Concurrency

G. Michele Pinna

Dipartimento di Matematica e Informatica
University of Cagliari

QCOMICAL@Nancy

A kind of introduction...

- what is concurrency about?
- concurrency is about handling multiple tasks over the same period of time, giving the illusion they are running simultaneously by rapidly switching between them
- parallelism is about actually executing multiple tasks at the exact same time on different processors
- communication is about on how tasks coordinate
- **classic** not quantum

A kind of introduction...

- we have that concurrency arises when several computational components operate simultaneously
- a sequential semantics fail to represent simultaneous events
- We require abstract models to describe
  - independence
  - synchronization
  - causal relationships
- languages to describe the components and how they iteract

Two fundamental perspectives

- *Interleaving*
  - concurrency represented as nondeterministic sequencing
  - loses explicit information about simultaneity
- *True Concurrency*
  - distinguishes causal dependence, independence and conflict
  - provides a more accurate view of concurrent processes

today mainly **interleaving**

Specification languages

- how to describe concurrent systems?

The best way would be having a mathematically rigorous language with well defined semantics that permit describing and verifying properties of concurrent communicating systems

- Process Algebras

A Process Algebra can be seen as a model of processes, regarded as agents that act and interact continuously with other similar agents and with their common environment

- there are a lot of them: CCS, CSP, ACP, π-calculus . . . and many more

What are Process Algebras?

- formal languages and calculi to specify and reason about concurrent interacting systems
- stress on composition, synchronisation, nondeterminism and interaction with an environment
- useful for specification, verification and modelling of protocols, concurrent software and distributed systems
- well suited to specify reactive behaviour: continuous interaction, no well-defined termination
- emphasis on *actions*: visible versus internal (the τ)
- *compositionality*: build complex systems from components

A tour on Process Algebra elements

Typical syntactic constructs

$$P ::= \mathbf{0} \mid a.P \mid P + Q \mid P \parallel Q \mid P \setminus L \mid P[f] \mid X \mid \text{rec } X\ P$$

where the intuition is

- $\mathbf{0}$ does nothing
- $a.P$ action prefixing
- $P + Q$ nondeterministic choice
- $P \parallel Q$ parallel composition
- $P \setminus L$ restrict to actions in L
- $P[f]$ rename actions according to f
- rec $X\ P$: recursion for infinite behaviour

Syntax – Other Building Blocks

one can have also other syntactic constructs

$$P ::= P; Q \mid P \oplus Q \mid P \lfloor\!\lfloor Q \mid A \triangleq P \cdots$$

where

- $P; Q$ a sequential composition
- $P \oplus Q$ external nondeterministic choice
- $P \lfloor\!\lfloor Q$ left parallel composition
- $A \triangleq P$ process definition
- ...

A brief recap: ingredients for a Process Algebra

- a minimal set of carefully chosen operators capturing the relevant aspect of systems behaviour and the way systems are composed in building process terms (we have seen a tiny bit)

- we focus on standard operators: action prefixing, choice, parallel composition, recursion...

$$P ::= \mathbf{0} \mid a.P \mid P + Q \mid P \parallel Q \mid P \setminus L \mid P[f] \mid X \mid \mathrm{rec}\ X\ P$$

The intuition behind the elements of a process algebra

- $\mathbf{0}$ does nothing
- it is a kind of successful termination (like $\surd$)
- we may introduce something for failure

Action Prefixing

$a.P$ first performs action a then behaves like P

- operational rule (informal): $a.P \xrightarrow{a} P$
- example: $coin.choc.\mathbf{0}$ — accept coin then dispense chocolate

Nondeterministic Choice

$P + Q$ represents a process that may behave as either P or Q

- operationally: if $P \xrightarrow{\alpha} P'$ then $P + Q \xrightarrow{\alpha} P'$ (and symmetrically for Q)
- distinguish internal choice (unobservable resolution) vs external choice (environment selects via offered actions)
- example: $\mathit{coin}.(\mathit{choc}.\mathbf{0} + \mathit{chips}.\mathbf{0})$ — accept a coin and then it chooses whether dispense chocolate or chips

Parallel Composition

$P \parallel Q$ executes P and Q concurrently

- interleaving: independent actions interleave
- example: $\mathit{choc}.\mathbf{0} \parallel \mathit{chips}.\mathbf{0}$ dispenses both chocolate and chips, either one after the other or together
- synchronisation: complementary actions may synchronise to a joint step (depending on calculus) - simultaneous
- example: $a.P \parallel \overline{a}.Q$ may synchronise yielding $P \parallel Q$

Restriction, Hiding and Renaming

- restriction $(P \setminus L)$: prevent actions (and co-action) in L from being visible — often turned into τ
- example: $(a.P \parallel \overline{a}.Q) \setminus \{a, \overline{a}\}$ has to synchronise yielding $(P \parallel Q) \setminus \{a, \overline{a}\}$
- renaming $P[f]$: apply a relabelling function f to action names
- example: $choc.\mathbf{0}[f]$ where $f(choc) = chips$ dispense chips

Recursion and Infinite Behaviour

- recursion lets us specify infinite behaviours
- guarded recursion (prefix on recursion) ensures well-behaved operational unfolding
- example: rec $X\ a.X$ models unbounded repetition of a
- many times we introduce names and definitions: $A \triangleq \alpha.B$ infinite behaviours arise when the name A occurs in B
- example: $X \triangleq a.X$ models unbounded repetition of a
- definitions are often handy

A bit more formal

- how to describe the behaviours?
- up to now we have given a kind of description of the behaviours of process algebras terms
- we want to be more formal to be able to reason on them
- let us discuss a bit about the semantics

Various ways of giving semantics

an operational semantics models a program as a labelled transition system

the states are just process algebra terms and the labels of the transitions represent the actions or the interactions

the operational semantics is close to an abstract machine-based view of computation and might be considered as a mathematical formalization of some implementation strategy

Various ways of giving semantics

a denotational semantics maps a language to some abstract model such that the meaning/denotation (in the model) of any composite program is determinable directly from the meanings/denotations of its subcomponents

denotational semantics attempt to distance themselves from any specific implementation strategy, describing the language at a level intended to capture the essential meaning of a term.

