Causal Process Algebra

Steve Reeves and David Streader

{dstr,stever}@cs.waikato.ac.nz.

Department of Computer Science

University of Waikato

Hamilton, New Zealand

Shadowy formal methods

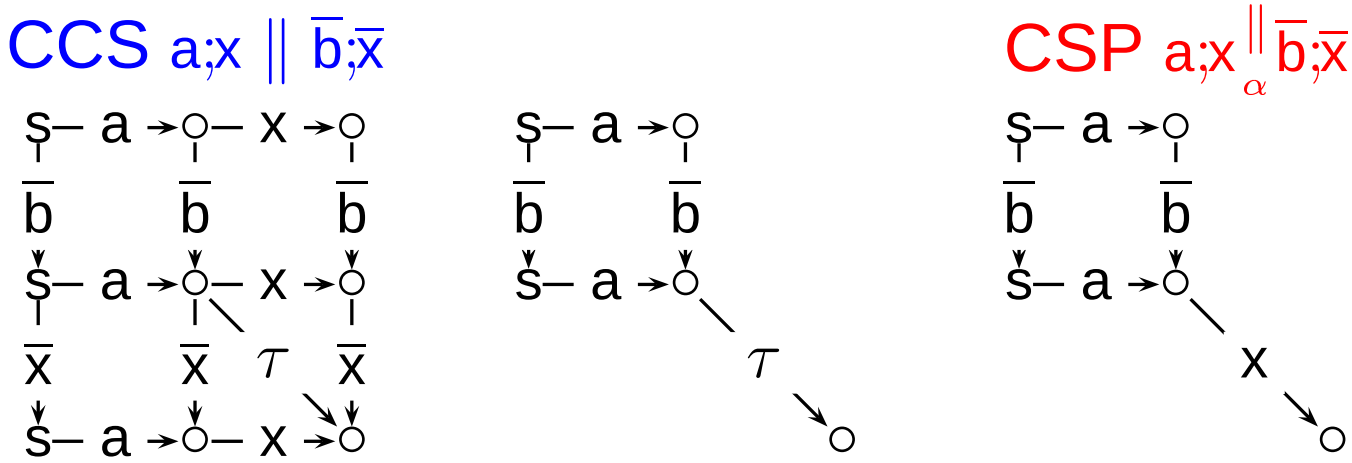
- Formal methods illuminate (by hiding)
- The better the formal method the stronger the light and the darker the shadows
- Looking in the shadows is no criticism

Shadowy formal methods

- Formal methods illuminate (by hiding)
- The better the formal method the stronger the light and the darker the shadows
- Looking in the shadows is no criticism
- process algebras communication abstracts away:
  1. one action causing another
  2. one process choosing

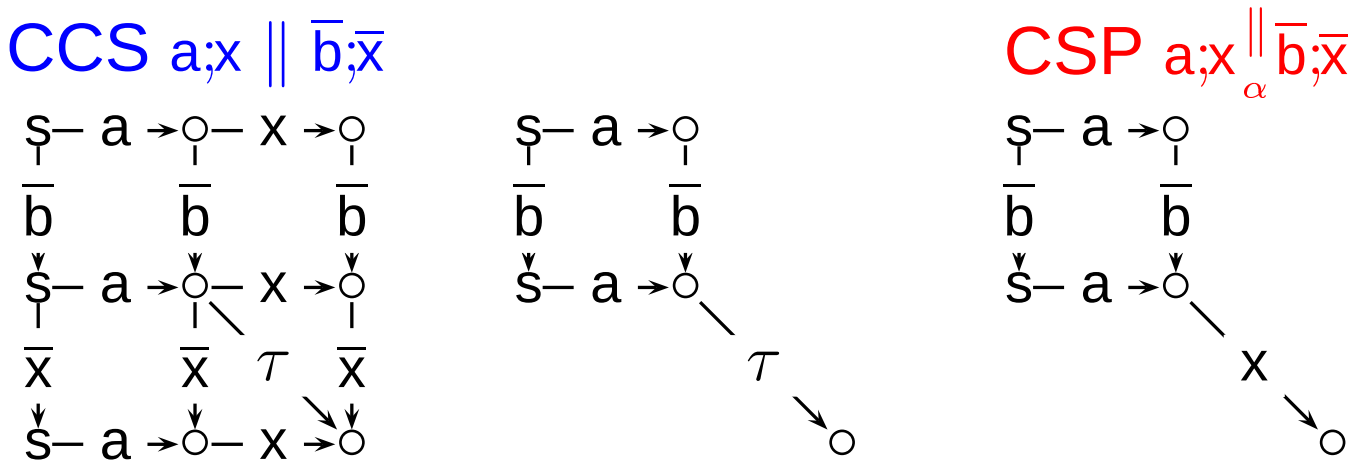
Process Model

- One set (kind) of observable actions
- communication or synchronisation:
  1. is exclusive - must occur if it can.
  2. is private - unobserved



