

# Causal Process Algebra

*Always getting the right drink from a coffee machine.*

Steve Reeves and David Streader

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

Department of Computer Science

University of Waikato

Hamilton, New Zealand

# Overview

- We will apply the use case methodology to the specification of event based processes.

# Overview

- We will apply the use case methodology to the specification of event based processes.
- Example specification to demonstrates problems with standard refinements.

# Overview

- We will apply the use case methodology to the specification of event based processes.
- Example specification to demonstrates problems with standard refinements.
- We introduce Causal Process Algebra CPA

# Overview

- We will apply the use case methodology to the specification of event based processes.
- Example specification to demonstrates problems with standard refinements.
- We introduce Causal Process Algebra CPA
- Define a refinement from PA to CPA

# Overview

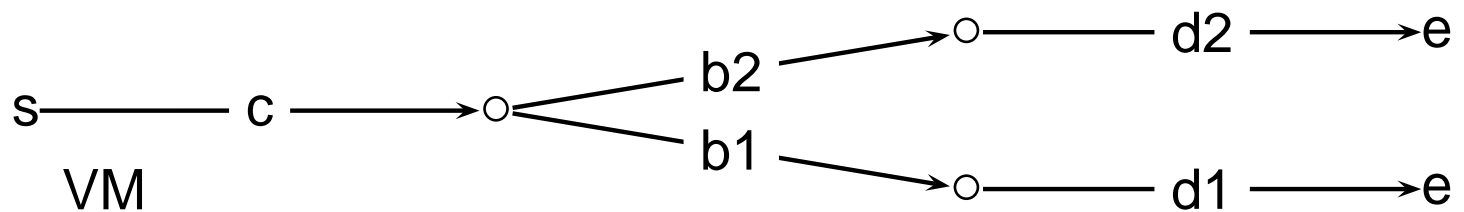
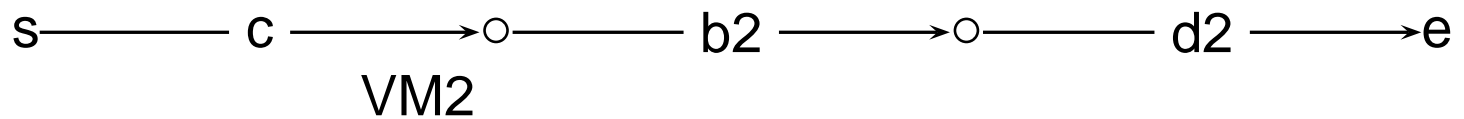
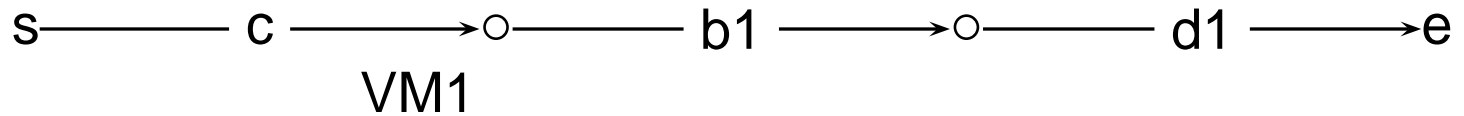
- We will apply the use case methodology to the specification of event based processes.
- Example specification to demonstrates problems with standard refinements.
- We introduce Causal Process Algebra CPA
- Define a refinement from PA to CPA
- Show how to satisfy our example specification
- Conclusion

# Overview

- We will apply the use case methodology to the specification of event based processes.
- Example specification to demonstrates problems with standard refinements.
- We introduce Causal Process Algebra CPA
- Define a refinement from PA to CPA
- Show how to satisfy our example specification
- Conclusion

