Pomsets with boxes

Protection, Separation, and Locality in Concurrent Kleene Algebra

FSCD 2020

1-4 July 2020

Paul Brunet & David Pym

University College London

UCL

Concurrent Kleene Algebra

an algebraic framework to reason about concurrent programs

☞ Programs: $0 \mid 1 \mid a \in A \mid e \cdot f \mid e \parallel f \mid e + f \mid e^\star$

Concurrent Kleene Algebra

an algebraic framework to reason about concurrent programs

☞ Programs: $0 \mid 1 \mid a \in A \mid e \cdot f \mid e \parallel f \mid e + f \mid e^*$

abort execution

Concurrent Kleene Algebra

an algebraic framework to reason about concurrent programs

skip

☞ Programs: $0 \mid 1 \mid a \in A \mid e \cdot f \mid e \parallel f \mid e + f \mid e^\star$

abort execution

Concurrent Kleene Algebra

an algebraic framework to reason about concurrent programs

skip

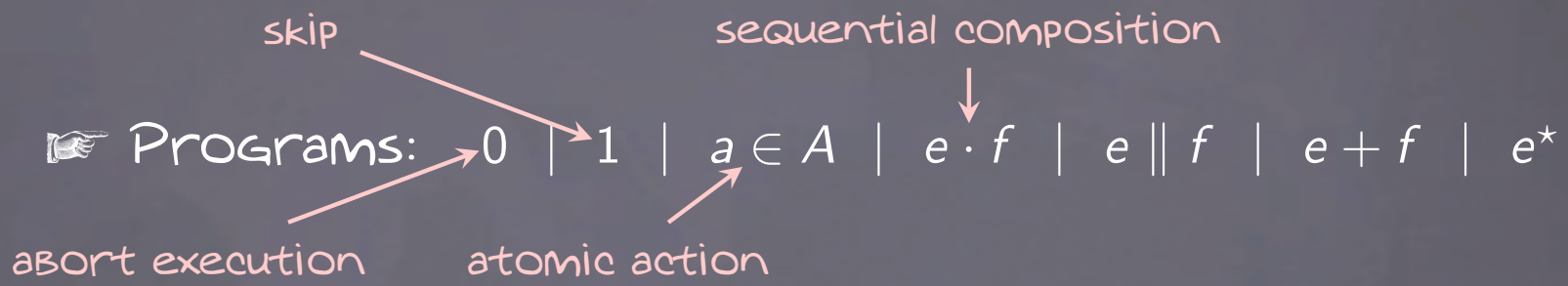
☞ Programs: $0 \mid 1 \mid a \in A \mid e \cdot f \mid e \parallel f \mid e + f \mid e^*$

abort execution

atomic action

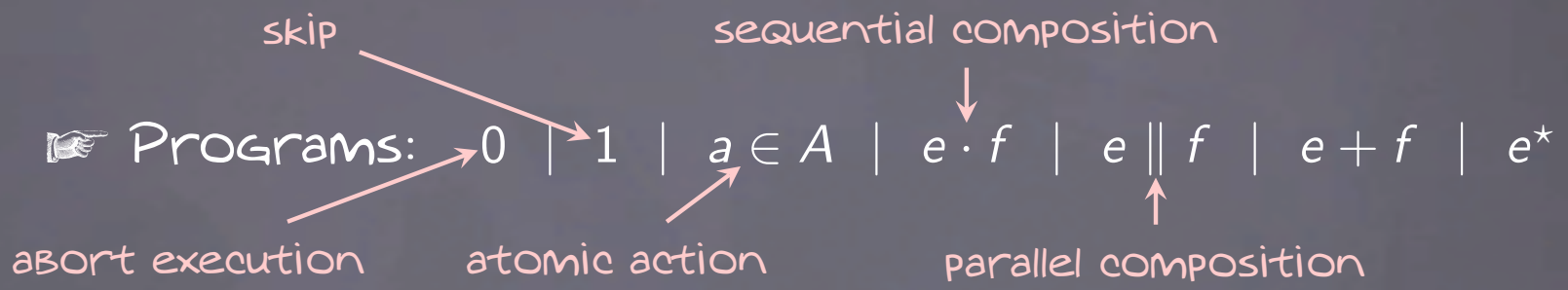
Concurrent Kleene Algebra

an algebraic framework to reason about concurrent programs



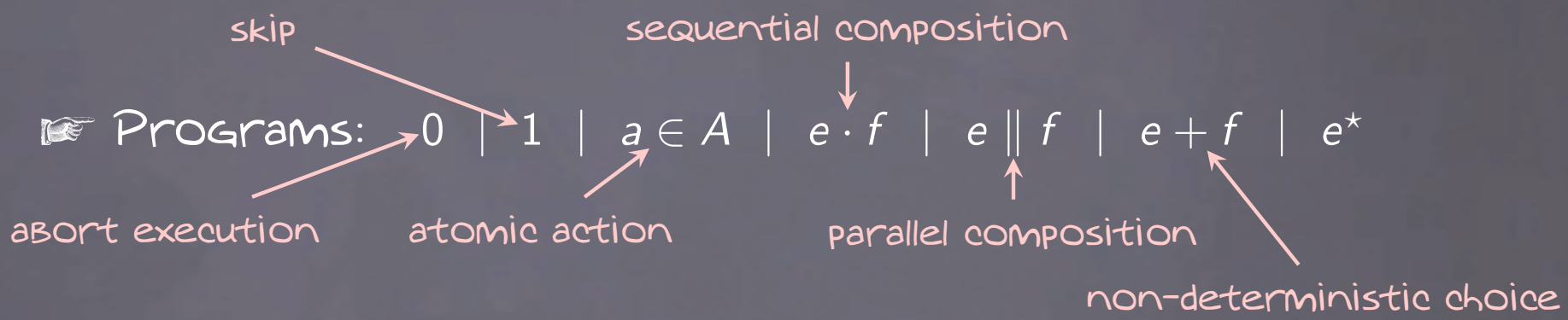
Concurrent Kleene Algebra

an algebraic framework to reason about concurrent programs



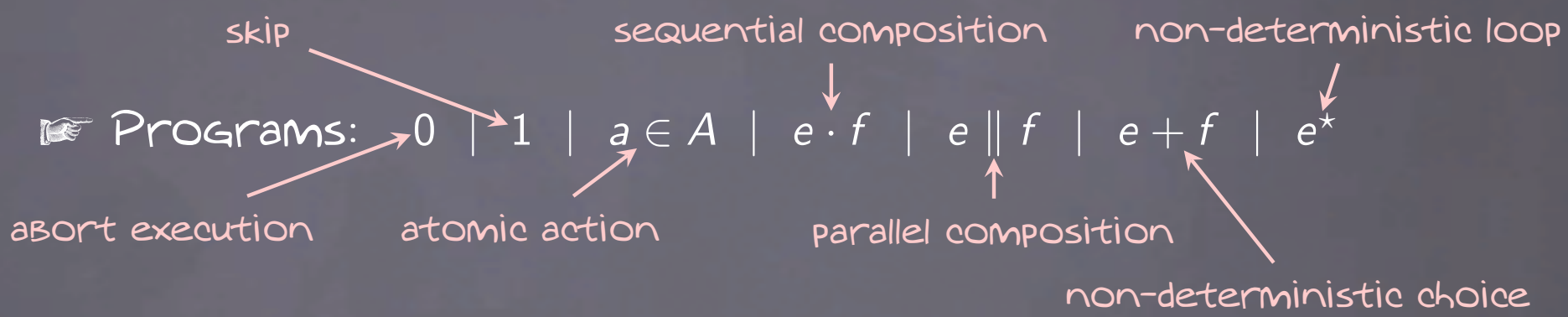
Concurrent Kleene Algebra

an algebraic framework to reason about concurrent programs



Concurrent Kleene Algebra

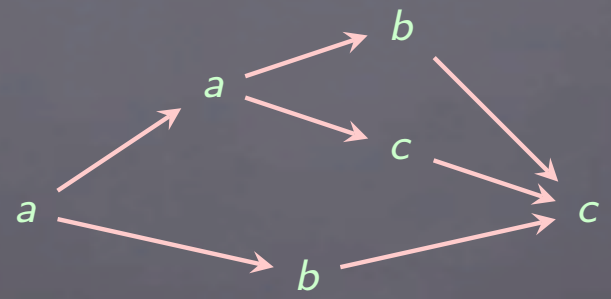
an algebraic framework to reason about concurrent programs