Process Model

- One set (kind) of observable actions
- communication or synchronisation:
  1. is exclusive - must occur if it can.
  2. is private - unobserved



- CCS VM || Rob$\backslash\alpha$ called **restricted Composition** by Milner
- CSP $(\text{VM} \parallel_{\alpha} \text{Rob})/\alpha$

Causal Process Algebra

- Causal Process Algebra CPA has two sets of observable actions: the **active** or **causal** actions and the **passive** or **reactive** actions.
- + priority

Causal Process Algebra

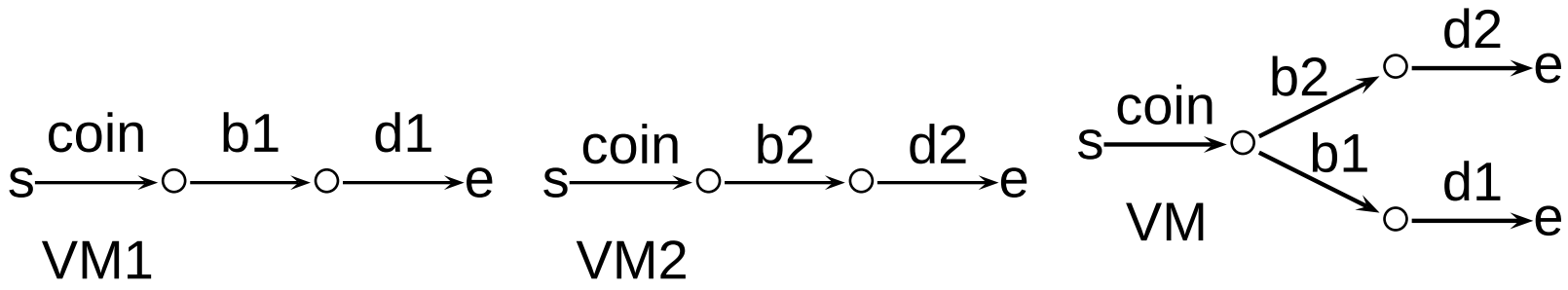
- Causal Process Algebra CPA has two sets of observable actions: the **active** or **causal** actions and the **passive** or **reactive** actions.
- + priority
- Four advantages:

Causal Process Algebra

- Causal Process Algebra CPA has two sets of observable actions: the **active** or **causal** actions and the **passive** or **reactive** actions.
- + priority
- Four advantages:
  1. extend refinement
  2. determinism by construction (Milner)
  3. small deterministic processes are implementable
  4. hiding private communication does not introduce nondeterminism

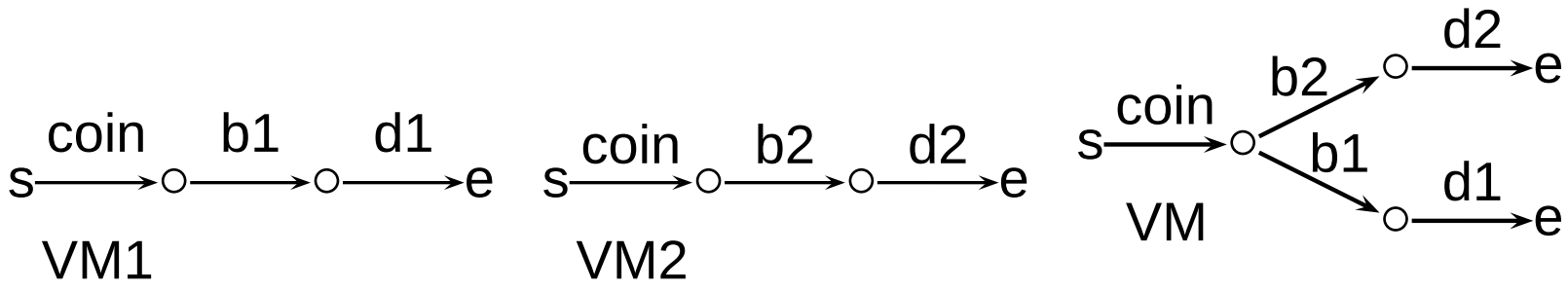
Example

- A world of Vending machines and Robots!



Example

- A world of Vending machines and Robots!