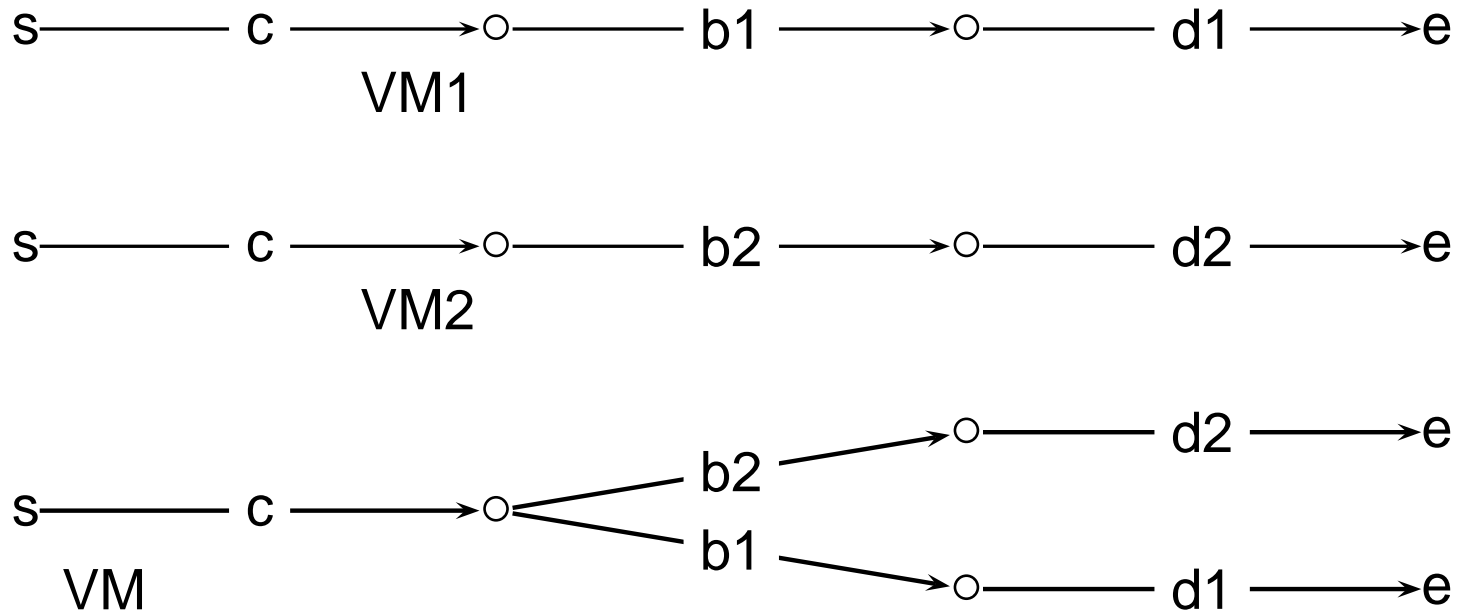
# Specification

Using handshake actions.



# Specification

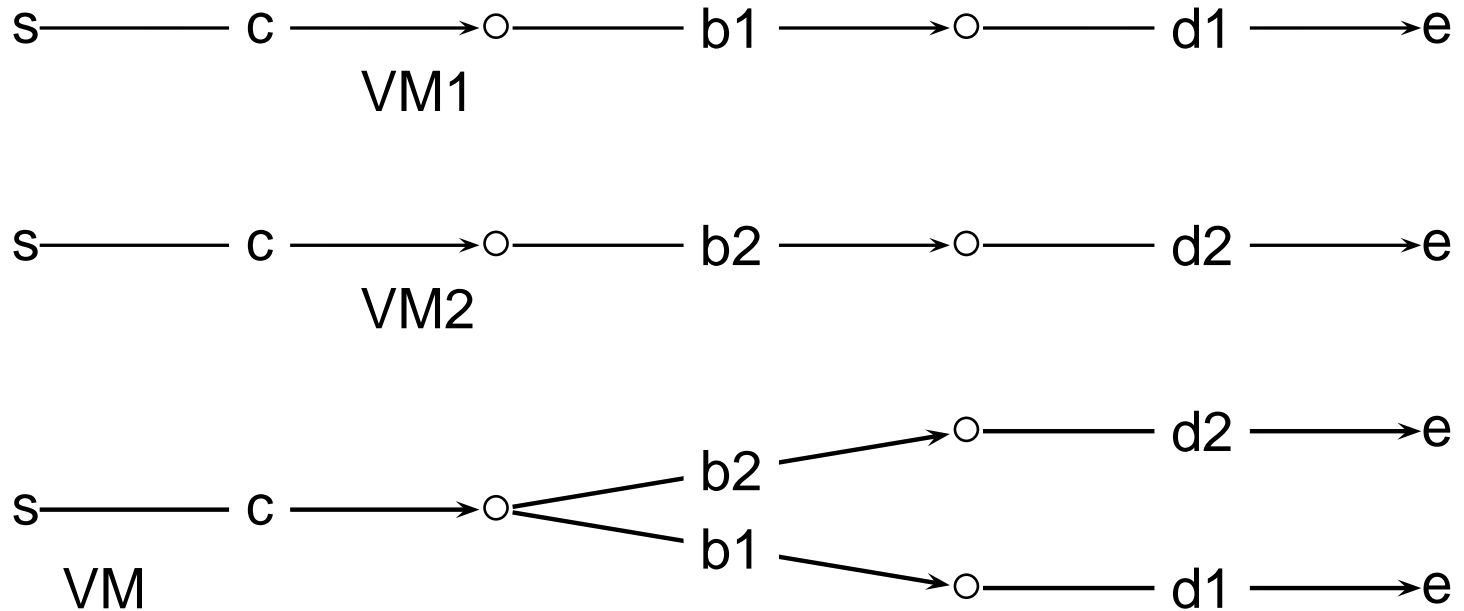
Using handshake actions.