Concurrent Kleene Algebra

an algebraic framework to reason about concurrent programs

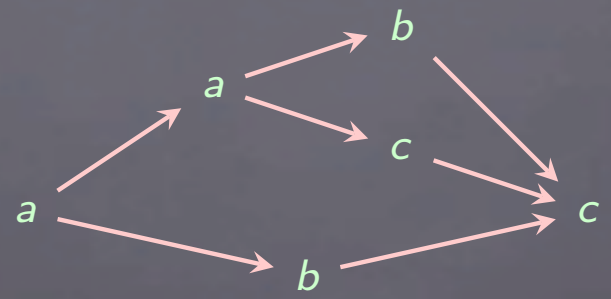
- Programs: $0 \mid 1 \mid a \in A \mid e \cdot f \mid e \parallel f \mid e + f \mid e^*$
- Semantics: sets of pomsets, i.e. partial traces



Concurrent Kleene Algebra

an algebraic framework to reason about concurrent programs

- Programs: $0 \mid 1 \mid a \in A \mid e \cdot f \mid e \parallel f \mid e + f \mid e^*$
- Semantics: sets of pomsets, i.e. partial traces



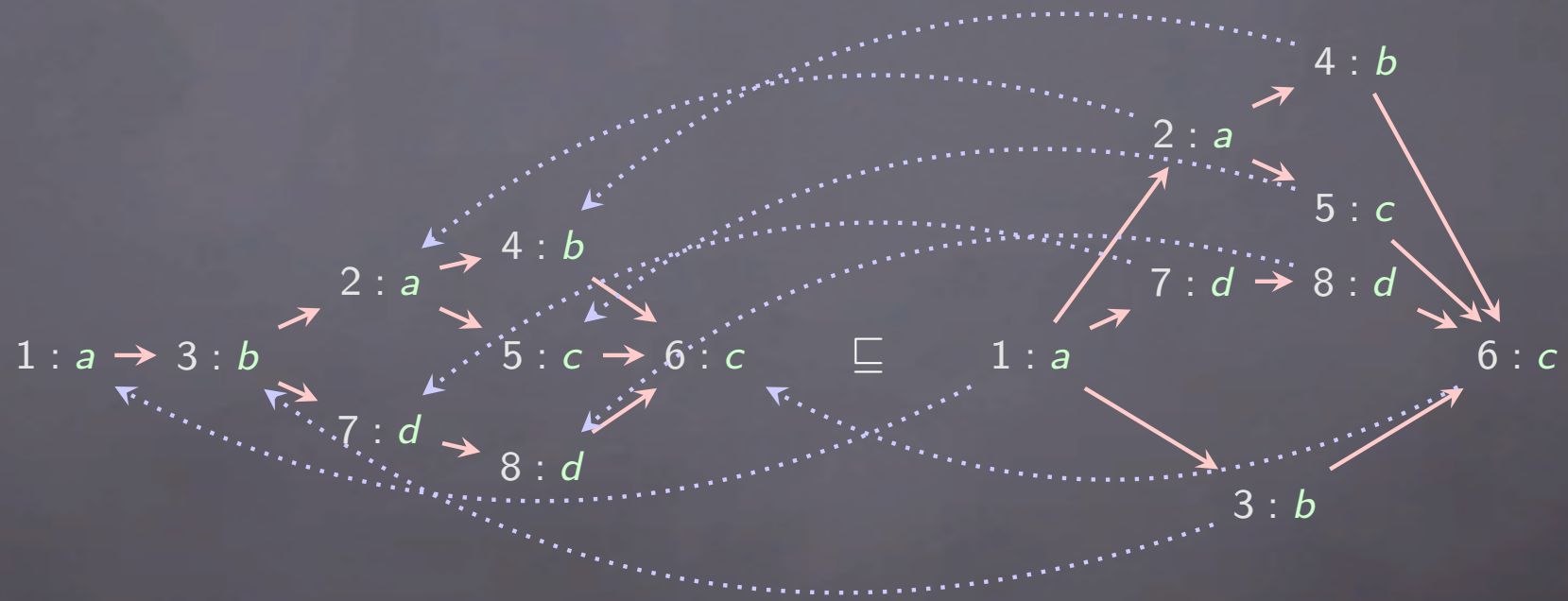
- Specifications: $[\![e]\!] \leq [\![f]\!]$.

  "All behaviours of the program e satisfy the property f."

Interleavings

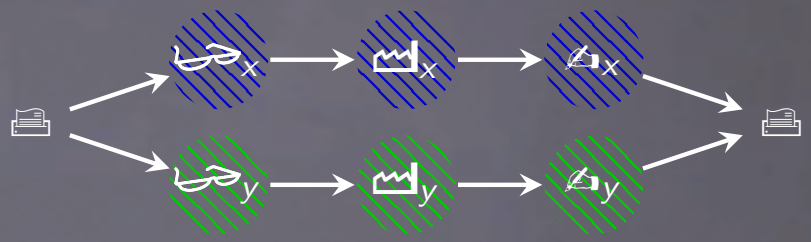
Interchange law

$$(a \parallel b) \cdot (c \parallel d) \leq (a \cdot c) \parallel (b \cdot d).$$



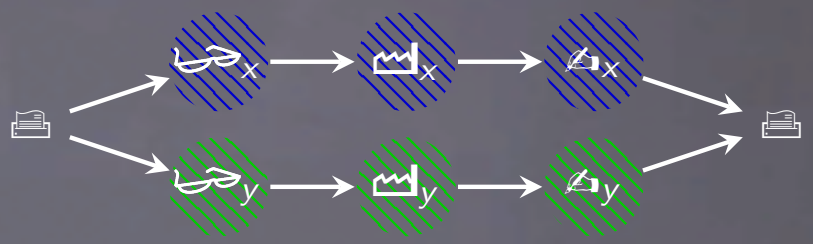
Mutual exclusion

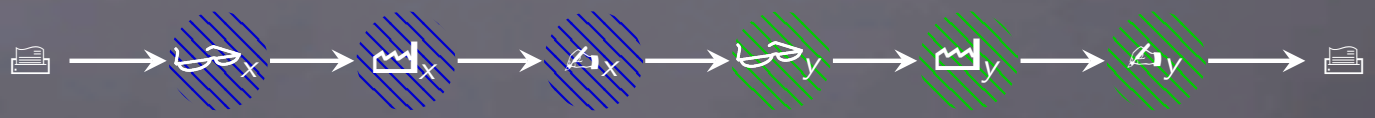
```
print(counter);
        ||
x:=counter;  ||  y:=counter;
x:=x+1;      ||  y:=y+1;
counter:=x;  ||  counter:=y;
        ||
print(counter);
```



Mutual exclusion

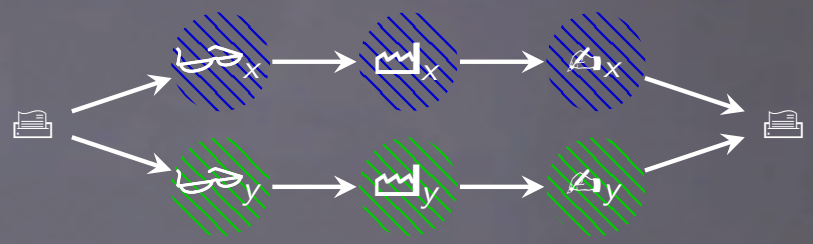
```
print(counter);
        ||
x:=counter;  ||  y:=counter;
x:=x+1;      ||  y:=y+1;
counter:=x;  ||  counter:=y;
        ||
print(counter);
```

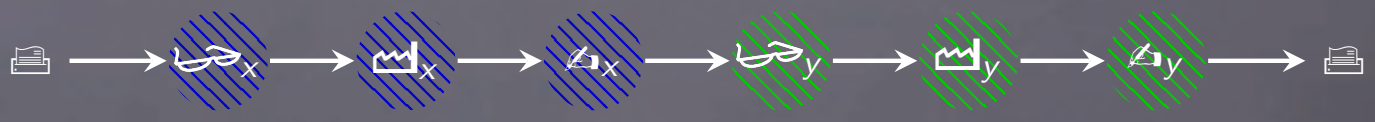


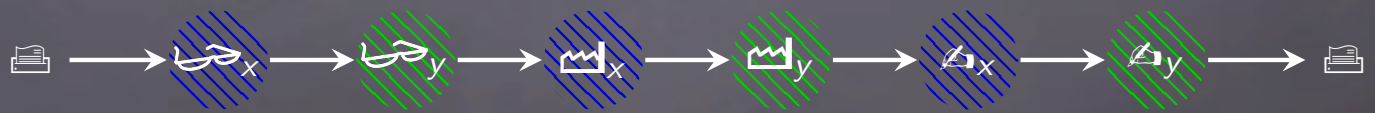


Mutual exclusion

```
print(counter);
        ||
x:=counter;  ||  y:=counter;
x:=x+1;      ||  y:=y+1;
counter:=x;  ||  counter:=y;
        ||
print(counter);
```