- Specification: The robot must select a button

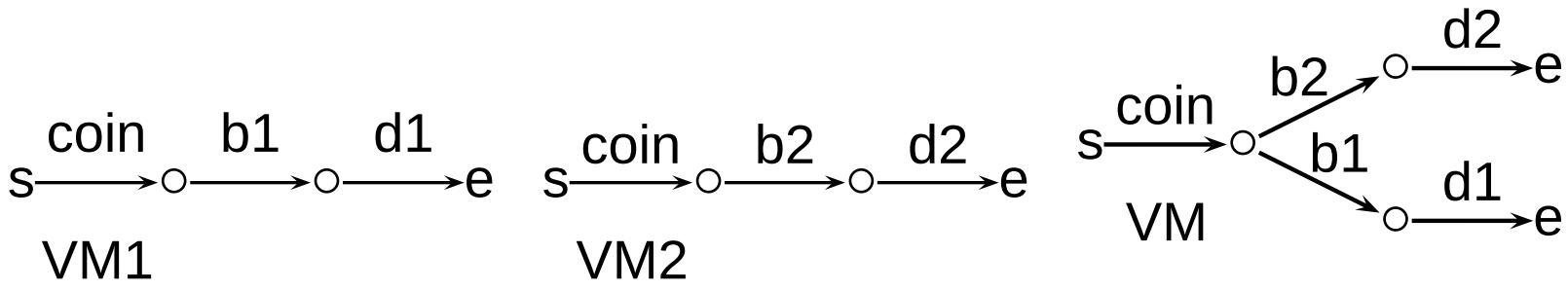
VM1 ‖ Rob will return one

VM2 ‖ Rob will return two

VM ‖ Rob will return one.

Example

- A world of Vending machines and Robots!



- Specification: The robot must select a button

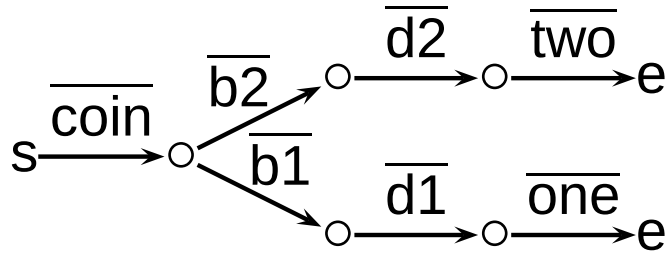
  VM1 || Rob will return one

  VM2 || Rob will return two

  VM || Rob will return one.

- Unsatisfiable in process algebra?
- Processes are at the wrong level of detail?

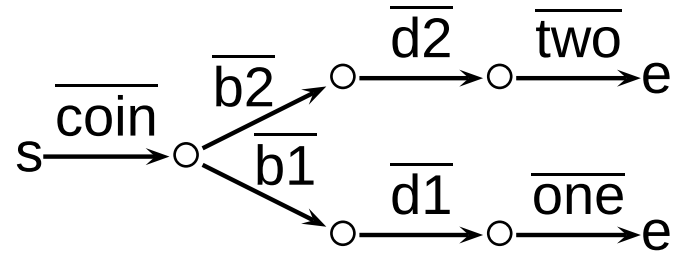
Solutions



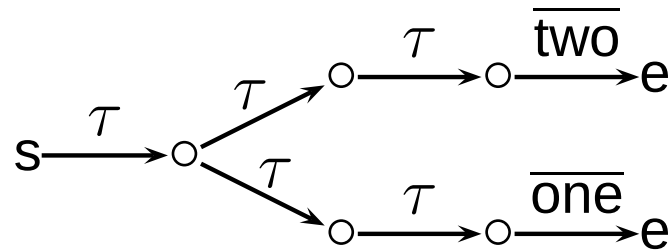
- Without Priority Rob

Solutions

- Without Priority Rob

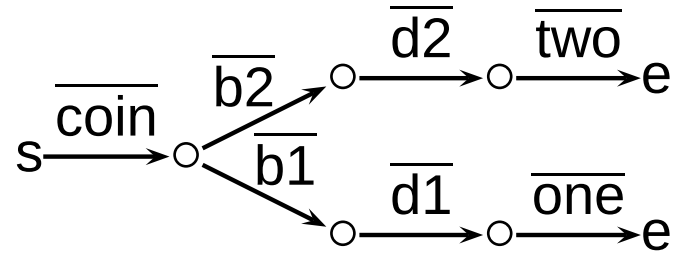


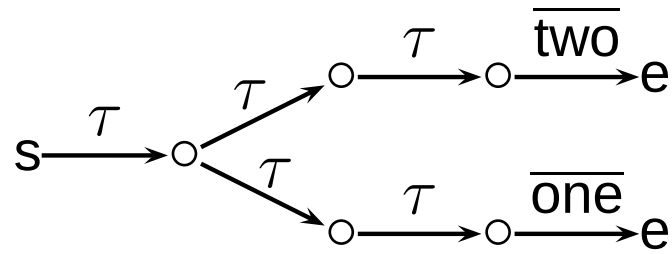
- $(\text{VM} \parallel_{\alpha} \text{Rob})/\alpha$