Various ways of giving semantics

an algebraic semantics is defined by a set of algebraic laws capturing the intended semantics of the constructs of the language

the laws are the basic axioms of an equational system, and process equivalence is defined in terms of what equalities can be proved using them

Labelled Transition System (LTS)

$$(\mathcal{Q}, \rightarrow, \mathcal{A}ct, q_0)$$

- a set of states $\mathcal{Q}$
- a labelled transition relation $\rightarrow \subseteq \mathcal{Q} \times \mathcal{A}ct \times \mathcal{Q}$
- an initial state q_0

we write $q \xrightarrow{a} q'$ for $(q, a, q') \in \rightarrow$

LTS are everywhere

An LTS for CCS

the labels

- fix $\mathcal{A}$ to be a set of actions and $\overline{\mathcal{A}} = \{\overline{a} \mid a \in \mathcal{A}\}$ be the set of their co-actions such that $\overline{\overline{a}} = a$
- prefix labels: $\alpha \ ::= \ a \mid \overline{a} \mid \tau$ with $a \in \mathcal{A}$
- $\mathcal{A}ct = \mathcal{A} \cup \overline{\mathcal{A}} \cup \{\tau\}$
- prefixes are actions: $\alpha \in \mathcal{A}ct$
- visible actions: $\mu \in \mathcal{A} \cup \overline{\mathcal{A}}$

An LTS for CCS

Processes are

$$P \ ::= \ \mathbf{0} \mid \alpha.P \mid P + Q \mid P \parallel Q \mid P \setminus L \mid X \mid \text{rec } X \ P$$

the LTS of a process P is $(\mathcal{Q}, \rightarrow, \mathcal{A}ct, P)$ where $\mathcal{Q}$ are processes and $\rightarrow$ is the relation defined by the SOS rules

we focus on the reachable part

Given a term P we construct the LTS associated to it using the SOS rules we will illustrate in the next slides

there is an equivalence relation around: e.g. $P \parallel Q$ is the same as $Q \parallel P$

Action Prefixing

recall: $\alpha.P$ first performs action α then behaves like P

- SOS rule:

$$\frac{}{a.P \xrightarrow{a} P}$$

Nondeterministic Choice

recall: $P + Q$ represents a process that may behave as either P or Q

- SOS rule

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \textit{and} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

Parallel Composition

recall: $P \parallel Q$ executes P and Q concurrently

- SOS rules for the interleaving

$$\frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q} \qquad\qquad \frac{Q \xrightarrow{\alpha} Q'}{P \parallel Q \xrightarrow{\alpha} P \parallel Q'}$$

- SOS rules for the synchronizations

$$\frac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\overline{\alpha}} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}$$

Restriction, Hiding and Renaming

recall: $P \setminus L$ prevent visible actions in L from being visible and $P[f]$ renames visible actions in P

- restriction SOS rule:

$$\frac{P \xrightarrow{\mu} P' \qquad \mu \notin L}{P \setminus L \xrightarrow{\mu} P' \setminus L}$$

- renaming SOS rule:

$$\frac{P \xrightarrow{\mu} P'}{P[f] \xrightarrow{f(\mu)} P'[f]}$$

Recursion and Infinite Behaviour

recall: rec $X.P$ executes P where X has to be substituted with the term itself

- SOS rule intuition: unfold one step of the recursive definition

$$\frac{P\{\text{rec } X.P/X\} \xrightarrow{\alpha} P'}{\text{rec } X.P \xrightarrow{\alpha} P'}$$

Recursion and Infinite Behaviour

recall: $A \triangleq P$ executes P where A has to be substituted with the term itself

- SOS rule:

$$\frac{P \xrightarrow{\alpha} P'}{A \xrightarrow{\alpha} P'} \quad \textit{where } A \triangleq P$$

Examples

- one can model a vending machine as follows

$$\begin{aligned} VM &= \textit{coin}.\, VM' \\ VM' &= \textit{choc}.VM + \textit{tea}.VM \end{aligned}$$

- really simple: upon inserting a coin it delivers either tea or chocolate but not both and then wait for another coin

$$\begin{tikzcd} VM \arrow[r, "\text{coin}"] & VM' \arrow[l, bend right=50, "\text{choc}"'] \arrow[l, bend left=50, "\text{tea}"] \end{tikzcd}$$

Examples

- one can model a counter as follows

$$C \;=\; up.\,(C \parallel down.\mathbf{0})$$

- each time an up is received then it increase the number of elements of the counter

- $C \xrightarrow{up} C \parallel down.\mathbf{0} \xrightarrow{up} C \parallel down.\mathbf{0} \parallel down.\mathbf{0} \xrightarrow{up} C \parallel down.\mathbf{0} \parallel down.\mathbf{0} \parallel down.\mathbf{0}$

$$\begin{tikzcd}
C \arrow[r, "up"] & C \parallel down.\mathbf{0} \arrow[l, bend right, "down"'] \arrow[r, "up"] & C \parallel down.\mathbf{0} \parallel down.\mathbf{0} \arrow[l, bend right, "down"'] & \cdots
\end{tikzcd}$$

C is equivalent to $C \parallel \mathbf{0}$ ...

Examples

- one can model a printer server as follows

$$\begin{aligned} PS &= work.\ print.PS \\ US &= \overline{work}.US \end{aligned}$$

$$(PS \parallel US) \setminus \{work\}$$

- the server and the user must synchronise on *work*

- $(PS \parallel US)\setminus\{work\} \xrightarrow{\tau} (print.PS \parallel US)\setminus\{work\} \xrightarrow{print} (PS \parallel US)$

$$\begin{tikzcd} (PS \parallel US) \setminus \{work\} \arrow[r, "\tau"] & (print.PS \parallel US) \setminus \{work\} \arrow[l, bend right, "print"'] \end{tikzcd}$$

Traces and trace equivalence

- a trace of process P is a (finite) sequence of actions it can perform
- paths in the LTS
- two processes are trace-equivalent if they have the same set of traces
- trace equivalence is easy but coarse: it ignores branching

$\mathit{coin}.(\mathit{choc}.\mathbf{0} + \mathit{tea}.\mathbf{0})$ and $\mathit{coin}.\mathit{choc}.\mathbf{0} + \mathit{coin}.\mathit{tea}.\mathbf{0}$ have the same traces

(Strong) bisimulation

- A binary relation R on processes is a bisimulation iff whenever $(P, Q) \in R$
    - For every $P \xrightarrow{\alpha} P'$ there exists $Q \xrightarrow{\alpha} Q'$ with $(P', Q') \in R$
    - Symmetrically for transitions from Q.
- P and Q are bisimilar, written $P \sim Q$, if there exists a bisimulation R with $(P, Q) \in R$

$\mathit{coin}.(\mathit{choc}.\mathbf{0} + \mathit{tea}.\mathbf{0})$ and $\mathit{coin}.\mathit{choc}.\mathbf{0} + \mathit{coin}.\mathit{tea}.\mathbf{0}$ are not bisimilar

Weak bisimulation

Abstract from internal τ actions

Use $\Rightarrow$ for zero-or-more τ steps require matching visible moves modulo τ sequences

Crucial for observational equivalence.