Actions synchronize via  $b1$  and  $\overline{b1}$  become  $\tau$  just as in CCS.

# Specification

Using handshake actions.



Robot use cases:

**a** - drink  $d1$  from VM1

**b** - drink  $d2$  from VM2

**c** - drink  $d1$  from VM.

# Stepwise development

1. Use case a - drink d1 from VM1 :

$$R1 \stackrel{\text{def}}{=} \bar{c};\overline{b1};\overline{d1};r1$$

# Stepwise development

1. Use case a - drink d1 from VM1 :

$$R1 \stackrel{\text{def}}{=} \bar{c};\overline{b1};\overline{d1};r1$$

2. But **R1** fails to satisfy use case b - drink d2 from VM2.

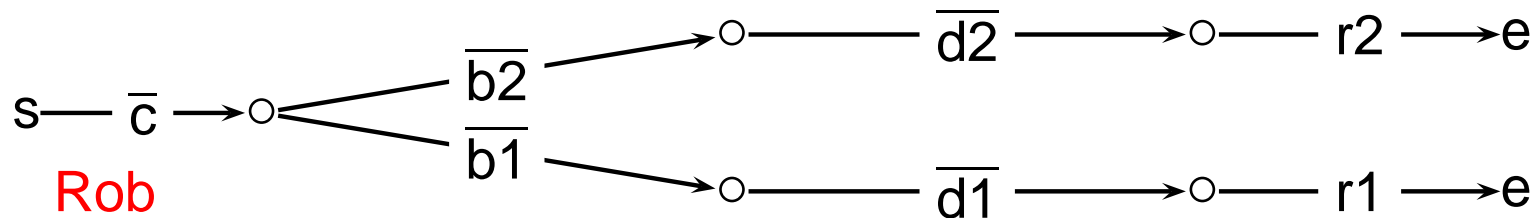
# Stepwise development

1. Use case a - drink d1 from VM1 :

$$R1 \stackrel{\text{def}}{=} \bar{c};\bar{b1};\bar{d1};r1$$

2. But **R1** fails to satisfy use case b - drink d2 from VM2.

3. **Rob** satisfies use cases a and b.



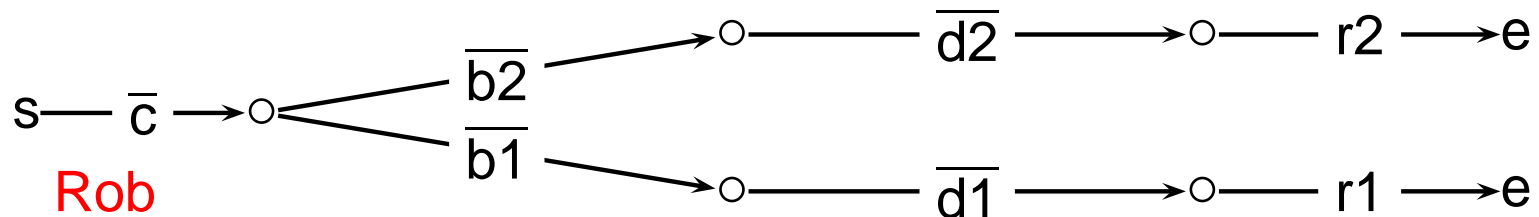
# Stepwise development

1. Use case a - drink d1 from VM1 :

$$R1 \stackrel{\text{def}}{=} \bar{c};\bar{b1};\bar{d1};r1$$

2. But **R1** fails to satisfy use case b - drink d2 from VM2.

3. **Rob** satisfies use cases a and b.