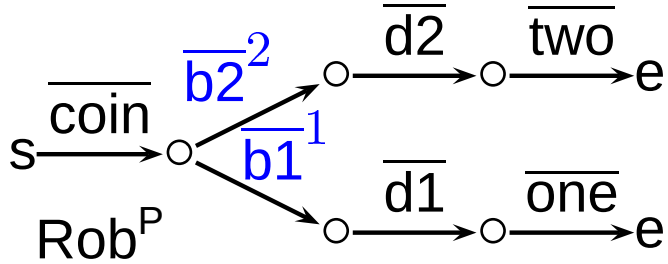
Solutions

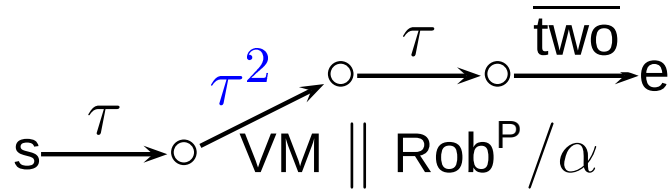
- Without Priority Rob





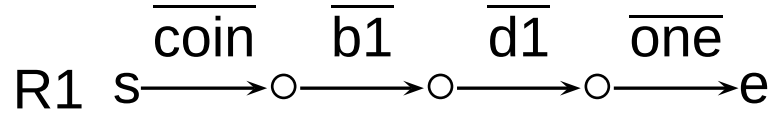
- $(\text{VM} \underset{\alpha}{\|} \text{Rob})/\alpha$
- With Priority





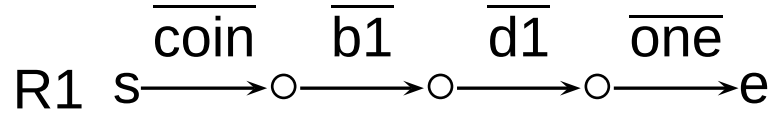
One - Extending refinement

- VM1 $\|$ R returns one



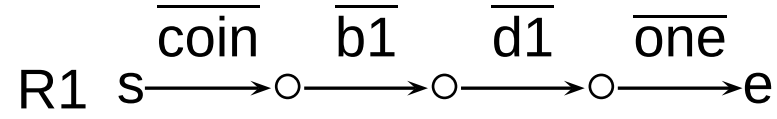
One - Extending refinement

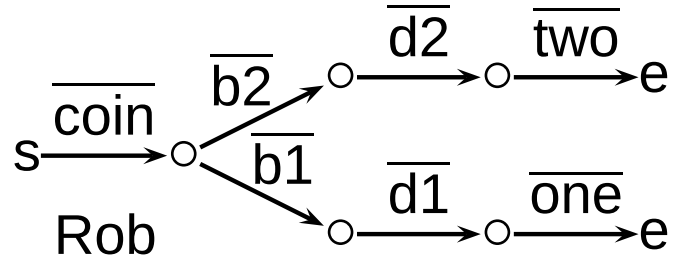
- VM1 $\|$ R returns one
- VM2 $\|$ R returns two