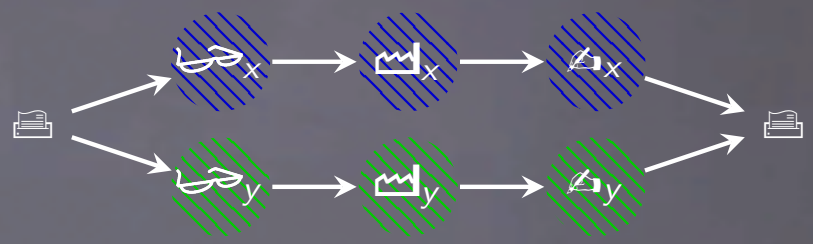






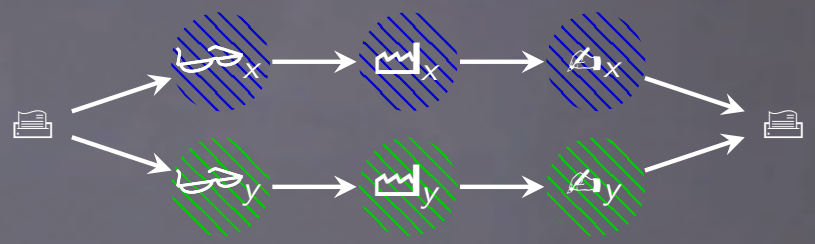
Mutual exclusion

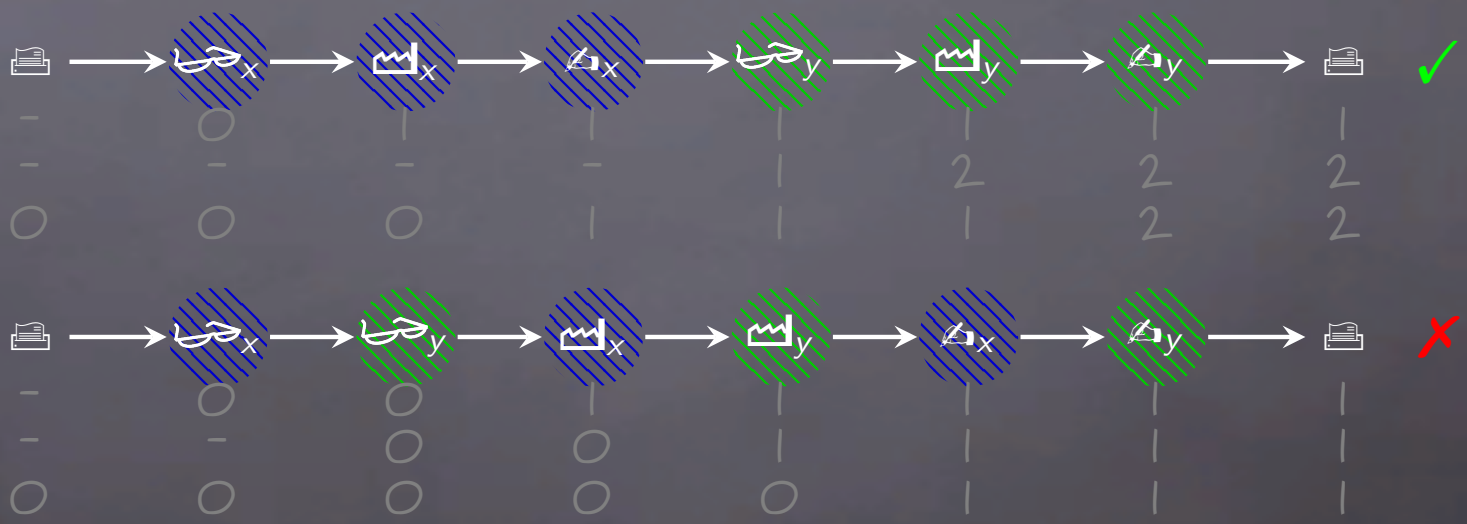
```
print(counter);
        ||
x:=counter;  ||  y:=counter;
x:=x+1;      ||  y:=y+1;
counter:=x;  ||  counter:=y;
        ||
print(counter);
```



Mutual exclusion

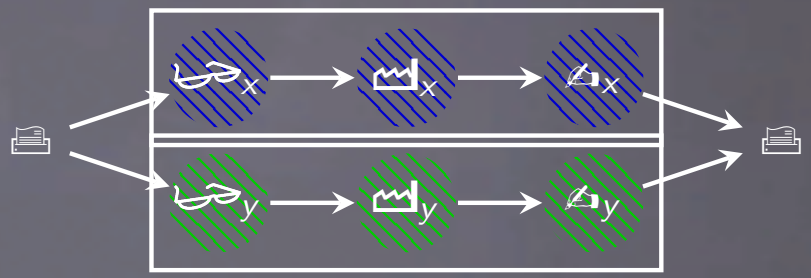
```
        print(counter);
atomic{          ||  atomic{
  x:=counter;    ||    y:=counter;
  x:=x+1;        ||    y:=y+1;
  counter:=x;    ||    counter:=y;
}                ||  }
        print(counter);
```

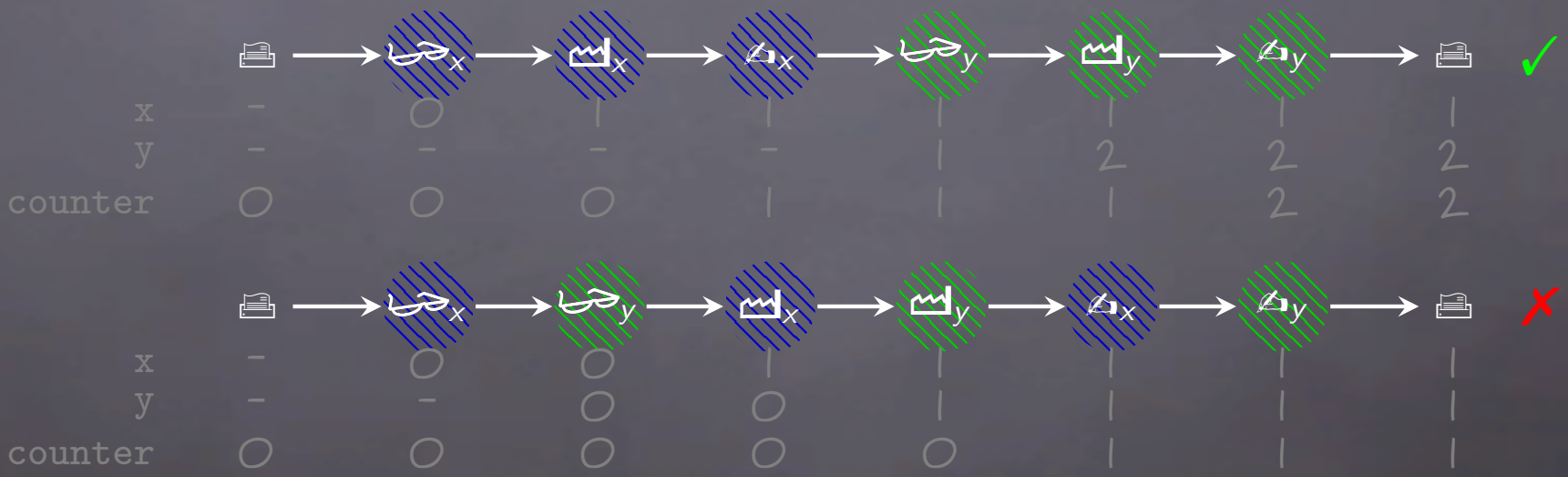




Mutual exclusion

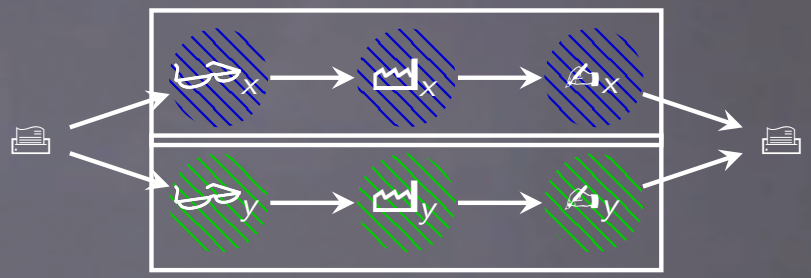
```
        print(counter);
atomic{          ||  atomic{
  x:=counter;    ||    y:=counter;
  x:=x+1;        ||    y:=y+1;
  counter:=x;    ||    counter:=y;
}                ||  }
        print(counter);
```