Relations among equivalences

- strong bisimilarity implies trace equivalence
- weak bisimilarity is coarser than strong bisimilarity but finer than traces in many settings
- congruence properties: strong bisimilarity is a congruence for CCS

A brief recup on LTSs

We have associated an LTS to an agent, and the paths in the LTS describe the behaviour of the agent (the traces)

we can take two LTSs to check whether they are bisimilar or not

Again on LTS

let us have a closer look at LTS

states are anonymous

$T = (\mathcal{S}, \longrightarrow, \mathcal{L}, s_0)$ where

- $\mathcal{S}$ are the states
- $\mathcal{L}$ are the labels
- $\longrightarrow \subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}$
- s_0 is the initial state

Operations on LTS: restriction

$T = (\mathcal{S}, \longrightarrow, \mathcal{L}, s_0)$

the restriction of T to $\mathcal{L}' \subseteq \mathcal{L}$ is

$T \restriction \mathcal{L}' = (\mathcal{S}, \longrightarrow', \mathcal{L}', s_0)$ where $\longrightarrow' = \{(s, \alpha, s') \mid s \xrightarrow{\alpha} s \textit{ and } \alpha \in \mathcal{L}'\}$

prune the transition relation

Example

$$\begin{tikzcd}
s_0 \arrow[r, "\text{up}"'] & s_1 \arrow[l, bend right, "\text{down}"'] \arrow[r, "\text{up}"'] & s_2 \arrow[l, bend right, "\text{down}"'] & \cdots
\end{tikzcd}$$

restrict down

$$\begin{tikzcd}
s_0 \arrow[r, "\text{up}"'] & s_1 \arrow[r, "\text{up}"'] & s_2 & \cdots
\end{tikzcd}$$

Operations on LTS: relabelling

$T = (\mathcal{S}, \longrightarrow, \mathcal{L}, s_0)$

the relabeling of T to $f : \mathcal{L} \to \mathcal{L}'$ is

$T[f] = (\mathcal{S}, \longrightarrow', \mathcal{L}', s_0)$ where $\longrightarrow' = \{(a, f(\alpha), s') \mid s \xrightarrow{\alpha} s\}$

rename the transition relation

Example

$$\begin{tikzcd}
s_0 \arrow[r, "\text{up}"'] & s_1 \arrow[l, bend right, "\text{down}"'] \arrow[r, "\text{up}"'] & s_2 \arrow[l, bend right, "\text{down}"'] & \cdots
\end{tikzcd}$$

rename up with inc and down with dec

$$\begin{tikzcd}
s_0 \arrow[r, "\text{ing}"'] & s_1 \arrow[l, bend right, "\text{dec}"'] \arrow[r, "\text{inc}"'] & s_2 \arrow[l, bend right, "\text{dec}"'] & \cdots
\end{tikzcd}$$

Operations on LTS: product

$T = (\mathcal{S}, \longrightarrow, \mathcal{L}, s_0)$ and $T' = (\mathcal{S}', \longrightarrow', \mathcal{L}', s_0')$

$$T \times T' = (\mathcal{S} \times \mathcal{S}', \longrightarrow'', \mathcal{L} \times_* \mathcal{L}', (s_0, s_0'))$$

where

$\mathcal{L} \times_* \mathcal{L}' = \mathcal{L} \cup \{*\} \times \mathcal{L}' \cup \{*\}$

$(s_1, s_1') \xrightarrow{(a,b)}'' (s_2, s_2')$ iff $s_1 \xrightarrow{a} s_2$ and $s_1' \xrightarrow{b}' s_2'$ with the convention that $s \xrightarrow{*} s$ (and $s \xrightarrow{*}' s$)

the idea is that $(a, \overline{a})$ is τ

Example

$$\begin{tikzcd} s_0 \arrow[d, "a"'] & s'_0 \arrow[d, "\overline{a}"] \\ s_1 & s'_1 \end{tikzcd}$$

$$\begin{tikzcd} & (s_0, s'_0) \arrow[dl, "a"'] \arrow[dd, "\tau"] \arrow[dr, "\overline{a}"] & \\ (s_1, s'_0) \arrow[dr, "\overline{a}"'] & & (s_0, s'_1) \arrow[dl, "a"] \\ & (s_1, s'_1) & \end{tikzcd}$$

τ is $(a, \overline{a})$

Operations on LTS: sum

$T = (\mathcal{S}, \longrightarrow, \mathcal{L}, s_0)$ and $T' = (\mathcal{S}', \longrightarrow', \mathcal{L}', s_0')$

$$T + T' = (\mathcal{S} \uplus \mathcal{S}' \cup \{(s_0, s_0')\}, \longrightarrow'', \mathcal{L} \uplus \mathcal{L}', (s_0, s_0'))$$

where

$s \xrightarrow{a}'' s'$ iff either $s, s' \in \mathcal{S}$ and there exists $s \xrightarrow{a} s$ or $s, s' \in \mathcal{S}'$ and there exists $s' \xrightarrow{b}' s'$ or if $s = (s_0, s_0')$ then there exists $s_0 \xrightarrow{a} s'$ (hence $s' \in \mathcal{S}$) or there exists $s_0' \xrightarrow{b}' s'$ (hence $s' \in \mathcal{S}'$)