4. **R1**  $\sqsubseteq_X$  **Rob** can be established using LOTOS's extension [1], *conf* [2] or  $\sqsubseteq_{F\delta}$  "weak sub-typing" of [3]

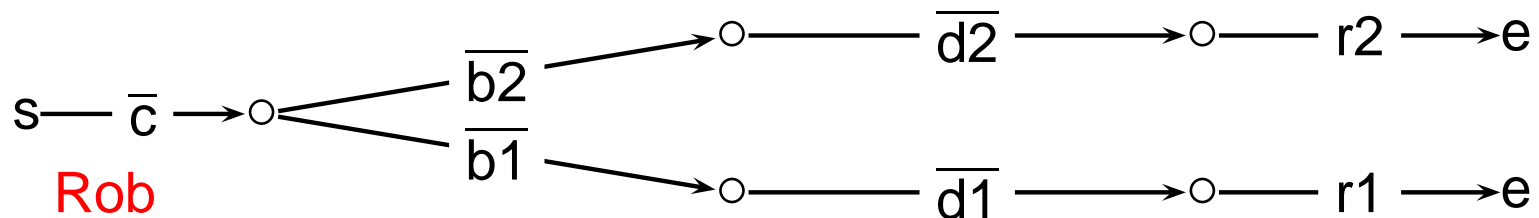
# Stepwise development

1. Use case a - drink d1 from VM1 :

$$R1 \stackrel{\text{def}}{=} \bar{c};\bar{b1};\bar{d1};r1$$

2. But **R1** fails to satisfy use case b - drink d2 from VM2.

3. **Rob** satisfies use cases a and b.



4. **R1**  $\sqsubseteq_X$  **Rob** can be established using LOTOS's extension [1], *conf* [2] or  $\sqsubseteq_{F\delta}$  "weak sub-typing" of [3]
5. At this point actions are viewed as being defined at an adequate level of abstraction.

# Problem

1. We are unable to satisfy the third use case c.

# Problem

1. We are unable to satisfy the third use case c.
2. From this we conclude that the actions we have chosen are not at an adequate level of abstraction.

# Problem

1. We are unable to satisfy the third use case *c*.
2. From this we conclude that the actions we have chosen are not at an adequate level of abstraction.
3. Using CSP, CCS, ACP . . . we would rewrite specification using different actions.

# Problem

1. We are unable to satisfy the third use case c.
2. From this we conclude that the actions we have chosen are not at an adequate level of abstraction.
3. Using CSP,CCS,ACP . . . we would rewrite specification using different actions.
4. For large processes, changing the formal specification could entail a huge amount of work.

# Problem

1. We are unable to satisfy the third use case  $c$ .
2. From this we conclude that the actions we have chosen are not at an adequate level of abstraction.
3. Using CSP, CCS, ACP . . . we would rewrite specification using different actions.
4. For large processes, changing the formal specification could entail a huge amount of work.
5. The same problem also occurs with feature addition.

# CPA

1. The operational semantics of CPA is given by PLTS

# CPA

1. The operational semantics of CPA is given by PLTS
2. Split actions into two sets **Active** -  $\bar{a}$  and **Passive**-  $a$

# CPA

1. The operational semantics of CPA is given by PLTS
2. Split actions into two sets **Active** -  $\bar{a}$  and **Passive**-  $a$
3. Set of priorities  $Pri$  with an irreflexive priority relation  $\angle \subseteq Pri \times Pri$  and a priority function  $Act_{Pri} : Obs \rightarrow Pri$  such that  $Act_{Pri}(\bar{a}) = Act_{Pri}(a)$ .

