notin A \quad \frac{X \xrightarrow{a[m]} X'}{X[f] \xrightarrow{f(a)[m]} X'[f]}$$

Proteins and complex example

Proteins and complex example



Proteins and complex example



We shall use actions, sequential composition, parallel and restriction:

$$A \stackrel{\text{df}}{=} (\bar{c}|\bar{d}) \cdot \bar{e} \cdot A, \quad B \equiv A \quad \text{and} \quad C \stackrel{\text{df}}{=} (c|d) \cdot e \cdot P$$

The system is $(A \mid C \mid B) \setminus \{c, d, e\}$. We omit restriction $\setminus \{c, d, e\}$.

Firstly, $A \mid C \mid B \xrightarrow{\tau[m]} (\bar{c}[m]|\bar{d}) \cdot \bar{e} \cdot A \mid (c[m]|d) \cdot e \cdot P \mid B$.

Then, B joins in just before A is able to complete the binding:

$$\xrightarrow{\tau[n]} (\bar{c}[m]|\bar{d}) \cdot \bar{e} \cdot A \mid (c[m]|d[n]) \cdot e \cdot P \mid (\bar{c}|\bar{d}[n]) \cdot \bar{e} \cdot B$$

Neither A nor B can complete the binding, blocking communication on e .

Then, B joins in just before A is able to complete the binding:

$$\xrightarrow{\tau^{[n]}} (\bar{c}[m]|\bar{d}) \cdot \bar{e} \cdot A \mid (c[m]|d[n]) \cdot e \cdot P \mid (\bar{c}|\bar{d}[n]) \cdot \bar{e} \cdot B$$

Neither A nor B can complete the binding, blocking communication on e .

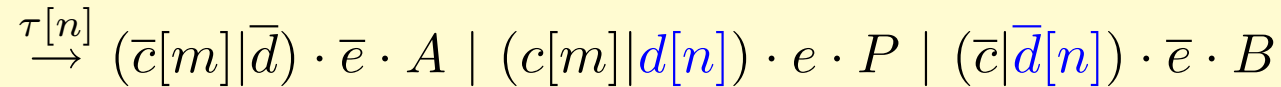
One of the proteins, B , reverses its partial binding and lets the other, A , in:

$$\xrightarrow{\tau^{[n]}} (\bar{c}[m]|\bar{d}) \cdot \bar{e} \cdot A \mid (c[m]|d) \cdot e \cdot P \mid B$$

$$\xrightarrow{\tau^{[k]}} (\bar{c}[m]|\bar{d}[k]) \cdot \bar{e} \cdot A \mid (c[m]|d[k]) \cdot e \cdot P \mid B$$

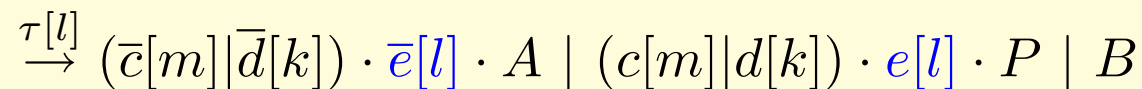
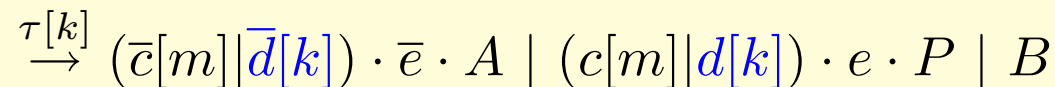
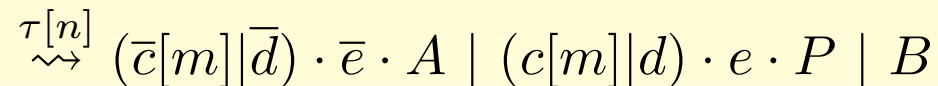
$$\xrightarrow{\tau^{[l]}} (\bar{c}[m]|\bar{d}[k]) \cdot \bar{e}[l] \cdot A \mid (c[m]|d[k]) \cdot e[l] \cdot P \mid B$$

Then, B joins in just before A is able to complete the binding:

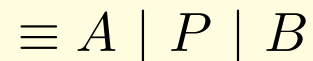


Neither A nor B can complete the binding, blocking communication on e .