Example

$$\begin{tikzcd}
s_0 \arrow[d, "a"'] & & s_0' \arrow[d, "b"] \\
s_1 & & s_1'
\end{tikzcd}$$

$$\begin{tikzcd}
 & (s_0, s_0') \arrow[dl, "a"'] \arrow[dr, "b"] & \\
s_1 & & s_1'
\end{tikzcd}$$

Operations on LTS: prefix

$T = (\mathcal{S}, \longrightarrow, \mathcal{L}, s_0)$

$$\alpha.T = (\mathcal{S} \cup \{s_{in}\}, \longrightarrow', \mathcal{L} \cup \{\alpha\}, s_{in})$$

where

$s_{in} \xrightarrow{\alpha}' s_0$ and $s \xrightarrow{\beta}' s'$ iff $s \xrightarrow{\beta} s$

Example

$$T = \quad s_0 \xrightarrow{b} s_1$$

$$a.T = \quad s \xrightarrow{a} s_0 \xrightarrow{b} s_1$$

An ordering on LTS

$T = (\mathcal{S}, \longrightarrow, \mathcal{L}, s_0)$ precedes $T' = (\mathcal{S}', \longrightarrow', \mathcal{L}', s_0')$ written $T \trianglelefteq T'$

- $\mathcal{S} \subseteq \mathcal{S}'$
- $s_0 = s_0'$
- $\mathcal{L} \subseteq \mathcal{L}'$
- $\longrightarrow \subseteq \longrightarrow'$

it is a partial order

if we take an ω-chain $T_0 \trianglelefteq T_1 \trianglelefteq T_2 \trianglelefteq \cdots \trianglelefteq T_n \cdots$ the least upper bound is

$$T = (\bigcup_{n\in\omega} \mathcal{S}_n, \bigcup_{n\in\omega} \longrightarrow_n, \bigcup_{n\in\omega} \mathcal{L}_n, s_0)$$

Operations on LTS

the operations relabelling, restriction, product, sum, prefixing are continuous operation (and their composition is)

the bottom is the LTS $T = (\{s\}, \emptyset, \emptyset, s)$

F is the continuous operator that define the actions we have to take

we have to calculate the fix-point of the operator F

with Tran we indicate the LTS

How do we use the operations?

we define a denotational semantics

we need an environment $\delta : \mathit{Var} \to \mathrm{Tran}$

Semantics

Consider again the process algebra

$$P \ ::= \ \mathbf{0} \mid \alpha.P \mid P + Q \mid P \parallel Q \mid P \setminus L \mid P[f] \mid X \mid \mathrm{rec}\ X\ P$$

the denotations are

$$\mathbf{T}[\![0]\!]\delta = (\{s\}, \emptyset, \emptyset, s) \qquad \mathbf{T}[\![P + Q]\!]\delta = \mathbf{T}[\![P]\!]\delta + \mathbf{T}[\![Q]\!]\delta$$

$$\mathbf{T}[\![\alpha.P]\!]\delta = \alpha.\mathbf{T}[\![P]\!]\delta \qquad \mathbf{T}[\![P \parallel Q]\!]\delta = \mathbf{T}[\![P]\!]\delta \times \mathbf{T}[\![Q]\!]\delta$$

$$\mathbf{T}[\![P \setminus L]\!]\delta = \mathbf{T}[\![P]\!]\delta \restriction L \qquad \mathbf{T}[\![X]\!]\delta = \delta(X)$$

$$\mathbf{T}[\![P[f]]\!]\delta = \mathbf{T}[\![P]\!]\delta[f] \qquad \mathbf{T}[\![\mathrm{rec}\ X\ P]\!]\delta = \mathit{fix}(\mathbf{T}[\![P]\!]\delta[X/T])$$

What we get?

the LTSs we construct are acyclic

they are almost tree

if we unfold them we get a synchronisations tree

we do not have the possibility of observing concurrency

A bit more on traces

up to now just sequences of actions σ

we know that some actions are independent (concurrent)

the rather simple idea is that two independent actions can be switched

$a.\mathbf{0} \parallel b.\mathbf{0}$

$$\begin{tikzcd}
 & s_1 \arrow[dr, "b"] & \\
s_0 \arrow[ur, "a"] \arrow[dr, "b"'] & & s_3 \\
 & s_2 \arrow[ur, "a"'] &
\end{tikzcd}$$

the two maximal traces are ab and ba, but a and b are independent and we would be able to point out this

Mazurkiewicz's traces

$(M, \mathcal{L}, I)$ where

- $\mathcal{L}$ is a set (actions)
- $I \subseteq \mathcal{L} \times \mathcal{L}$ is an irreflexive and symmetric relation (the independence relation)
- $M \subseteq \mathcal{L}^*$ such that
  - $\sigma a \in M \Rightarrow \sigma \in M$ for all $\sigma \in \mathcal{L}^*$ and $a \in \mathcal{L}$ (prefix closed)
  - $\sigma ab\sigma' \in M$ and $a\ I\ b$ the $\sigma ba\sigma' \in M$ for all $\sigma, \sigma' \in \mathcal{L}^*$ and $a, b \in \mathcal{L}$ (I-closed)
  - $\sigma a \in M$, $\sigma b \in M$ and $a\ I\ b$ then $\sigma ab \in M$ for all for all $\sigma \in \mathcal{L}^*$ and $a, b \in \mathcal{L}$

Mazurkiewicz's traces

$a.\mathbf{0} \parallel b.\mathbf{0}$

$$\begin{tikzcd}
 & s_1 \arrow[dr, "b"] & \\
s_0 \arrow[ur, "a"] \arrow[dr, "b"'] & & s_3 \\
 & s_2 \arrow[ur, "a"'] &
\end{tikzcd}$$

Mazurkiewicz's traces

$a.\mathbf{0} \parallel b.\mathbf{0}$

$$\begin{tikzcd}
 & s_1 \arrow[dr, "b"] & \\
s_0 \arrow[ur, "a"] \arrow[dr, "b"'] & & s_3 \\
 & s_2 \arrow[ur, "a"'] &
\end{tikzcd}$$

language: $\{a, b, ab, ba\}$

Mazurkiewicz's traces

$a.\mathbf{0} \parallel b.\mathbf{0}$

$$\begin{tikzcd}
 & s_1 \arrow[dr, "b"] & \\
s_0 \arrow[ur, "a"] \arrow[dr, "b"'] & & s_3 \\
 & s_2 \arrow[ur, "a"'] &
\end{tikzcd}$$