# CPA

1. The operational semantics of CPA is given by PLTS
2. Split actions into two sets **Active** -  $\bar{a}$  and **Passive**-  $a$
3. Set of priorities  $Pri$  with an irreflexive priority relation  $\angle \subseteq Pri \times Pri$  and a priority function  $Act_{Pri} : Obs \rightarrow Pri$  such that  $Act_{Pri}(\bar{a}) = Act_{Pri}(a)$ .
4. The transitions of process  $A$  are  $T_A$   
We write  $n \xrightarrow{a^p} m$  or  $n \xrightarrow{(a,p)} m$  iff  $(n, (a, p), m) \in T_A$

# CPA

1. The operational semantics of CPA is given by PLTS
2. Split actions into two sets **Active** -  $\bar{a}$  and **Passive**-  $a$
3. Set of priorities  $Pri$  with an irreflexive priority relation  $\angle \subseteq Pri \times Pri$  and a priority function  $Act_{Pri} : Obs \rightarrow Pri$  such that  $Act_{Pri}(\bar{a}) = Act_{Pri}(a)$ .

4. The transitions of process  $A$  are  $T_A$

We write  $n \xrightarrow{a^p} m$  or  $n \xrightarrow{(a,p)} m$  iff  $(n, (a, p), m) \in T_A$

5. Well behaved PLTS respect  $Act_{Pri}$

$$n \xrightarrow{(x,p)}_A m \Rightarrow (x, p) \in Act_{Pri}$$

# CPA

1. Transitions of a PLTS  $n \xrightarrow{(x,p)} m$  could be transitions of a LTS  $n \xrightarrow{x} m$  or  $n \xrightarrow{"(x,p)"} m$ .

# CPA

1. Transitions of a PLTS  $n \xrightarrow{(x,p)} m$  could be transitions of a LTS  $n \xrightarrow{x} m$  or  $n \xrightarrow{“(x,p)”} m$ .
2. For well behaved PLTS they give isomorphic semantics.  
([2, 4])

# CPA

1. Transitions of a PLTS  $n \xrightarrow{(x,\rho)} m$  could be transitions of a LTS  $n \xrightarrow{x} m$  or  $n \xrightarrow{“(x,\rho)”} m$ .
2. For well behaved PLTS they give isomorphic semantics.  
([2, 4])

$$\begin{aligned} \text{Ref}(\rho, C) &\stackrel{\text{def}}{=} \{X \mid s_C \xrightarrow{\rho} s \wedge X \subseteq (\text{Actions} \times \text{Pri}) - \pi(s)\} \\ (A \sqsubseteq_C C) &\Leftrightarrow \forall \rho \in \text{Tr}(A). \text{Ref}(\rho, C) \subseteq \text{Ref}(\rho, A). \\ (A \sqsubseteq_F C) &\Leftrightarrow \forall \rho. \text{Ref}(\rho, C) \subseteq \text{Ref}(\rho, A). \end{aligned}$$

# Priority refinement

- On well behaved PLTS

# Priority refinement

- On well behaved PLTS

- $(x, p) \preceq (y, q) \Leftrightarrow p \angle q.$

# Priority refinement

- On well behaved PLTS

- $(x, p) \preceq (y, q) \Leftrightarrow p \angle q.$

- Priority relation  $\sqsubseteq_p \subseteq Act_{Pri} \times Act_{Pri}$  is defined by

$$Act_{Pri1} \sqsubseteq_p Act_{Pri2} \Leftrightarrow \preceq_1 \subseteq \preceq_2$$

# Priority refinement

- On well behaved PLTS

- $(x, p) \preceq (y, q) \Leftrightarrow p \angle q.$

- Priority relation  $\sqsubseteq_p \subseteq Act_{Pri} \times Act_{Pri}$  is defined by

$$Act_{Pri1} \sqsubseteq_p Act_{Pri2} \Leftrightarrow \preceq_1 \subseteq \preceq_2$$

- Refinement  $\sqsubseteq_{tp}$  is the transitive closure of  $\sqsubseteq_p \cup \sqsubseteq_t$

# Vertical refinement

- On well behaved PLTS

# Vertical refinement

- On well behaved PLTS
- Two mappings  $\llbracket - \rrbracket : LTS \rightarrow PLTS$  and  $vA : PLTS \rightarrow LTS$

# Vertical refinement

- On well behaved PLTS
- Two mappings  $\llbracket - \rrbracket : LTS \rightarrow PLTS$  and  $v_A : PLTS \rightarrow LTS$
- $\llbracket - \rrbracket$  adds same priority  $p_c$  to all transitions.