One - Extending refinement

- VM1 $\|$ R returns one
- VM2 $\|$ R returns two

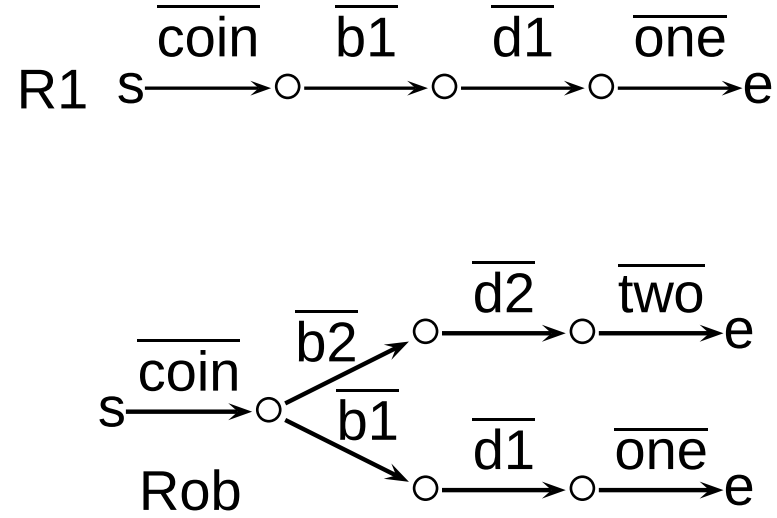




- R1 $\sqsubseteq$ Rob

One - Extending refinement

- VM1 $\|$ R returns one
- VM2 $\|$ R returns two



- R1 $\sqsubseteq$ Rob
- VM $\|$ R returns one
  requires our process model be rebuilt.

Two - Preserving Determinism

- If a deterministic robot interacts with a deterministic vending machine is the drink returned determined?

Two - Preserving Determinism

- If a deterministic robot interacts with a deterministic vending machine is the drink returned determined?
- We say yes Process algebra no!

Two - Preserving Determinism

- If a deterministic robot interacts with a deterministic vending machine is the drink returned determined?
- We say yes Process algebra no!
- We are not the first to find this problematic!
- Milner [**?**, p232] express the hope of finding “design rules, not too restrictive, which ensure that from determinate components we must arrive at a determinate system”

Two - Preserving Determinism

- If a deterministic robot interacts with a deterministic vending machine is the drink returned determined?
- We say yes Process algebra no!
- We are not the first to find this problematic!
- Milner [**?**, p232] express the hope of finding “design rules, not too restrictive, which ensure that from determinate components we must arrive at a determinate system”
- defines *confluence* [**?**, p237] “to strengthen determinacy in such a way that it will be preserved by restricted Composition”

Three - Implementable

- We assume Nondeterministic behaviour not implementable on a deterministic finite state machine

Three - Implementable

- We assume Nondeterministic behaviour not implementable on a deterministic finite state machine
- Hence CSP/CCS Rob and VM not implementable!

Three - Implementable

- We assume Nondeterministic behaviour not implementable on a deterministic finite state machine
- Hence CSP/CCS Rob and VM not implementable!
- Occam does not implement CSP/CCS style parallel composition
- Occam is consistent with CSP failure refinement:

$$(\mathsf{VM} \underset{\alpha}{\|} \mathsf{Rob})/\alpha \sqsubseteq_{F} \mathsf{VM} \|_{Occam} \mathsf{Rob}$$

- Not consistent with may and must testing refinement:

$$(\mathsf{VM} \underset{\alpha}{\|} \mathsf{Rob})/\alpha \not\sqsubseteq_{Test} \mathsf{VM} \|_{Occam} \mathsf{Rob}$$

CPA

- a set of names *Names*:
- **passive** actions $Act \stackrel{\text{def}}{=} \{\mathrm{a} \mid \mathrm{a} \in Names\}$ and
- **active** actions $\overline{Act} \stackrel{\text{def}}{=} \{\overline{\mathrm{a}} \mid \mathrm{a} \in Names\}$
- irreflexive priority relation $\angle \subseteq Pri \times Pri$
- priority function $Act_{Pri} : Names \longrightarrow Pri$
- function and relation lifted to actions or transitions.
- PLTS

$$T_{\mathrm{A}} \subseteq \{(n, (\mathrm{x}, p), m) \mid n, m \in N_{\mathrm{A}} \wedge (\mathrm{x}, p) \in Act_{Pri} \wedge \mathrm{x} \in Obs\} \cup \{(n, (\tau, p), m) \mid p \in Pri\}$$

$$n \xrightarrow{(\mathrm{x},p)} m$$

CPA

- A is p-deterministic if all transitions leaving the same node have different names, whereas
- A is deterministic if transitions with the same priority and pre-node are *passive* actions with different names.

CPA

- A is p-deterministic if all transitions leaving the same node have different names, whereas
- A is deterministic if transitions with the same priority and pre-node are *passive* actions with different names.
- A is deterministic iff $\forall_{\ddot{\mathrm{a}} \in \ddot{Act}, \overline{\mathrm{b}} \in \overline{Act} \cup \{\tau\}}$

$$x \xrightarrow{(\ddot{\mathrm{a}},p)} z \wedge x \xrightarrow{(\overline{\mathrm{b}},p)} y \Rightarrow z = y \wedge \ddot{\mathrm{a}} = \overline{\mathrm{b}}$$

$$x \xrightarrow{(\mathrm{a},p)} z \wedge x \xrightarrow{(\mathrm{a},p)} y \Rightarrow z = y$$