One of the proteins, B , reverses its partial binding and lets the other, A , in:



In nature actions like e are **irreversible**. Then,



Conservativity

Nonstandard processes can be **pruned** to corresponding standard processes:

$$\pi(a[m].b.c) = b.c \qquad \pi(a.b + c[m].d) = d$$

Conservativity The new forward transition relation \rightarrow is in a sense conservative over the standard \rightarrow_S :

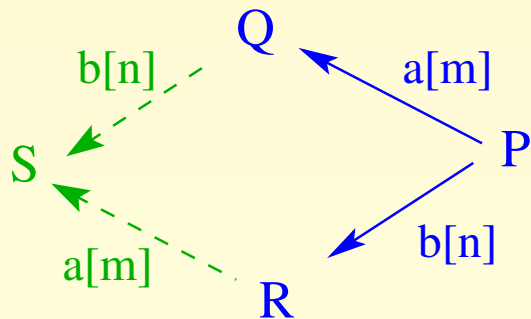
- If $P \xrightarrow{a[m]} P'$ then $\pi(P) \xrightarrow{a}_S \pi(P')$
- If $\pi(P) \xrightarrow{a}_S P'$ then $\forall m$ fresh in $P \exists P'' . P \xrightarrow{a[m]} P''$ and $\pi(P'') = P'$.

Properties of \rightarrow and \rightsquigarrow

- **Well-founded** (WF) property: there is no infinite reverse computation.
- **Unique Transitions** (UT) property: if $P \xrightarrow{a} Q$ and $P \xrightarrow{b} Q$ then $a = b$.
- No repeated labels in forward computations: cannot have $P \xrightarrow{a} \xrightarrow{s} \xrightarrow{a}$.
- **Event determinism** (ED) [cf. van Glabbeek 96]: if $P \xrightarrow{a} Q$ and $P \xrightarrow{a} R$, and (P, a, Q) and (P, a, R) are occurrences of the **same event**, then $Q = R$.
- **Reverse Diamond** and **Forward Diamond** properties.

Reverse Diamond property

No longer **unambiguous reversibility**, but do get **Reverse Diamond (RD)** property:

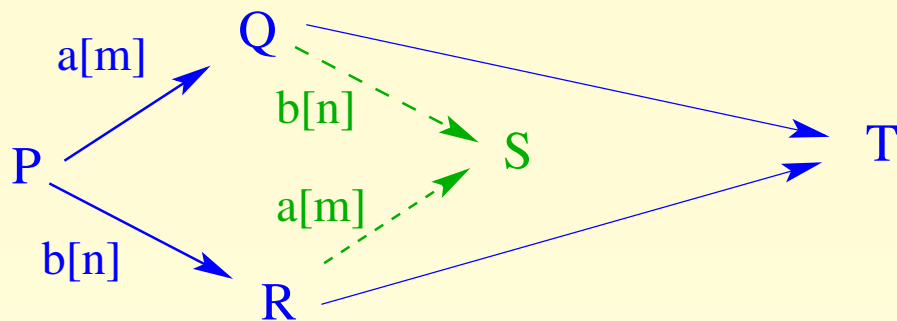


If all operators are either dynamic, static, or mixed as defined above (satisfying certain requirements), then RD holds.

Deduce from RD that the reachable processes are closed under \rightsquigarrow .

Forward Diamond property

We also have **Forward Diamond** (FD) property:



If all operators are either dynamic, static, or mixed as defined above (satisfying certain requirements), then FD holds.

Forward-reverse bisimulation

A symmetric relation \mathcal{S} on Proc is a **forward-reverse (FR) bisimulation** if whenever $\mathcal{S}(P, Q)$ then

- $p(P) \Leftrightarrow p(Q)$ for all $p \in \text{Pred}$;
- if $P \xrightarrow{\mu} P'$ then there is Q' such that $Q \xrightarrow{\mu} Q'$ and $\mathcal{S}(P', Q')$;
- if $P \xrightarrow{\sim\mu} P'$ then there is Q' such that $Q \xrightarrow{\sim\mu} Q'$ and $\mathcal{S}(P', Q')$.

We define $P \sim_{\text{FR}} Q$ iff there is an FR bisimulation \mathcal{S} such that $\mathcal{S}(P, Q)$.

Congruence Theorem: Forward-reverse bisimulation is a congruence for the reformulated operators.

Forward-reverse bisimulation is (somewhat) similar to:

- **back-and-forth bisimulation** on event structures (cf. Bednarczyk (1991); Goltz, Kuiper and Penczek (1992)).
- **(hereditary) history-preserving bisimulation** (Bednarczyk (1991); van Glabbeek and Goltz 1989).

Quite refined. For instance distinguishes

$$(a + c) \mid b + (a \mid b) + a \mid (b + c) \quad \text{and} \quad (a + c) \mid b + a \mid (b + c)$$

which are equated by **history-preserving bisimulation (Absorption Law)**.