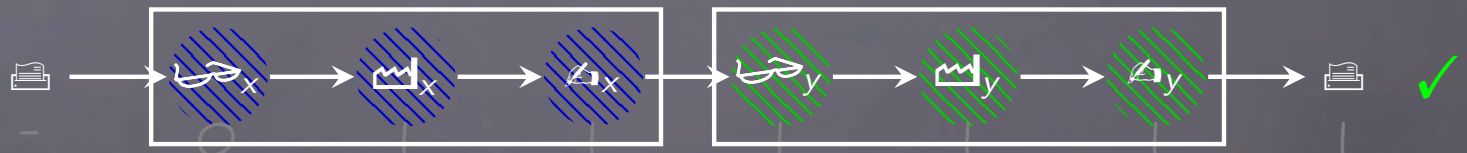




Mutual exclusion

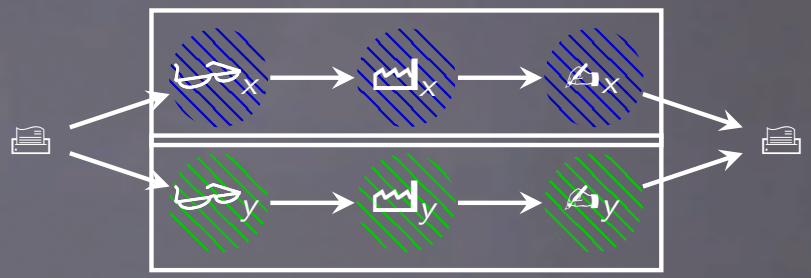
```
        print(counter);
atomic{          ||  atomic{
  x:=counter;    ||    y:=counter;
  x:=x+1;        ||    y:=y+1;
  counter:=x;    ||    counter:=y;
}                ||  }
        print(counter);
```

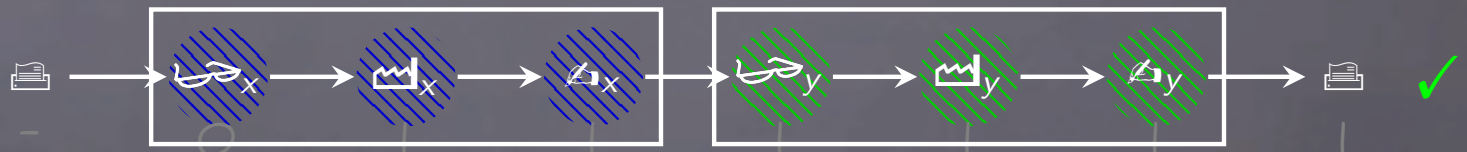


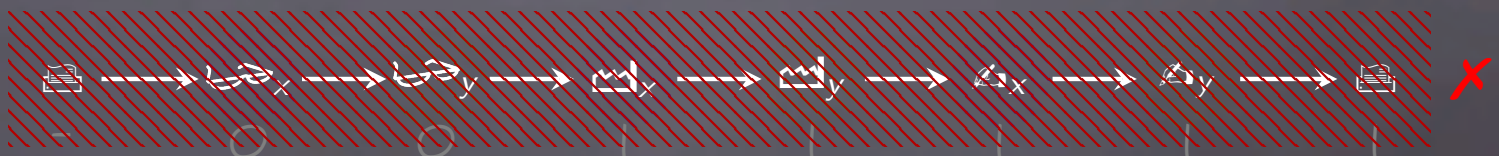


Mutual exclusion

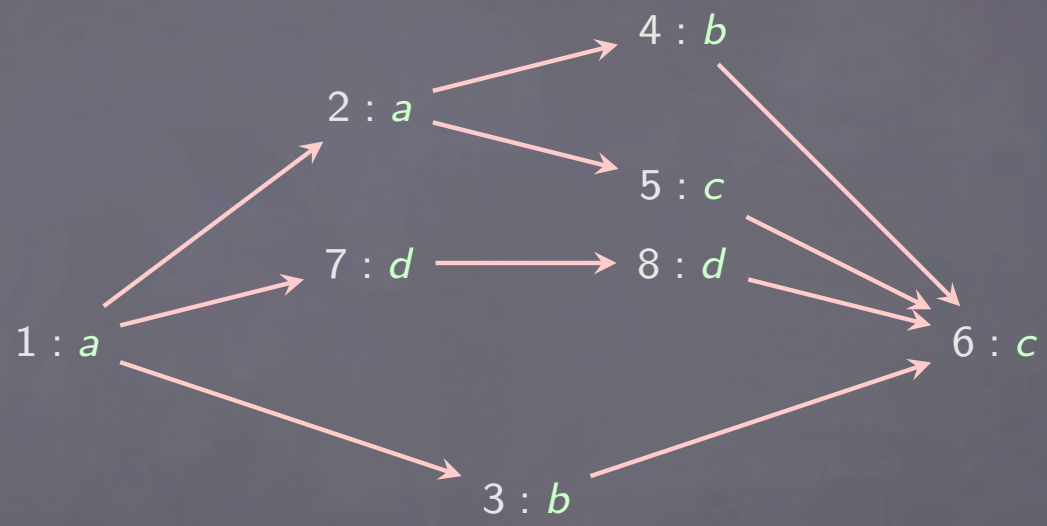
```
        print(counter);
atomic{          ||  atomic{
   x:=counter;   ||     y:=counter;
   x:=x+1;       ||     y:=y+1;
   counter:=x;   ||     counter:=y;
}                ||  }
        print(counter);
```







Pomsets with boxes



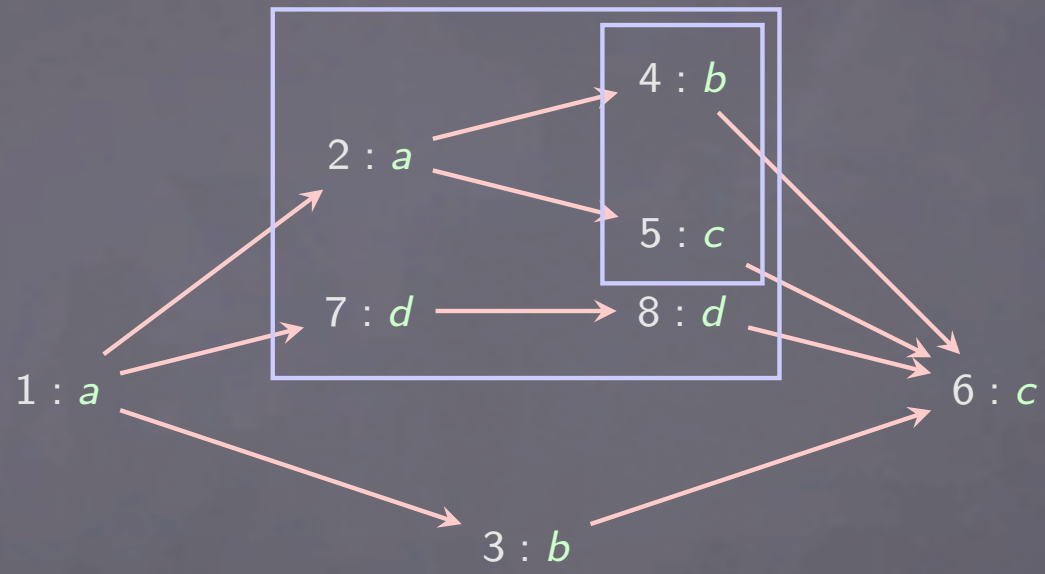
finite set of events

labeling function
$: E_P \to A$

$P = \langle E_P, \leqslant_P, \lambda_P \rangle$

partial order
$\subseteq E_P \times E_P$

Pomsets with boxes



finite set of events

labeling function $: E_P \to A$

$$P = \langle E_P, \leqslant_P, \lambda_P, B_P \rangle$$

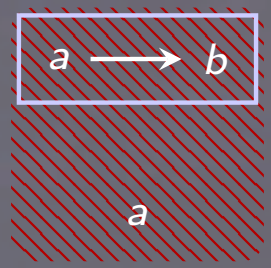
partial order $\subseteq E_P \times E_P$

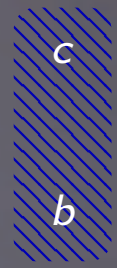
set of boxes $\subseteq \mathcal{P}^{\neq \emptyset}(E_P)$