•

CPA

- A is p-deterministic if all transitions leaving the same node have different names, whereas
- A is deterministic if transitions with the same priority and pre-node are *passive* actions with different names.
- A is deterministic iff $\forall_{\ddot{a} \in \ddot{Act}, \bar{b} \in \overline{Act} \cup \{\tau\}}$

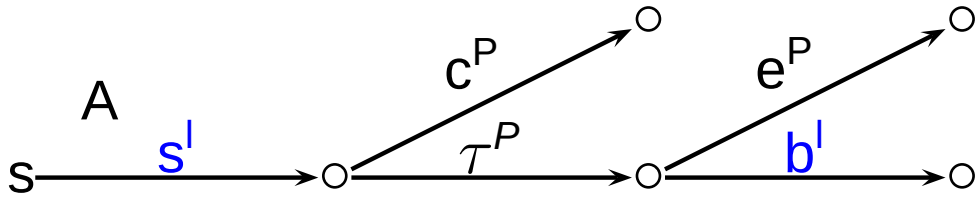
$$x \xrightarrow{(\ddot{a},p)} z \wedge x \xrightarrow{(\bar{b},p)} y \Rightarrow z = y \wedge \ddot{a} = \bar{b}$$

$$x \xrightarrow{(a,p)} z \wedge x \xrightarrow{(a,p)} y \Rightarrow z = y$$

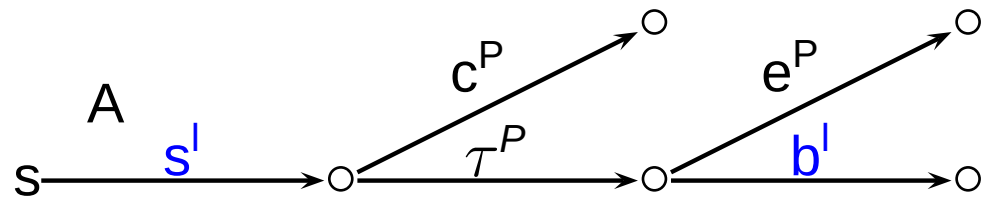
- We define $\preceq$: $Act_{Pri} \times Act_{Pri}$ by $(x, p) \preceq (y, p') \stackrel{\text{def}}{=} p \angle p'$.

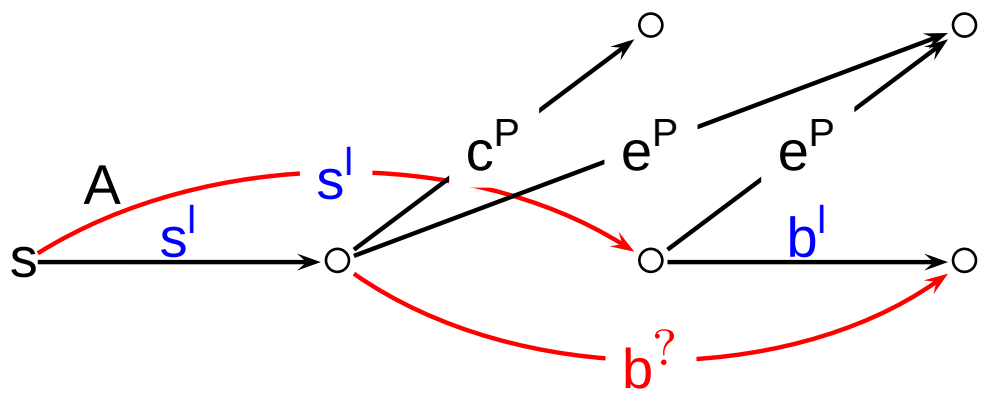
  Define $\sqsubseteq_p \subseteq Act_{Pri} \times Act_{Pri}$ by $Act_{Pri1} \sqsubseteq_p Act_{Pri2} \stackrel{\text{def}}{=} \preceq_1 \subseteq \preceq_2$.
  Lift $\sqsubseteq_p$ to processes.
  $\sqsubseteq_{tp}$ is the transitive closure of $\sqsubseteq_p \cup \sqsubseteq_t$