$$\llbracket T_A \rrbracket \stackrel{\text{def}}{=} \{ n \xrightarrow{(x, p_c)} m \mid n \xrightarrow{x} _A m \}$$

# Vertical refinement

- On well behaved PLTS
- Two mappings  $\llbracket - \rrbracket : LTS \rightarrow PLTS$  and  $vA : PLTS \rightarrow LTS$
- $\llbracket - \rrbracket$  adds same priority  $p_c$  to all transitions.

$$\llbracket T_A \rrbracket \stackrel{\text{def}}{=} \{n \xrightarrow{(x, p_c)} m \mid n \xrightarrow{x} _A m\}$$

- $vA$  forgets the priority.

$$vA(T_P) \stackrel{\text{def}}{=} \{n \xrightarrow{x} m \mid n \xrightarrow{(x, p)} _P m\}$$

# Stepwise refinement

●  $R1 \sqsubseteq_C Rob$

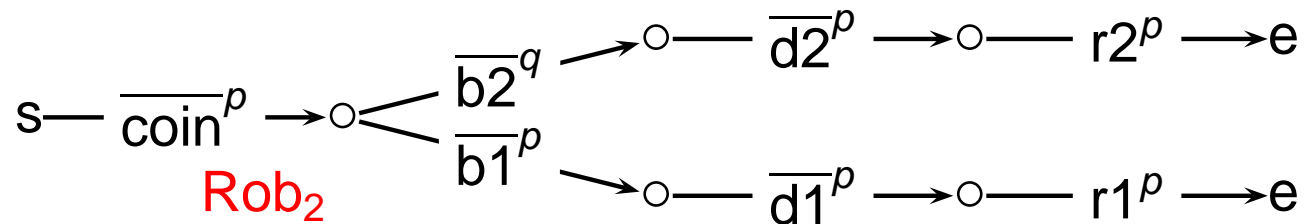
# Stepwise refinement

●  $R1 \sqsubseteq_C Rob$

●  $R1 \sqsubseteq_C Rob \sqsubseteq_{vR} \llbracket Rob \rrbracket$

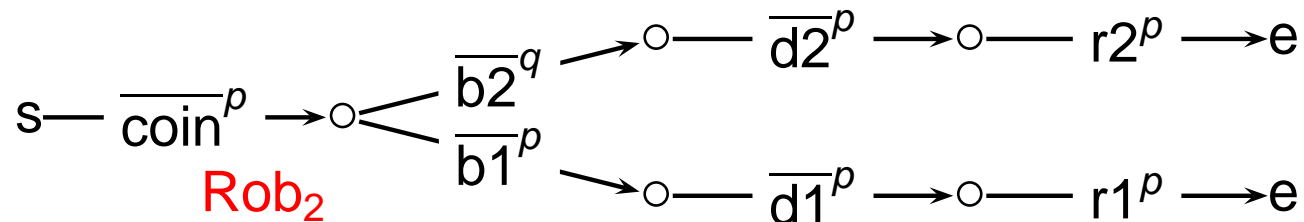
# Stepwise refinement

- $R1 \sqsubseteq_C Rob$
- $R1 \sqsubseteq_C Rob \sqsubseteq_{vR} \llbracket Rob \rrbracket$
- $R1 \sqsubseteq_C Rob \sqsubseteq_{vR} \llbracket Rob_H \rrbracket \sqsubseteq_p Rob_2$



# Stepwise refinement

- $R1 \sqsubseteq_C \text{Rob}$
- $R1 \sqsubseteq_C \text{Rob} \sqsubseteq_{vR} \llbracket \text{Rob} \rrbracket$
- $R1 \sqsubseteq_C \text{Rob} \sqsubseteq_{vR} \llbracket \text{Rob}_H \rrbracket \sqsubseteq_p \text{Rob}_2$



- if  $q \angle p$   $\text{Rob}_2$  is a satisfies our specification.