Combining pomsets

$a =$ 

$1 =$ 

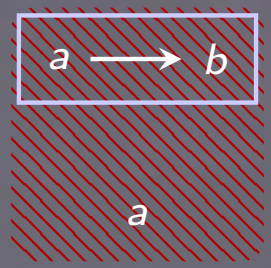
$P_1 =$ 

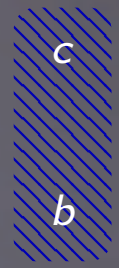
$P_2 =$ 

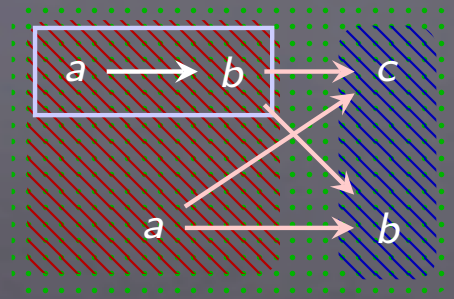
Combining pomsets

$a =$ 

$1 =$ 

$P_1 =$ 

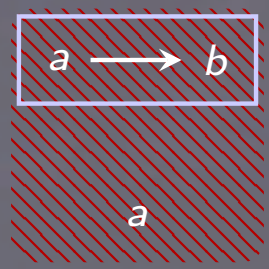
$P_2 =$ 

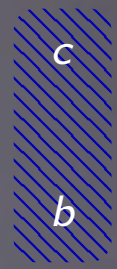
$P_1; P_2 =$ 

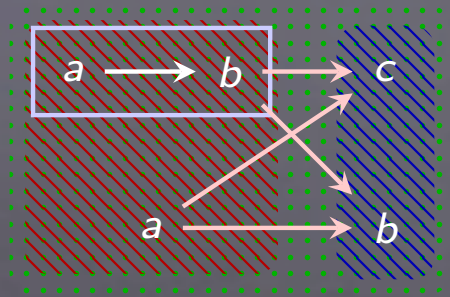
Combining pomsets

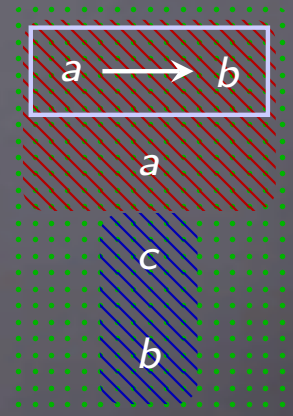
$a =$ 

$1 =$ 

$P_1 =$ 

$P_2 =$ 

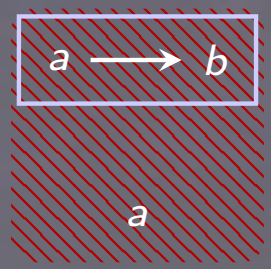
$P_1; P_2 =$ 

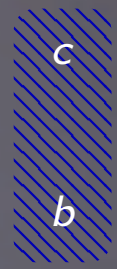
$P_1 \parallel P_2 =$ 

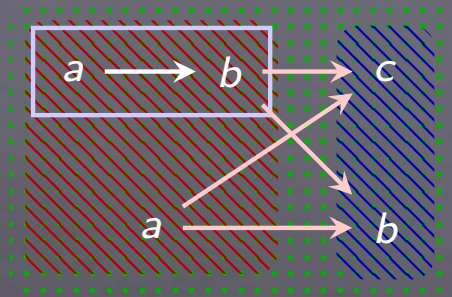
Combining pomsets

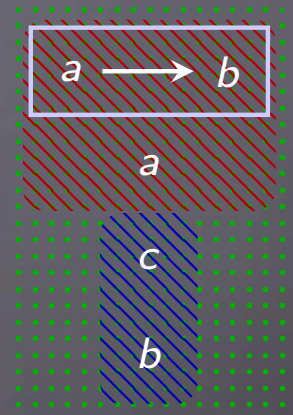
$a =$ 

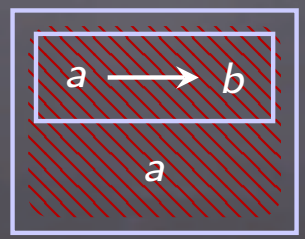
$1 =$ 

$P_1 =$ 

$P_2 =$ 

$P_1; P_2 =$ 

$P_1 \parallel P_2 =$ 

$[P_1] =$ 

Characterisation of SP-pomsets with boxes

Question

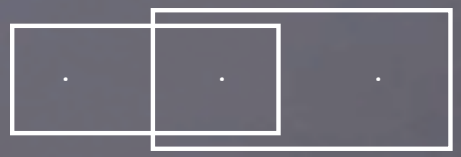
What pomsets can be built with the signature $\langle A, \cdot, \|, [-] \rangle$?

Characterisation of SP-pomsets with boxes

Question

What pomsets can be built with the signature $\langle A, \cdot, \|, [-] \rangle$?

Those that do not include the following patterns:



Axiomatisation of isomorphism

$$BiMon_{\square} \vdash \quad P;(Q;R) = (P;Q);R$$
$$BiMon_{\square} \vdash \quad P;1 = 1;P$$

$$BiMon_{\square} \vdash \quad P \parallel (Q \parallel R) = (P \parallel Q) \parallel R$$
$$BiMon_{\square} \vdash \quad P \parallel Q = Q \parallel P$$
$$BiMon_{\square} \vdash \quad P \parallel 1 = 1 \parallel P$$

$$BiMon_{\square} \vdash \quad [1] = 1$$
$$BiMon_{\square} \vdash \quad [[P]] = [P]$$

Axiomatisation of isomorphism

$$BiMon_{\square} \vdash \quad P;(Q;R) = (P;Q);R$$
$$BiMon_{\square} \vdash \quad P;1 = 1;P$$

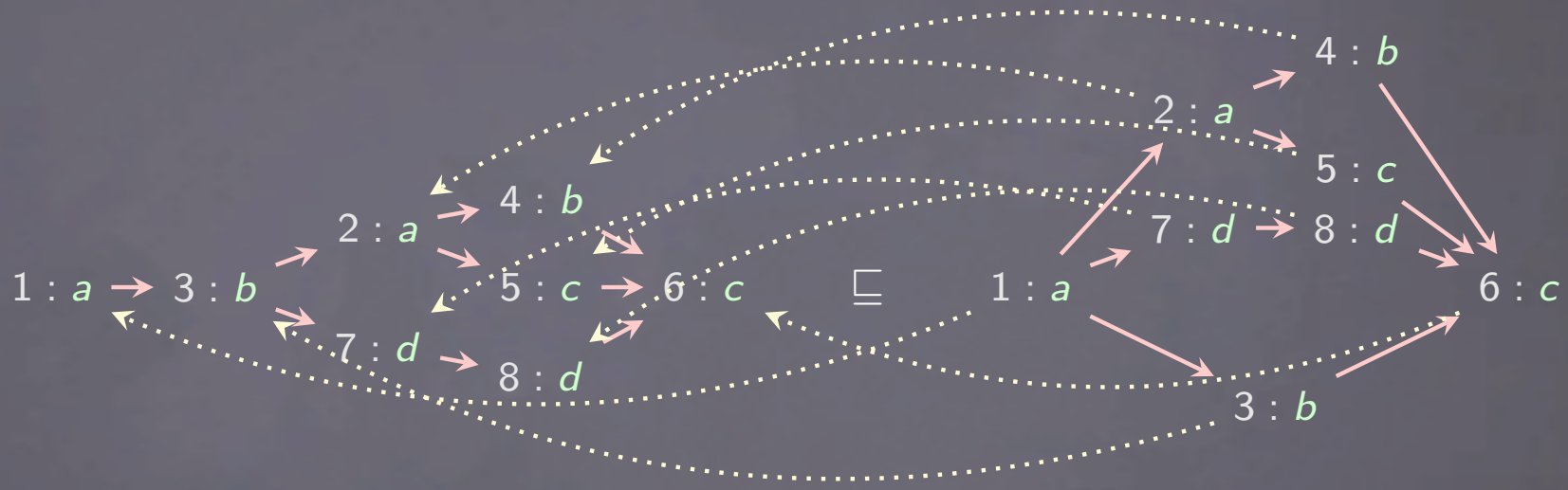
$$BiMon_{\square} \vdash \quad P \parallel (Q \parallel R) = (P \parallel Q) \parallel R$$
$$BiMon_{\square} \vdash \quad P \parallel Q = Q \parallel P$$
$$BiMon_{\square} \vdash \quad P \parallel 1 = 1 \parallel P$$

$$BiMon_{\square} \vdash \quad [1] = 1$$
$$BiMon_{\square} \vdash \quad [[P]] = [P]$$