Observational semantics

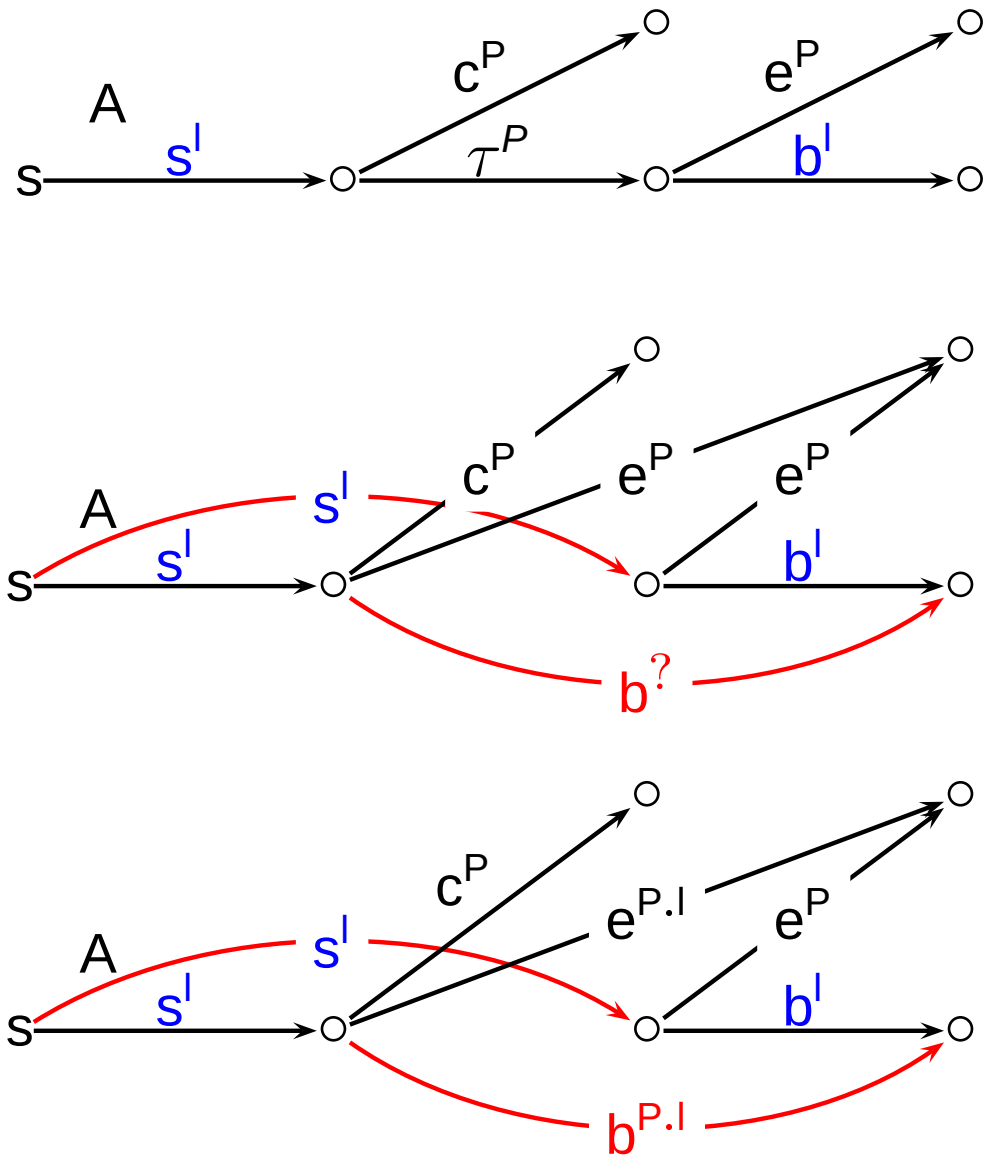


Observational semantics

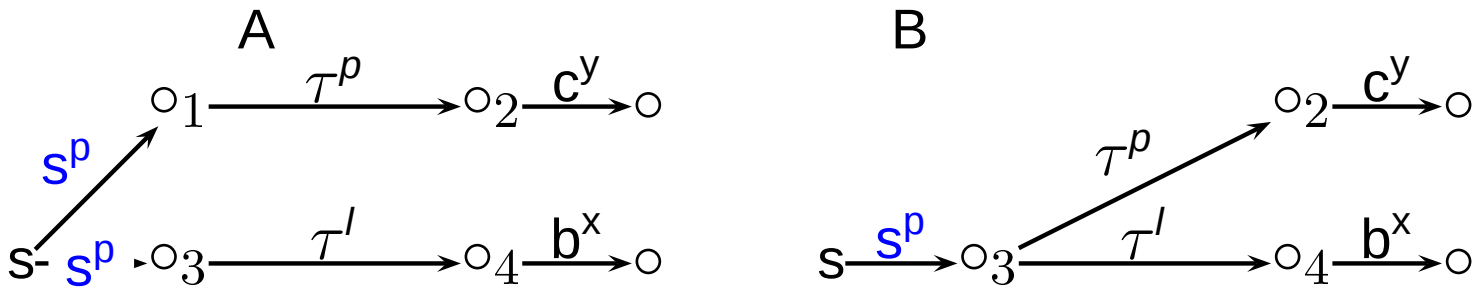




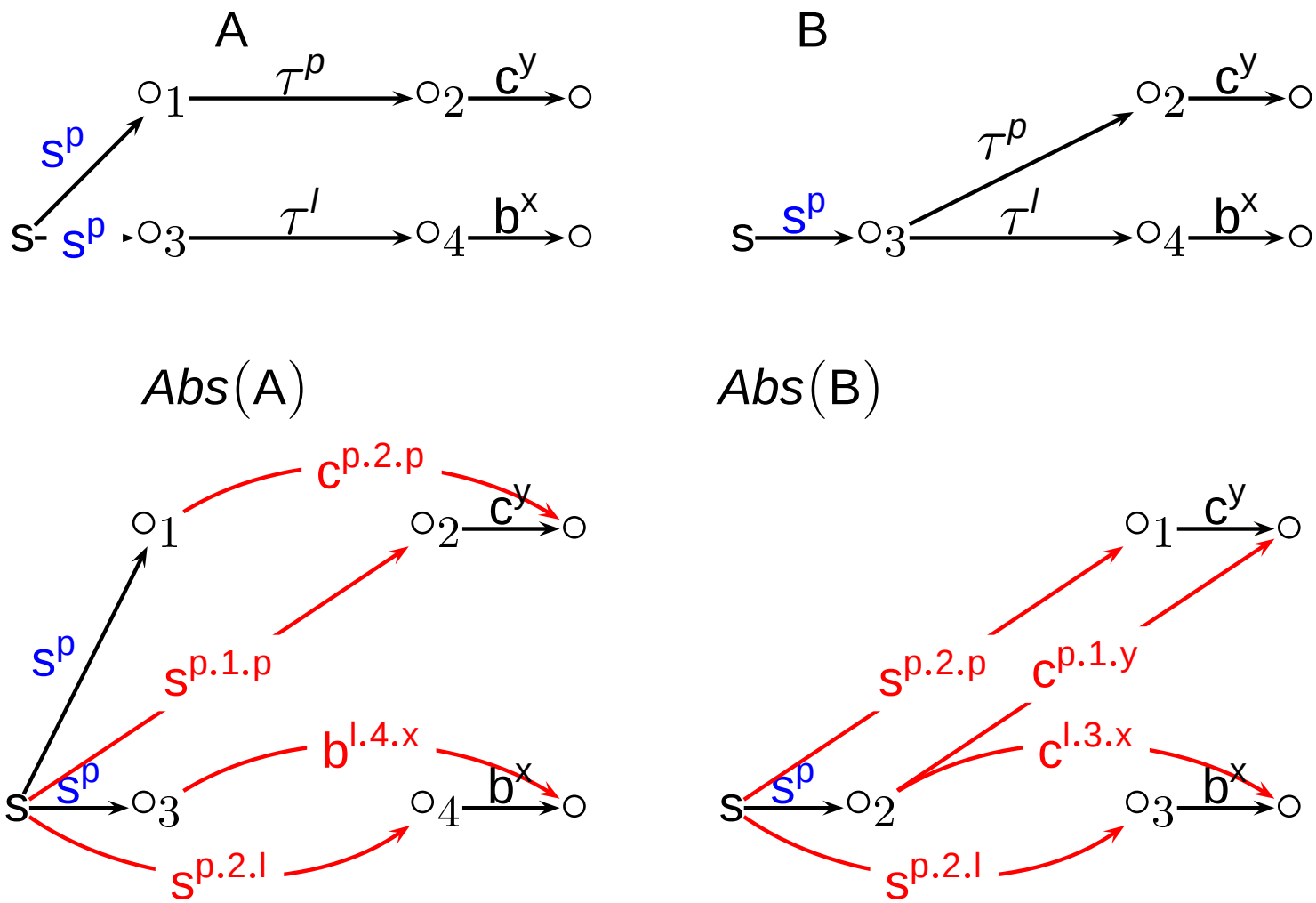
Observational semantics



Observational semantics(2)



Observational semantics(2)



End

- Causal Process Algebra CPA has two sets of observable actions: the **active** or **causal** actions and the **passive** or **reactive** actions.
- + priority
- Disadvantage more complexe semantics
- Four advantages:
  1. extend refinement
  2. determinism by construction (Milner)
  3. small deterministic processes are implementable
  4. hiding private communication does not introduce nondeterminism