Mazurkiewicz's trace: $(\{a, b, ab, ba\}, \{a, b\}, a\ I\ b)$

Mazurkiewicz's traces

$a.\mathbf{0} \parallel b.\mathbf{0}$

$$\begin{tikzcd}
 & s_1 \arrow[dr, "b"] & \\
s_0 \arrow[ur, "a"] \arrow[dr, "b"'] & & s_3 \\
 & s_2 \arrow[ur, "a"'] &
\end{tikzcd}$$

Mazurkiewicz's trace: $(\{a, b, ab, ba\}, \{a, b\}, a\ I\ b)$

the point is that from the executions a and b, knowing that a and b are independent, we get ab and ba

Mazurkiewicz's traces

$a.\mathbf{0} \parallel b.\mathbf{0}$

$$\begin{tikzcd}
 & s_1 \arrow[dr, "b"] & \\
s_0 \arrow[ur, "a"] \arrow[dr, "b"'] & & s_3 \\
 & s_2 \arrow[ur, "a"'] &
\end{tikzcd}$$

Mazurkiewicz's trace: $(\{a, b, ab, ba\}, \{a, b\}, a\ I\ b)$

the point is that from the executions a and b, knowing that a and b are independent, we get ab and ba

define and equivalence $\sigma ab\sigma' \simeq \sigma ba\sigma'$ when $a\ I\ b$, and a trace σ is now an equivalence class $[\sigma]_{\simeq}$

Mazurkiewicz's traces

$a.\mathbf{0} \parallel b.\mathbf{0}$

$$\begin{tikzcd}
 & s_1 \arrow[dr, "b"] & \\
s_0 \arrow[ur, "a"] \arrow[dr, "b"'] & & s_3 \\
 & s_2 \arrow[ur, "a"'] &
\end{tikzcd}$$

Mazurkiewicz's trace: $(\{a, b, ab, ba\}, \{a, b\}, a\ I\ b)$

the point is that from the executions a and b, knowing that a and b are independent, we get ab and ba

define and equivalence $\sigma ab\sigma' \simeq \sigma ba\sigma'$ when $a\ I\ b$, and a trace σ is now an equivalence class $[\sigma]_\simeq$

traces languages generalise languages (in the latter the independence relation is empty)

Recall CCS again

presented in a slightly different way to avoid recursion

$$\begin{array}{lll} \text{(Actions)} & \alpha & ::= a \mid \overline{a} \mid \tau \\ \text{(CCS Processes)} & P, Q & ::=^{\text{co}} \sum_{i \in I} \alpha_i.P_i \mid (P \parallel Q) \mid P \backslash a \end{array}$$

we omit relabelling and the sum is generalised, but it is essentially the same

the terms are defined coinductively

The SOS rules

$$\frac{z \in I}{\sum_{i \in I} \alpha_i.P_i \xrightarrow{\alpha_z} P_z} \ (\textsc{act})$$

$$\frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q} \ (\textsc{par-l}) \qquad \frac{Q \xrightarrow{\alpha} Q'}{P \parallel Q \xrightarrow{\alpha} P \parallel Q'} \ (\textsc{par-r})$$

$$\frac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\bar{\alpha}} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} \ (\textsc{syn}) \qquad \frac{P \xrightarrow{\alpha} P' \qquad \alpha \notin \{a, \bar{a}\}}{P \backslash a \xrightarrow{\alpha} P' \backslash a} \ (\textsc{r-res})$$

we obtain the usual LTS

Reversible CCS

we need to know what has been done

$$\begin{array}{llll}
\text{(CCS Processes)} & P, Q & ::=^{\text{co}} & \sum_{i \in I} \alpha_i.P_i \mid (P \parallel Q) \mid P \backslash a \\
\text{(RCCS Processes)} & R, S & ::= & m \triangleright P \mid (R \parallel S) \mid R \backslash a \\
\text{(Memories)} & m & ::= & \langle *, \alpha, Q \rangle \cdot m \mid \langle m, \alpha, Q \rangle \cdot m' \\
& & & \quad \mid \langle 1 \rangle \cdot m \mid \langle 2 \rangle \cdot m \mid \langle \rangle
\end{array}$$

Some rules we need

$$\begin{array}{lll}
(\textsc{split}) & m \triangleright (P \parallel Q) \equiv \langle 1 \rangle \cdot m \triangleright P \parallel \langle 2 \rangle \cdot m \triangleright Q & \\
(\textsc{res}) & m \triangleright P\backslash a \equiv (m \triangleright P)\backslash a & \text{if } a \notin \mathtt{fn}(m) \\
(\alpha) & R \equiv S \qquad \text{if } R =_\alpha S &
\end{array}$$

Forward SOS rules

$$\frac{}{m \triangleright \sum_{i \in I} \alpha_i . P_i \xrightarrow{m:\alpha_z} \langle *, \alpha_z^z, \sum_{i \in I \setminus \{z\}} \alpha_i . P_i \rangle \cdot m \triangleright P_z}$$

$$\frac{R \xrightarrow{m:\alpha} R'}{R \parallel S \xrightarrow{m:\alpha} R' \parallel S} \qquad \frac{S \xrightarrow{m:\alpha} S'}{R \parallel S \xrightarrow{m:\alpha} R \parallel S'}$$

$$\frac{R \xrightarrow{m_1:\alpha} R' \qquad S \xrightarrow{m_2:\bar{\alpha}} S'}{R \parallel S \xrightarrow{m_1,m_2:\tau} R'_{m_2 @ m_1} \parallel S'_{m_1 @ m_2}}$$

$$\frac{R \xrightarrow{m:\alpha} R' \qquad \alpha \notin \{a, \bar{a}\}}{R \setminus a \xrightarrow{m:\alpha} R' \setminus a} \qquad \frac{R \equiv R' \quad R' \xrightarrow{m:\alpha} S' \quad S' \equiv S}{R \xrightarrow{m:\alpha} S}$$