Theorem

$$P \equiv Q \Leftrightarrow \ BiMon_{\square} \vdash P = Q.$$

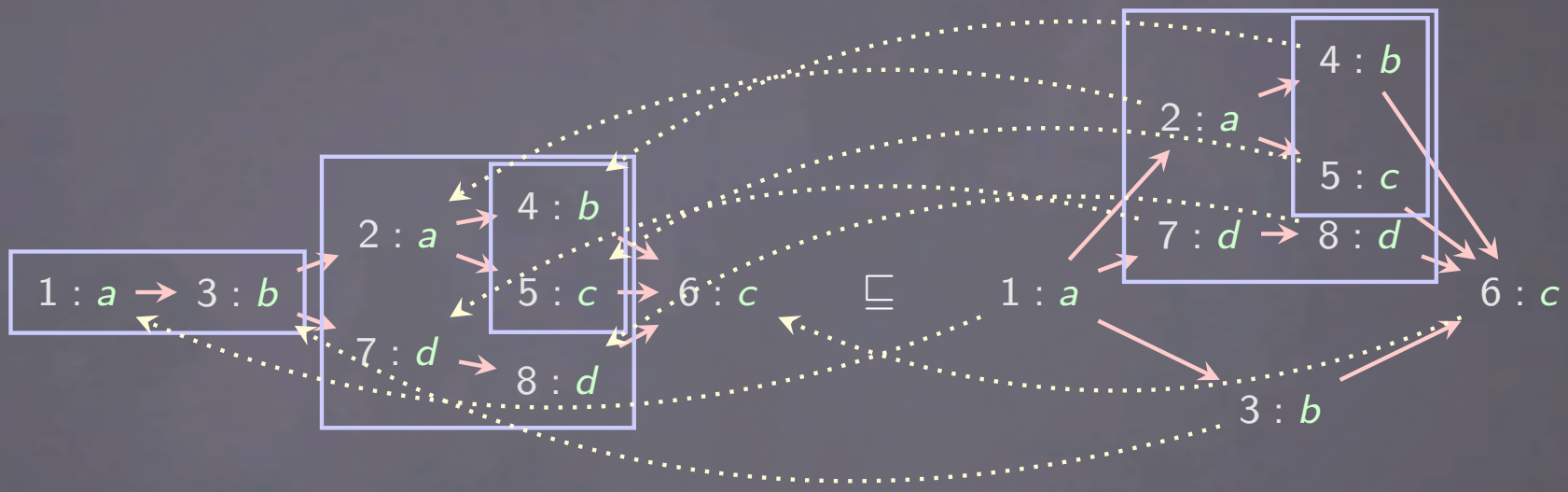
Subsumption with boxes



$P \sqsubseteq Q$ when there is a homomorphism from Q to P, i.e. a bijective map $\varphi : E_Q \to E_P$ such that

1. $\lambda_P \circ \varphi = \lambda_Q$
2. $\varphi(\leq_Q) \subseteq \leq_P$

Subsumption with boxes



$P \sqsubseteq Q$ when there is a homomorphism from Q to P, i.e. a bijective map $\varphi : E_Q \to E_P$ such that

1) $\lambda_P \circ \varphi = \lambda_Q$
2) $\varphi(\leq_Q) \subseteq \leq_P$
3) $\varphi(\mathcal{B}_P) \subseteq \mathcal{B}_Q$

Axiomatisation of subsumption

$$BiMon_{\square} \subseteq CMon_{\square}$$

$$CMon_{\square} \vdash \quad (P \parallel Q);(R \parallel S) \sqsubseteq (P;R) \parallel (Q;S)$$

$$CMon_{\square} \vdash \quad [P] \sqsubseteq P$$

Axiomatisation of subsumption

$$BiMon_{\square} \subseteq CMon_{\square}$$

$$CMon_{\square} \vdash \quad (P \parallel Q);(R \parallel S) \sqsubseteq (P;R) \parallel (Q;S)$$

$$CMon_{\square} \vdash \quad [P] \sqsubseteq P$$

Theorem

$$P \sqsubseteq Q \Leftrightarrow \ CMon_{\square} \vdash P \sqsubseteq Q.$$

Example: voting protocol

$$
\begin{aligned}
\texttt{VoteProc} &\coloneqq \texttt{Choose};\texttt{Publish} \\
\texttt{Choose} &\coloneqq [\texttt{Vote}(1)] \parallel \cdots \parallel [\texttt{Vote}(n)] \\
\texttt{Vote}(i) &\coloneqq \sum_{1 \leqslant j \leqslant k} \boxtimes_{i,j}; \hookrightarrow_j; \boxplus; \boxdot_j \\
\texttt{Publish} &\coloneqq \text{☎}_1 \parallel \cdots \parallel \text{☎}_n
\end{aligned}
$$

v_i : voter i
c_j : counter for candidate j
☎$_i$: the contents of counters $c_1, \ldots, c_k$ is sent to voter v_i
$\boxtimes_{i,j}$: voter v_i chooses counter c_j
$\hookrightarrow_j$: the content of counter c_j is loaded into a local variable
$\boxplus$: the local variable is incremented
$\boxdot_j$: the content of the local variable is stored in counter c_j

Non-deterministic programs

$$e, f ::= 0 \mid 1 \mid a \mid e \cdot f \mid e \parallel f \mid e + f \mid [e]\,.$$

Axioms

- Bisemiring with boxes: $SR_{\Box} = BiMon_{\Box} +$

$$e + f = f + e \qquad e + (f + g) = (e + f) + g \qquad e + 0 = e$$

$$e \cdot (f + g) = e \cdot f + e \cdot g \qquad (e + f) \cdot g = e \cdot g + f \cdot g$$

$$e \parallel (f + g) = e \parallel f + e \parallel g \qquad (e + f) \parallel g = e \parallel g + f \parallel g$$

$$e \cdot 0 = 0 \cdot e = 0 \qquad e \parallel 0 = 0$$

$$[e + f] = [e] + [f] \qquad [0] = 0$$

- Concurrent semiring with boxes:
  $CSR_{\Box} = SR_{\Box}$ + interchange + $[e] \leq e$.

Completeness theorems

$[\![e]\!]$: finite set of pomsets with boxes.

$[\![e]\!] \subseteq [\![f]\!]$: $\forall P \in [\![e]\!], \exists Q \in [\![f]\!] : P \equiv Q$.

$[\![e]\!] \sqsubseteq [\![f]\!]$: $\forall P \in [\![e]\!], \exists Q \in [\![f]\!] : P \sqsubseteq Q$.

Completeness theorems

$[\![e]\!]$: finite set of pomsets with boxes.

$[\![e]\!] \subseteq [\![f]\!]$: $\forall P \in [\![e]\!], \exists Q \in [\![f]\!] : P \equiv Q$.

$[\![e]\!] \sqsubseteq [\![f]\!]$: $\forall P \in [\![e]\!], \exists Q \in [\![f]\!] : P \sqsubseteq Q$.

Theorem

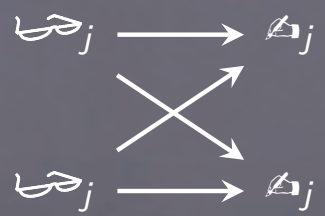
$$[\![e]\!] \subseteq [\![f]\!] \Leftrightarrow SR_{\Box} \vdash e + f = f.$$

$$[\![e]\!] \sqsubseteq [\![f]\!] \Leftrightarrow CSR_{\Box} \vdash e + f = f.$$

Mutual exclusion (II)

$$
\begin{aligned}
\texttt{VoteProc} &:= \texttt{Choose}; \texttt{Publish} \\
\texttt{Choose} &:= [\texttt{Vote}(1)] \parallel \cdots \parallel [\texttt{Vote}(n)] \\
\texttt{Vote}(i) &:= \textstyle\sum_{1 \leqslant j \leqslant k} \boxtimes_{i,j}; \hookrightarrow_j; \boxplus; \lightning_j \\
\texttt{Publish} &:= \text{☎}_1 \parallel \cdots \parallel \text{☎}_n
\end{aligned}
$$

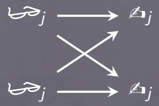
Breaking mutual exclusion
$\Leftrightarrow$ admitting an execution with the following "pattern":



Mutual exclusion (II)

$$\begin{aligned}
\texttt{VoteProc} &:= \texttt{Choose}; \texttt{Publish}\\
\texttt{Choose} &:= [\texttt{Vote}\,(1)] \parallel \cdots \parallel [\texttt{Vote}\,(n)]\\
\texttt{Vote}\,(i) &:= \textstyle\sum_{1\leqslant j\leqslant k} \boxtimes_{i,j}; 👓_j; ♔; ✍_j\\
\texttt{Publish} &:= ☎_1 \parallel \cdots \parallel ☎_n
\end{aligned}$$

Breaking mutual exclusion
$\Leftrightarrow$ admitting an execution with the following "pattern":



Problem: cannot be expressed as *program* $\sqsubseteq$ *property*.