# Pruning

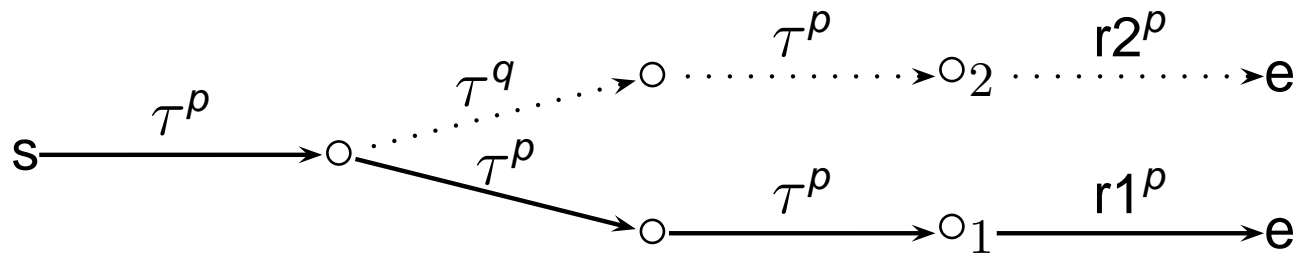
$n \xrightarrow{(x,q)} m$  is priority unreachable (can never be executed)

iff  $q \angle p \wedge (n \xrightarrow{(x,p)} \vee n \xrightarrow{(\tau,p)})$ .

# Pruning

$n \xrightarrow{(x,q)} m$  is priority unreachable (can never be executed)

iff  $q \angle p \wedge (n \xrightarrow{(x,p)} \vee n \xrightarrow{(\tau,p)})$ .

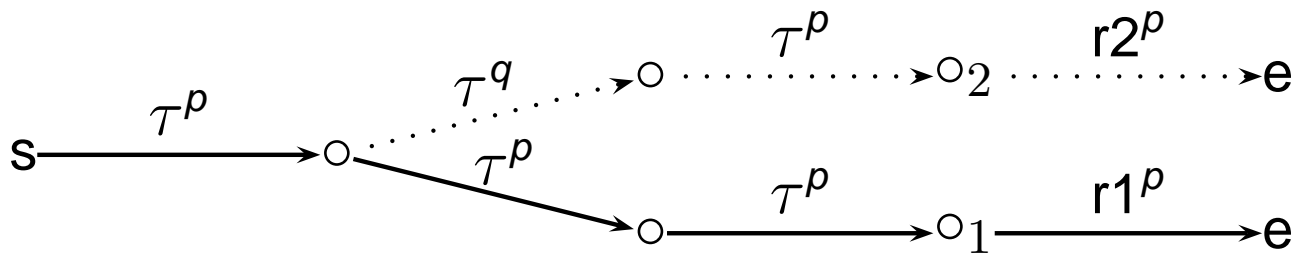


$\llbracket \text{VM} \rrbracket \parallel_{\text{Act} \cup \overline{\text{Act}}} \text{Rob}_2$

# Pruning

$n \xrightarrow{(x,q)} m$  is priority unreachable (can never be executed)

iff  $q \angle p \wedge (n \xrightarrow{(x,p)} \vee n \xrightarrow{(\tau,p)})$ .



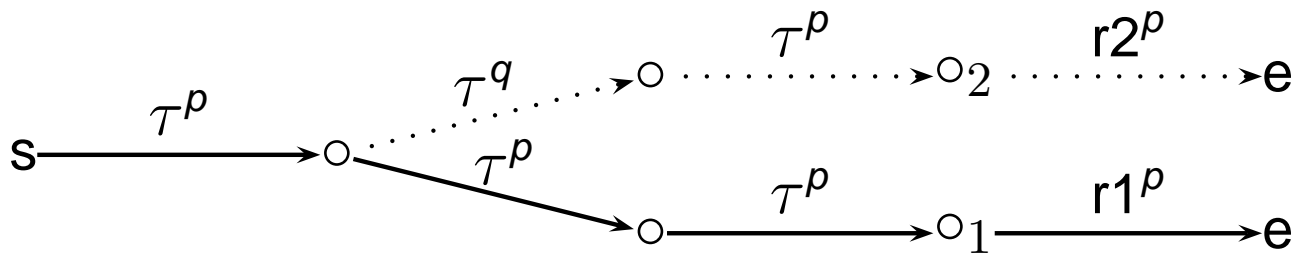
$\llbracket \text{VM} \rrbracket \parallel_{\text{Act} \cup \overline{\text{Act}}} \text{Rob}_2$

$\text{VM} \parallel_{\text{Act} \cup \overline{\text{Act}}} \text{Rob} \sqsubseteq_{oF} \nu A(\text{Prune}(\text{VM} \parallel_{\text{Act} \cup \overline{\text{Act}}} \text{Rob}_2))$

# Pruning

$n \xrightarrow{(x,q)} m$  is priority unreachable (can never be executed)

iff  $q \angle p \wedge (n \xrightarrow{(x,p)} \vee n \xrightarrow{(\tau,p)})$ .