Backward SOS rules

$$\frac{}{\langle *, \alpha_z^z, \sum\limits_{i\in I\setminus\{z\}} \alpha_i.P_i\rangle \cdot m \triangleright P_z \overset{m:\alpha_z}{\rightsquigarrow} m \triangleright \sum\limits_{i\in I} \alpha_i.P_i}$$

$$\frac{R \overset{m:\alpha}{\rightsquigarrow} R'}{R \parallel S \overset{m:\alpha}{\rightsquigarrow} R' \parallel S} \qquad \frac{S \overset{m:\alpha}{\rightsquigarrow} S'}{R \parallel S \overset{m:\alpha}{\rightsquigarrow} R \parallel S'}$$

$$\frac{R \overset{m_1:\alpha}{\rightsquigarrow} R' \qquad S \overset{m_2:\bar{\alpha}}{\rightsquigarrow} S'}{R_{m_2@m_1} \parallel S_{m_1@m_2} \overset{m_1,m_2:\tau}{\rightsquigarrow} R' \parallel S'}$$

$$\frac{R \overset{m:\alpha}{\rightsquigarrow} R' \qquad \alpha \notin \{a, \bar{a}\}}{R\backslash a \overset{m:\alpha}{\rightsquigarrow} R'\backslash a} \qquad \frac{R \equiv R' \quad R' \overset{m:\alpha}{\rightsquigarrow} S' \quad S' \equiv S}{R \overset{m:\alpha}{\rightsquigarrow} S}$$

Sum up

We have defined two relations: $\rightarrow$ (for forward moves) and $\rightsquigarrow$ (for backward moves)

the LTS is $(\mathcal{Q}, \rightarrow \cup \rightsquigarrow, \mathcal{A}ct, R)$

An RCCS process of the form $\langle\rangle \triangleright P$ is referred to as initial

Example

$$P = a.(b \parallel c) \parallel (\overline{a} \parallel d)$$

$$\langle\rangle \triangleright P \equiv \big(\langle 0\rangle \cdot \langle\rangle \triangleright a.(b \parallel c)\big) \parallel \big(\langle 1\rangle \cdot \langle\rangle \triangleright (\overline{a} \parallel d)\big) = R_1$$

$$R_1 \equiv \langle 0\rangle \cdot \langle\rangle \triangleright a.(b \parallel c) \parallel \langle 0\rangle \cdot \langle 1\rangle \cdot \langle\rangle \triangleright \overline{a} \parallel \langle 1\rangle \cdot \langle 1\rangle \cdot \langle\rangle \triangleright d = R_2$$

$$R_2 \xrightarrow{m_1, m_2 : \tau} \langle m_1, a, \mathbf{0}\rangle \cdot \langle 0\rangle \cdot \langle\rangle \triangleright (b \parallel c) \parallel \big(\langle m_2, \overline{a}, \mathbf{0}\rangle \cdot \langle 0\rangle \cdot \langle 1\rangle \cdot \langle\rangle \triangleright \mathbf{0}\big) \parallel \big(\langle 1\rangle \cdot \langle 1\rangle \cdot \langle\rangle \triangleright d\big) = R_3$$

$$\begin{aligned} R_3 \equiv\ & \big(\langle 0\rangle \cdot \langle m_1, a, \mathbf{0}\rangle \cdot \langle 0\rangle \cdot \langle\rangle \triangleright b\big) \parallel \big(\langle 1\rangle \cdot \langle m_1, a, \mathbf{0}\rangle \cdot \langle 0\rangle \cdot \langle\rangle \triangleright c\big) \parallel \\ & \big(\langle m_2, \overline{a}, \mathbf{0}\rangle \cdot \langle 0\rangle \cdot \langle 1\rangle \cdot \langle\rangle \triangleright \mathbf{0}\big) \parallel \big(\langle 1\rangle \cdot \langle 1\rangle \cdot \langle\rangle \triangleright d\big) \end{aligned}$$

where $m_1 = \langle 0\rangle \cdot \langle\rangle$ and $m_2 = \langle 1\rangle \cdot \langle\rangle$.

Event Structures

Event Structures

A quite general definition:

an event structure describes sequences of events that can be triggered by combinations of other events, with certain forbidden combinations of events

events are regarded as atomic actions

The main notion will be the one of configuration

General Event Structures

$\mathcal{E} = (E, \mathit{Con}, \vdash)$ where

- E is a set of events
- $\mathit{Con} \subseteq \mathbf{2}^{E}_{\mathit{fin}}$ is a consistency predicate such that if $X \in \mathit{Con}$ and $Y \subseteq X$ then $Y \in \mathit{Con}$
- $\vdash \subseteq \mathit{Con} \times E$ is an entail relation such that if $X \vdash e$ and $X \subseteq Y$ then $Y \vdash e$

The consistency predicate tells us which events can stay together, and the entail gives us a kind of deduction: establishing what is needed to add an event

Configurations of General Event Structures

$\mathcal{E} = (E, Con, \vdash)$, take $x \subseteq E$

- $\forall X \subseteq_{fin} x$ it holds that $X \in Con$ (consistent)
- $\forall e \in x.\ \exists e_0, \ldots, e_n \in x.\ e_n = e\ and\ \forall i \leq n,\ \{e_0, \ldots, e_{i_1}\} \vdash e_i$ (secured)

the idea is that one can extend the configuration adding one event

configurations are ordered by inclusion

Nondeterminism or Conflict

$$\mathcal{E} = (\{e, e'\}, \{\emptyset, \{e\}, \{e'\}\}, \emptyset \vdash e, \emptyset \vdash e', \{e\} \vdash e, \{e'\} \vdash e')$$

has the following configurations: $\emptyset, \{e\}$ and $\{e'\}$

e and e' are in conflict

have the same proper entailment, nondeterminism

Concurrency

$$
\begin{aligned}
\mathcal{E} \;=\; & (\{e, e'\}, \{\emptyset, \{e\}, \{e'\}, \{e, e'\}\}, \emptyset \vdash e, \emptyset \vdash e', \{e\} \vdash e, \\
& \{e'\} \vdash e', \{e\} \vdash e', \{e'\} \vdash e, \{e, e'\} \vdash e, \{e, e'\} \vdash e')
\end{aligned}
$$