Pomset logic

$$\varphi, \psi ::= \bot \mid a \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \varphi \blacktriangleright \psi \mid \varphi \star \psi \mid [\varphi] \mid (\!|\varphi|\!)$$

- $P \models_{\sqsupseteq} \varphi \blacktriangleright \psi$ iff $\exists P_1, P_2$ s.t. $P \sqsupseteq P_1 \cdot P_2$, $P_1 \models_{\sqsupseteq} \varphi$ and $P_2 \models_{\sqsupseteq} \psi$
- $P \models_{\sqsupseteq} \varphi \star \psi$ iff $\exists P_1, P_2$ s.t. $P \sqsupseteq P_1 \parallel P_2$, $P_1 \models_{\sqsupseteq} \varphi$ and $P_2 \models_{\sqsupseteq} \psi$
- $P \models_{\sqsupseteq} [\varphi]$ iff $\exists Q$ s.t. $P \sqsupseteq [Q]$ and $Q \models_{\sqsupseteq} \varphi$
- $P \models_{\sqsupseteq} (\!|\varphi|\!)$ iff $\exists P', Q$ s.t. $P \sqsupseteq P'$ and $P' \Rsh Q$ and $Q \models_{\sqsupseteq} \varphi$.

Pomset logic

$$\varphi, \psi ::= \bot \mid a \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \varphi \blacktriangleright \psi \mid \varphi \star \psi \mid [\varphi] \mid (\!|\varphi|\!)$$

- $P \models_{\sqsupseteq} \varphi \blacktriangleright \psi$ iff $\exists P_1, P_2$ s.t. $P \sqsupseteq P_1 \cdot P_2$, $P_1 \models_{\sqsupseteq} \varphi$ and $P_2 \models_{\sqsupseteq} \psi$
- $P \models_{\sqsupseteq} \varphi \star \psi$ iff $\exists P_1, P_2$ s.t. $P \sqsupseteq P_1 \parallel P_2$, $P_1 \models_{\sqsupseteq} \varphi$ and $P_2 \models_{\sqsupseteq} \psi$
- $P \models_{\sqsupseteq} [\varphi]$ iff $\exists Q$ s.t. $P \sqsupseteq [Q]$ and $Q \models_{\sqsupseteq} \varphi$
- $P \models_{\sqsupseteq} (\!|\varphi|\!)$ iff $\exists P', Q$ s.t. $P \sqsupseteq P'$ and $P' \Rsh Q$ and $Q \models_{\sqsupseteq} \varphi$.

Pomset logic

$$\varphi, \psi ::= \bot \mid a \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \varphi \blacktriangleright \psi \mid \varphi \star \psi \mid [\varphi] \mid (\!|\varphi|\!)$$

- $P \models_{\sqsupseteq} \varphi \blacktriangleright \psi$ iff $\exists P_1, P_2$ s.t. $P \sqsupseteq P_1 \cdot P_2$, $P_1 \models_{\sqsupseteq} \varphi$ and $P_2 \models_{\sqsupseteq} \psi$
- $P \models_{\sqsupseteq} \varphi \star \psi$ iff $\exists P_1, P_2$ s.t. $P \sqsupseteq P_1 \parallel P_2$, $P_1 \models_{\sqsupseteq} \varphi$ and $P_2 \models_{\sqsupseteq} \psi$
- $P \models_{\sqsupseteq} [\varphi]$ iff $\exists Q$ s.t. $P \sqsupseteq [Q]$ and $Q \models_{\sqsupseteq} \varphi$
- $P \models_{\sqsupseteq} (\!|\varphi|\!)$ iff $\exists P', Q$ s.t. $P \sqsupseteq P'$ and $P' \Rsh Q$ and $Q \models_{\sqsupseteq} \varphi$.

Theorem

$$P \sqsupseteq Q \Leftrightarrow \forall \varphi, \, (P \models_{\sqsupseteq} \varphi \Rightarrow Q \models_{\sqsupseteq} \varphi)\,.$$

$$P \equiv Q \Leftrightarrow \forall \varphi, \, (P \models_{\equiv} \varphi \Leftrightarrow Q \models_{\equiv} \varphi)\,.$$

Pomset logic

Remark

When the satisfaction relation is $\models_{\equiv}$ we may use negations, with:

- $P \models_{\equiv} \neg\varphi$ iff $P \not\models_{\equiv} \varphi$.

$$\varphi, \psi ::= \bot \mid a \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \varphi \blacktriangleright \psi \mid \varphi \star \psi \mid [\varphi] \mid (\!|\varphi|\!)$$

- $P \models_{\sqsupseteq} \varphi \blacktriangleright \psi$ iff $\exists P_1, P_2$ s.t. $P \sqsupseteq P_1 \cdot P_2$, $P_1 \models_{\sqsupseteq} \varphi$ and $P_2 \models_{\sqsupseteq} \psi$
- $P \models_{\sqsupseteq} \varphi \star \psi$ iff $\exists P_1, P_2$ s.t. $P \sqsupseteq P_1 \parallel P_2$, $P_1 \models_{\sqsupseteq} \varphi$ and $P_2 \models_{\sqsupseteq} \psi$
- $P \models_{\sqsupseteq} [\varphi]$ iff $\exists Q$ s.t. $P \sqsupseteq [Q]$ and $Q \models_{\sqsupseteq} \varphi$
- $P \models_{\sqsupseteq} (\!|\varphi|\!)$ iff $\exists P', Q$ s.t. $P \sqsupseteq P'$ and $P' \Rsh Q$ and $Q \models_{\sqsupseteq} \varphi$.

Theorem

$$P \sqsupseteq Q \Leftrightarrow \forall \varphi, \, (P \models_{\sqsupseteq} \varphi \Rightarrow Q \models_{\sqsupseteq} \varphi)\,.$$

$$P \equiv Q \Leftrightarrow \forall \varphi, \, (P \models_{\equiv} \varphi \Leftrightarrow Q \models_{\equiv} \varphi)\,.$$

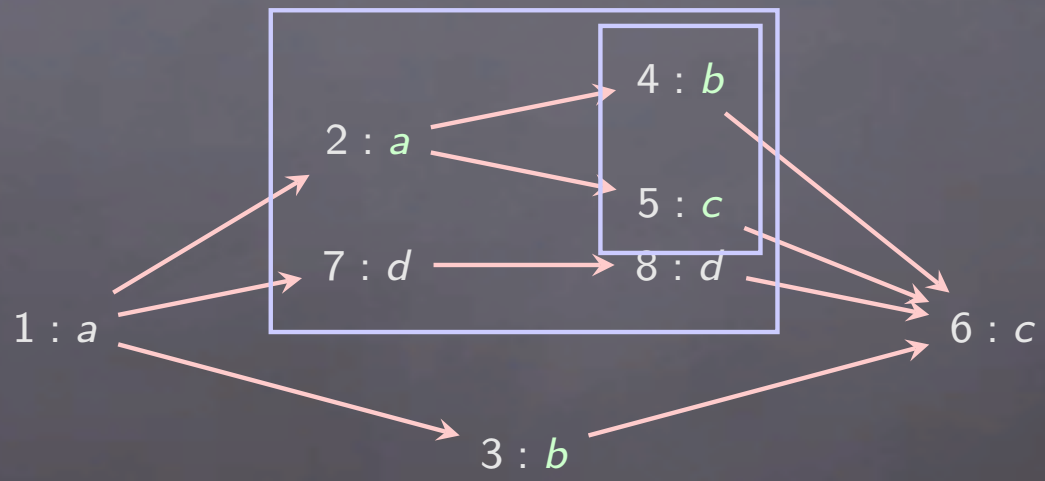
Contexts

☞ $P \models_{\sqsupseteq} (\!|\varphi|\!)$ iff $\exists P', Q$ such that $P \sqsupseteq P'$ and $P' \trianglerighteq Q$ and $Q \models_{\sqsupseteq} \varphi$.

Subpomset

$P \trianglerighteq Q$ iff there exists a set of events $A \subseteq E_P$ s.t. $Q \equiv P|_A$.

$A = \{2, 3, 4, 5\}$



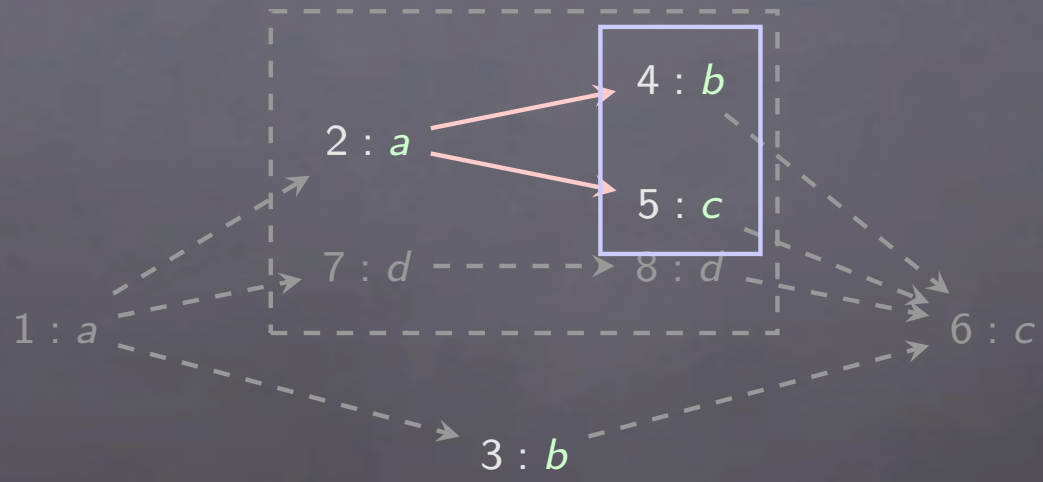
Contexts

☞ $P \models_{\sqsupseteq} (\!|\varphi|\!)$ iff $\exists P', Q$ such that $P \sqsupseteq P'$ and $P' \trianglerighteq Q$ and $Q \models_{\sqsupseteq} \varphi$.