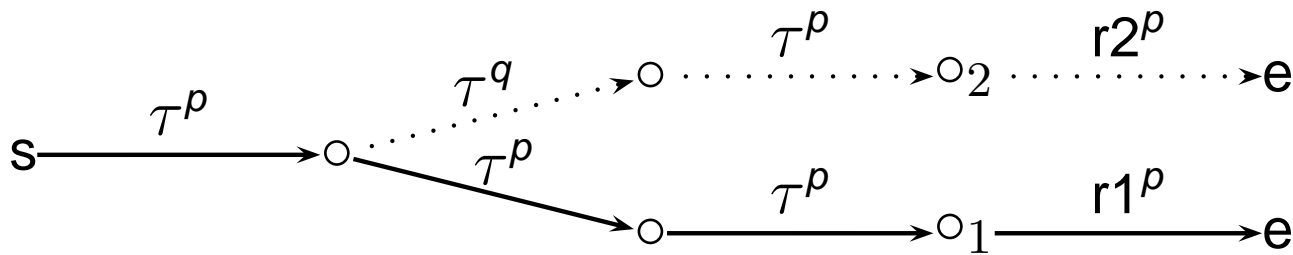
$\llbracket \text{VM} \rrbracket \parallel_{\text{Act} \cup \overline{\text{Act}}} \text{Rob}_2$

$\text{VM} \parallel_{\text{Act} \cup \overline{\text{Act}}} \text{Rob} \sqsubseteq_{oF} \nu A(\text{Prune}(\text{VM} \parallel_{\text{Act} \cup \overline{\text{Act}}} \text{Rob}_2))$

# Pruning

$n \xrightarrow{(x,q)} m$  is priority unreachable (can never be executed)

iff  $q \angle p \wedge (n \xrightarrow{(x,p)} \vee n \xrightarrow{(\tau,p)})$ .



$\llbracket \text{VM} \rrbracket \parallel_{\text{Act} \cup \overline{\text{Act}}} \text{Rob}_2$

$\text{VM} \parallel_{\text{Act} \cup \overline{\text{Act}}} \text{Rob} \sqsubseteq_{oF} \nu A(\text{Prune}(\text{VM} \parallel_{\text{Act} \cup \overline{\text{Act}}} \text{Rob}_2))$

# Conclusion

- We have shown how to apply use case specification with event based processes

# Conclusion

- We have shown how to apply use case specification with event based processes
- We have define causal process algebra CPA.

# Conclusion

- We have shown how to apply use case specification with event based processes
- We have define causal process algebra CPA.
- We have defined an observational semantics for CPA.
- We have developed a solution to our example specification by stepwise refinement.

# Conclusion

- We have shown how to apply use case specification with event based processes
- We have define causal process algebra CPA.
- We have defined an observational semantics for CPA.
- We have developed a solution to our example specification by stepwise refinement.

# Conclusion

- We have shown how to apply use case specification with event based processes
- We have define causal process algebra CPA.
- We have defined an observational semantics for CPA.
- We have developed a solution to our example specification by stepwise refinement.

# References

- [1] Brinksma, E., Scollo, G.: Formal notions of implementation and conformance in LOTOS. Technical Report INF-86-13, Twente University of Technology, Department of Informatics, Enschede, The Netherlands (1986)
- [2] Brinksma, E., Scollo, G., Steenbergen, C.: LOTOS specifications, their implementation and their tests. Protocol Specification, Testing and Verification **VI** (1986) 349–360
- [3] Fischer, C., Wehrheim, H.: Behavioural subtyping relations for object-oriented formalisms. LNCS **1816** (2000) 469–483
- [4] Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall International Series in Computer Science (1985)