has the following configurations: $\emptyset, \{e\}$, $\{e'\}$ and $\{e, e'\}$

e and e' are in concurrent

Kind of causality

$$\begin{aligned}\mathcal{E} \;=\; & (\{e, e', e''\}, \{\emptyset, \{e\}, \{e'\}, \{e''\}, \{e, e''\}, \{e', e''\}\} \\ & \emptyset \vdash e, \emptyset \vdash e', \{e\} \vdash e, \{e'\} \vdash e', \{e\} \vdash e'', \{e'\} \vdash e'', \dots)\end{aligned}$$

has the following configurations: $\emptyset, \{e\}$, $\{e'\}$, $\{e, e''\}$ and $\{e', e''\}$

e and e' are events and each of them causes e'', but not both together

which one? Observe that e'' cannot happen alone

Avoid ignoring the proper causes

stability:

$$X \vdash e \textit{ and } Y \vdash e \textit{ and } X \cup Y \cup \{e\} \in \mathit{Con} \textit{ implies } X \cap Y \vdash e$$

minimal set of causes for an event

$$\begin{aligned} \mathcal{E} \;=\; & (\{e, e', e''\}, \{\emptyset, \{e\}, \{e'\}, \{e''\}, \{e, e''\}, \{e', e''\}\} \\ & \emptyset \vdash e, \emptyset \vdash e', \{e\} \vdash e, \{e'\} \vdash e', \{e\} \vdash e'', \{e'\} \vdash e'', \ldots) \end{aligned}$$

it is not stable: $\{e\} \vdash e''$, $\{e'\} \vdash e''$ but we do not have $\{e, e', e''\}$
and we do not have $\{e\} \cap \{e'\} \vdash e''$

Prime Event Structures

the $\vdash$ is modelled via a partial order and the *Con* is induced by a binary irreflexive conflict relation

$$\mathcal{E} = (E, \leq, \#)$$

where

- E is a set of events
- $\leq \subseteq E \times E$ is a partial order (causality)
- $\# \subseteq E \times E$ is a symmetric and irreflexive relation (conflict) inherited along the causality relation: if $e \# e' \leq e''$ then $e \# e''$

Configurations

$\mathcal{E} = (E, \leq, \#)$

the consistency predicate is induced by the conflict relation: subsets of events $X \subseteq E$ conflict-free $\forall e, e' \in X.\ e \neq e'$ implies $\neg(e \# e')$

the causality relation gives the entailment: $X \vdash e$ implies that $\forall e' \leq e.\ e' \in X$

a configuration is a subset $x \subseteq E$ which is conflict-free (i.e. consistent) and such that for each $e \in x$ the set $\lfloor e \rfloor = \{e' \mid e' \leq e\} \subseteq x$ (i.e. secured)

x can be totally ordered coherently with $\leq$ and this gives,
$\forall e \in x.\ \exists e_0, \ldots, e_n \in x.\ e_n = e \textit{ and } \forall i \leq n,\ \{e_0, \ldots, e_{i_1}\} \vdash e_i$

Configurations

are ordered by inclusion

they form a finitary algebraic domain (a Scott domain)

they have a nice and useful structure

concurrency is around: you can say that two events are concurrent when they are

- nor in conclict
- nor in causally related

Bundle Event Structures

is or causality so bad?

$\mathcal{E} = (E, \mapsto, \#)$

the partial order is substituted with a new relation $\mapsto$ which has and has a more operational flavour

$\mapsto \subseteq \mathbf{2}^{E}_{fin} \times E$ is the enabling relation such that if $X \mapsto e$ then for all $e_1, e_2 \in X.\ e_1 \neq e_2$ implies $e_1 \# e_2$

for each $e \in E$ the set $\{X \subseteq E \mid X \mapsto e\}$ is finite

Configurations

a configuration x is a subset of events such that

- it is conflict free
- there exists a linearization $\{e_1, \ldots, e_n, \ldots\}$ of the events in x such that $\forall e_i$ and for all bundles $X_i \mapsto e_i$ it holds that $X_i \cap \{e_1, \ldots, e_{i-1}\} \neq \emptyset$

observe that bundles are subsets of events pairwise in conflict

being secured pop out again (it suggests the possible ordering in the executions)

Or-causality

the general event structure

$$\begin{aligned}\mathcal{E} \;=\; & (\{e, e', e''\}, \{\emptyset, \{e\}, \{e'\}, \{e''\}, \{e, e'\}, \{e, e''\}, \{e', e''\}\}, \\ & \emptyset \vdash e, \emptyset \vdash e', \{e\} \vdash e, \{e'\} \vdash e', \{e\} \vdash e', \\ & \{e'\} \vdash e, \{e\} \vdash e'', \{e'\} \vdash e'', \ldots)\end{aligned}$$

the bundle event structure

$$\mathcal{E} = (\{e, e', e''\}, \emptyset \mapsto e, \emptyset \mapsto e', \{e, e'\} \mapsto e'', e \# e')$$

Event Structures: conflicts

the conflict can depend on the ordering

e can be in conflict with e' only when e' happens before

asymmetric conflict

Asymmetric Event Structures

$\mathsf{E} = (E, <, \nearrow)$, where

- $< \subseteq E \times E$ is an irreflexive partial order, called *causality*, defined such that $\forall e \in E.\ \lfloor e \rfloor = \{e' \in E \mid e' \leq e\}$ is finite
- $\nearrow \subseteq E \times E$, called *weak causality*, is defined such that for all $e, e' \in E$
  - $e < e' \ \Rightarrow\ e \nearrow e'$
  - $\nearrow \cap (\lfloor e \rfloor \times \lfloor e \rfloor)$ is acyclic
  - if $e \# e'$ and $e' < e''$ then $e \# e''$, where $e \# e'$ iff $e \nearrow e'$ and $e' \nearrow e$

weak causality establishes an order, stating which event can happen before

Configurations

$\mathcal{E} = (E, <, \nearrow)$

a configuration $x \subseteq E$ is such that

- $\nearrow$ is well-founded on x
- $\forall e \in x.\ \lfloor e \rfloor = \{e' \mid e' < e\} \subseteq x$
- $\forall e \in x$ the set $\{e' \in x \mid e' \nearrow e\}$ is finite