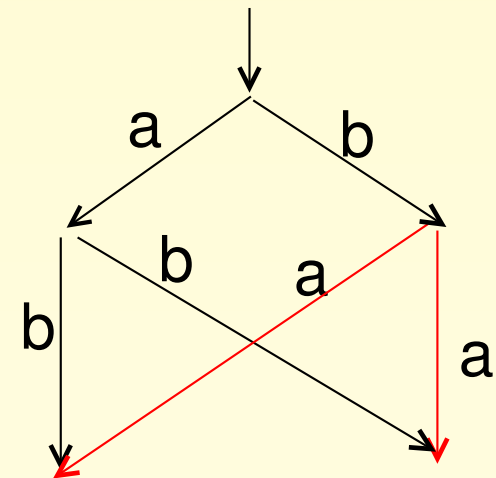
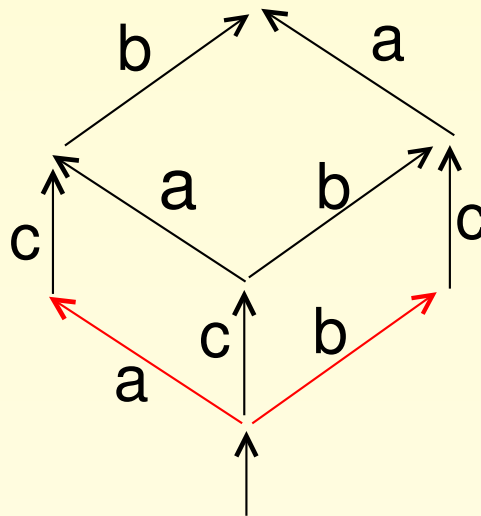
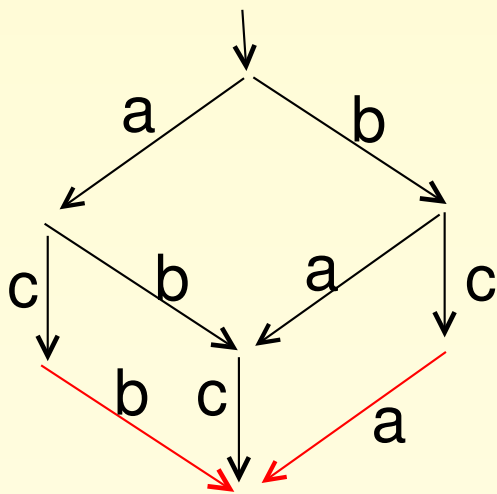
Conjecture: Forward-reverse bisimulation coincides with hereditary history-preserving bisimulation.

Prime graphs and prime event structures

A process graph is **prime** if it satisfies WF, UT, ED, RD and FD properties.

Process graphs obtained from our format are **prime**. Also, they have **no repeated labels** and have **no auto-concurrency**.

Examples of graphs that are **not** prime:



van Glabbeek and Vaandrager (1997) showed that **prime event structures** are embeddable in process graphs (in fact in prime graphs):

$$es(\text{cg}(E)) = E$$

The converse: **prime graphs** are embeddable in prime event structures:

$$\text{cg}(es(G)) = G$$

So, prime event structures **correspond** to prime graphs*.

van Glabbeek and Vaandrager (1997) showed that **prime event structures** are embeddable in process graphs (in fact in prime graphs):

$$es(\text{cg}(E)) = E$$

The converse: **prime graphs** are embeddable in prime event structures:

$$\text{cg}(es(G)) = G$$

So, prime event structures **correspond** to prime graphs*.

Bednarczyk (1991): hereditary history-preserving bisimulation coincides with back-and-forth bisimulation on prime event structures **without auto-concurrency**.

Hence, forward-reverse bisimulation **coincides** with hereditary history-preserving bisimulation.

We obtain a non-interleaving model for CCS(K), and a format of SOS rules for a non-interleaving process equivalence.

Conclusions

- Reversible computation useful for modelling biological systems
- Reversible forms of operators given by SOS rules
- Can handle most operators of CCS, CSP and ACP
- Use of predicates to disambiguate rules
- Use of keys when dealing with auto-concurrency and synchronisation
- Conservativity, reverse and forward diamond properties
- Forward-reverse bisimulation and a congruence result, correspondence with hereditary history-preserving bisimulation

Future Work

- Extend the format with simple **predicates**
- Reversible and **irreversible** actions
- Modelling of biological systems
- **Non-interleaving** semantics: check the proofs
- Logics and testing semantics