Subpomset

$P \trianglerighteq Q$ iff there exists a set of events $A \subseteq E_P$ s.t. $Q \equiv P|_A$.

$A = \{2, 3, 4, 5\}$



Extension to non-deterministic programs

Definition

$$e \models^{\forall}_{R} \varphi \text{ iff } \forall P \in [\![e]\!], P \models_{R} \varphi$$

$$e \models^{\exists}_{R} \varphi \text{ iff } \exists P \in [\![e]\!], P \models_{R} \varphi$$

Extension to non-deterministic programs

Definition

$$e \models^{\forall}_{R} \varphi \text{ iff } \forall P \in [\![e]\!], P \models_{R} \varphi$$

$$e \models^{\exists}_{R} \varphi \text{ iff } \exists P \in [\![e]\!], P \models_{R} \varphi$$

Theorem

$$\begin{aligned} SR_{\square} \vdash e \leq f &\Leftrightarrow \forall \varphi,\ e \models^{\exists}_{\equiv} \varphi \Rightarrow f \models^{\exists}_{\equiv} \varphi \\ &\Leftrightarrow \forall \varphi,\ f \models^{\forall}_{\equiv} \varphi \Rightarrow e \models^{\forall}_{\equiv} \varphi \\ CSR_{\square} \vdash e \leq f &\Leftrightarrow \forall \varphi,\ e \models^{\exists}_{\sqsupseteq} \varphi \Rightarrow f \models^{\exists}_{\sqsupseteq} \varphi \\ &\Leftrightarrow \forall \varphi,\ f \models^{\forall}_{\sqsubseteq} \varphi \Rightarrow e \models^{\forall}_{\sqsubseteq} \varphi \end{aligned}$$

Mutual exclusion (III)

$$\begin{aligned}\texttt{VoteProc} &:= \texttt{Choose}; \texttt{Publish}\\ \texttt{Choose} &:= [\texttt{Vote}(1)] \parallel \cdots \parallel [\texttt{Vote}(n)]\\ \texttt{Vote}(i) &:= \textstyle\sum_{1 \leqslant j \leqslant k} \boxtimes_{i,j}; \hookrightarrow_j; \boxminus; \lightning_j\\ \texttt{Publish} &:= \text{☎}_1 \parallel \cdots \parallel \text{☎}_n\end{aligned}$$

Breaking mutual exclusion
$\Leftrightarrow$ admitting an execution with the following "pattern":

$$\mathsf{conflict}_j := \langle\!\langle (\hookrightarrow_j \star \hookrightarrow_j) \blacktriangleright (\lightning_j \star \lightning_j) \rangle\!\rangle.$$

$$\begin{aligned}\texttt{VoteProc}' &:= \texttt{Choose}'; \texttt{Publish}\\ \texttt{Choose}' &:= \texttt{Vote}(1) \parallel \cdots \parallel \texttt{Vote}(n)\end{aligned}$$

$$\texttt{VoteProc} \not\models^{\exists}_{\sqsupseteq} \mathsf{conflict}_j \qquad \texttt{VoteProc}' \models^{\exists}_{\sqsupseteq} \mathsf{conflict}_j.$$

Spacio-Temporal separation

$$\begin{aligned}\texttt{VoteProc} &:= \texttt{Choose}; \texttt{Publish}\\ \texttt{Choose} &:= [\texttt{Vote}(1)] \parallel \cdots \parallel [\texttt{Vote}(n)]\\ \texttt{Vote}(i) &:= \textstyle\sum_{1 \leqslant j \leqslant k} \boxtimes_{i,j}; \hookrightarrow_j; \lightning; \lightning_j\\ \texttt{Publish} &:= \text{☎}_1 \parallel \cdots \parallel \text{☎}_n\end{aligned}$$

$$\mathsf{SendAfterVote} := \neg\left(\left\|\bigvee_i \text{☎}_i\right\|\right) \blacktriangleright \neg\left(\left\|\bigvee_{i,j} \boxtimes_{i,j}\right\|\right).$$

$$\texttt{Choose} \models^{\forall}_{\equiv} \neg\left(\left\|\bigvee_i \text{☎}_i\right\|\right) \text{ and } \texttt{Publish} \models^{\forall}_{\equiv} \neg\left(\left\|\bigvee_{i,j} \boxtimes_{i,j}\right\|\right)$$

$$\Rightarrow \texttt{VoteProc} \models^{\forall}_{\equiv} \mathsf{SendAfterVote}$$

$$\mathsf{VoteThenSend} := \left\|\left(\bigvee_j \boxtimes_{1,j} \blacktriangleright \text{☎}_1\right) \star \ldots \star \left(\bigvee_j \boxtimes_{n,j} \blacktriangleright \text{☎}_n\right)\right\|$$

$$\texttt{VoteProc} \models^{\forall}_{\equiv} \mathsf{VoteThenSend}.$$

Unique votes

$$\begin{aligned}\texttt{VoteProc} &:= \texttt{Choose}; \texttt{Publish}\\ \texttt{Choose} &:= [\texttt{Vote}(1)] \parallel \cdots \parallel [\texttt{Vote}(n)]\\ \texttt{Vote}(i) &:= \textstyle\sum_{1 \leqslant j \leqslant k} \boxtimes_{i,j}; \hookrightarrow_j; \lightning; \text{✍}_j\\ \texttt{Publish} &:= \text{☎}_1 \parallel \cdots \parallel \text{☎}_n\end{aligned}$$

$$\texttt{VoteProc} \not\models \overset{\exists}{\sqsubseteq} \bigvee_{j',j} \langle\!|[\langle\!|\text{✍}_j \star \text{✍}_{j'}|\!\rangle]|\!\rangle$$

Further work

Algebra:

Logic:

Further work

Algebra:

- ☞ Where's my star?

Logic:

Further work

Algebra:

- Where's my star?
- Merge axiom:

$$[a; b] \leq [a] ; [b]$$

Logic:

Further work

Algebra:

- Where's my star?
- Merge axiom:

$$[a; b] \leq [a] ; [b]$$

- More operators/axioms: observations, nominals, probabilities...

Logic:

Further work

Algebra:

- Where's my star?
- Merge axiom:

$$[a; b] \leq [a] ; [b]$$

- More operators/axioms: observations, nominals, probabilities...

Logic:

- Deduction rules / proof system?

Further work

Algebra:

- Where's my star?
- Merge axiom:

$$[a; b] \leq [a] ; [b]$$

- More operators/axioms: observations, nominals, probabilities...

Logic:

- Deduction rules / proof system?
- Logic of Behaviour

Further work

Algebra:

- Where's my star?
- Merge axiom:

$$[a; b] \leq [a] ; [b]$$

- More operators/axioms: observations, nominals, probabilities...

Logic:

- Deduction rules / proof system?
- Logic of Behaviour
  - Traditional approaches to program logic rely on states e.g. Hennessy-Milner Logic, (Propositional) Dynamic Logic...

Further work

Algebra:

- Where's my star?
- Merge axiom:

$$[a; b] \leq [a] \,;\, [b]$$

- More operators/axioms: observations, nominals, probabilities...

Logic:

- Deduction rules / proof system?
- Logic of Behaviour
  - Traditional approaches to program logic rely on states e.g. Hennessy-Milner Logic, (Propositional) Dynamic Logic...
  - Pomset logic relies on an abstract notion of "behaviour" instead.

Further work

Algebra:

- Where's my star?
- Merge axiom:

$$[a; b] \leq [a] ; [b]$$

- More operators/axioms: observations, nominals, probabilities...

Logic:

- Deduction rules / proof system?
- Logic of Behaviour
  - Traditional approaches to program logic rely on states e.g. Hennessy-Milner Logic, (Propositional) Dynamic Logic...
  - Pomset logic relies on an abstract notion of "behaviour" instead.
  - What kind of properties can be expressed?

That's all folks!

Thank you!

See more at:
http://paul.brunet-zamansky.fr