well-founded implies conflict-free

$x \sqsubseteq y$ (y extends x) if $x \subseteq y$ and $\neg(e' \nearrow e)$ for all $e \in x$ and $e' \in y \setminus x$

configurations are partially ordered by $\sqsubseteq$

Example

$$\mathcal{E} = (\{e, e', e''\}, e < e'', e' < e'', e \nearrow e', e \nearrow e'', e' \nearrow e'')$$

the configurations are $\emptyset, \{e\}, \{e'\}, \{e, e'\}$ and $\{e, e', e''\}$

$$\begin{tikzcd} & \{e'\} \arrow[r] & \{e, e'\} \arrow[d] \\ \emptyset \arrow[ur] \arrow[dr] & & \\ & \{e\} & \{e, e', e''\} \end{tikzcd}$$

you cannot go from $\{e\}$ to $\{e, e'\}$ because of $e \nearrow e'$

Flow Event Structures

$E = (E, \prec, \#)$, where

- E is a set of events
- $\prec \subseteq E \times E$ is an irreflexive relation called the flow relation
- $\# \subseteq E \times E$ is a symmetric conflict relation

The flow relation is the one representing the dependencies among events

the conflict relation is required to be just symmetric

Configurations

$E = (E, \prec, \#)$

$x \subseteq E$ is a configuration

- conflict-free
- for each $e \in x$, for each $e' \prec e$ either $e' \in x$ or there exists $e'' \in x$ such that $e'' \prec e$ and $e' \# e''$
- $\prec^*$ restricted to x is a partial order

Other relevant causality/conflict notions?

an event can be inhibited for a while but not forever

the event e is disabled by e' and re-enabled by e''

the conflicts may have a kind of duration

Introducing a new relation

we introduce a new ternary relation $\vdash\!\circ \subseteq \mathbf{2}^E_1 \times E \times \mathbf{2}^E$

- E is a set of events
- $\vdash\!\circ \subseteq \mathbf{2}^E_1 \times E \times \mathbf{2}^E$ is a called disabling-enabling relation and the intuition is:
    - if $\vdash\!\circ(\{e'\}, e, A)$ then the event e' inhibits the event e
    - e can be enabled again by one of the events in A

causality: $\vdash\!\circ(\emptyset, e, A)$ means e depends on one of the events in A

the conflicts implies that the first component is not empty:
$\vdash\!\circ(\{e'\}, e, \emptyset)$ implies that e' cannot happen after e

Causality and conflicts

we define a general notion of causality ($<$), weak causality ($\nearrow$) and conflict ($\#$)

$\#_p A$ means that the events in A are pairwise in conflict (symmetric)

$$\frac{\vdash\!\circ(\emptyset, e, A) \qquad \#_p A}{A < e}(< 1) \qquad \frac{A < e \qquad \forall e' \in A.\ A_{e'} < e' \qquad \#_p(\cup\{A_{e'} \mid e' \in A\}}{(\cup\{A_{e'} \mid e' \in A\}) < e}(< 2)$$

$$\frac{\vdash\!\circ(\{e'\}, e, \emptyset)}{e \nearrow e'}(\nearrow 1) \qquad \frac{e \in A < e'}{e \nearrow e'}(\nearrow 2) \qquad \frac{\#\{e, e'\}}{e \nearrow e'}(\nearrow 3)$$

$$\frac{e_0 \nearrow \ldots \nearrow e_n \nearrow e_0}{\#\{e_0, \ldots, e_n\}}(\#1) \qquad \frac{A' < e \qquad \forall e' \in A'.\ \#(A \cup \{e'\})}{\#(A \cup \{e\})}(\#2)$$

Inhibitor Event Structures

$\mathcal{E} = (E, \multimap)$ where, for all $e \in E$, $a \in \mathbf{2}^E_1$ and $A \subseteq E$

- if $\multimap(a, e, A)$ then $\#_p A$ and $\forall e' \in a.\ \forall e'' \in A.\ e' < e''$
- if $A < e$ then $\multimap(\emptyset, e, A)$
- if $e \nearrow e'$ then $\multimap(\{e'\}, e, \emptyset)$

the relation $\multimap$ alone suffices

Configurations

there is a characterisation similar to the ones in prime or asymmetric event structures (omitted here)

$x \subseteq E$ must be acyclic with respect to $\nearrow$

an event e can be added to x if $\forall \multimap(a, e, A)$ it holds that if $a \subseteq x$ then $A \cap x \neq \emptyset$

x is secured with respect the entailment defined by the relation $\vdash\!\circ$

Example

$\mathcal{E} = (\{e, e', e''\}, \vdash\!\!\circ(\emptyset, e, \emptyset), \vdash\!\!\circ(\emptyset, e', \{e\}), \vdash\!\!\circ(\{e\}, e'', \{e'\}))$

configurations are $\emptyset, \{e\}, \{e''\}\{e, e''\}, \{e, e'\}$ and $\{e, e', e''\}$

the only one missing is $\{e', e''\}$ as e' depends on e

the ordering is such that from $\{e\}$ one cannot add e'' but from $\{e, e'\}$ the event e'' can be added

e'' is inhibited by e and re-enabled by e'

enabling is not monotonic

Comparing Event Structures

we can see when it is possible to encode one into another

each prime event structure is a bundle one: $e \leq e'$ gives $\{e\} \mapsto e'$ but not the vice versa

in prime event structure the or-causality is forbidden

bundle can model or-causality

each prime even structure is also an asymmetric one (the conflict is modelled with the weak causality)

Other kind of causality?

Causality pattern in service oriented computations may vary depending on conditions

to reason on these computations we need to model them

Shrinking causality

Shrinking event structure: some causal relation may be dropped

$\gamma = (E, \rightarrow, \#, \lhd)$ where

- $(E, \rightarrow, \#)$ is a prime event structure and
- $\lhd \subseteq E \times E \times E$ is the shrinking relation s.t.
  $\forall e, e', e'' \in E.\ e' \lhd [e \rightarrow e''] \implies e \rightarrow e''$

Shrinking causality

Shrinking event structure: some causal relation may be dropped

$\gamma = (E, \rightarrow, \#, \triangleleft)$ where

- $(E, \rightarrow, \#)$ is a prime event structure and
- $\triangleleft \subseteq E \times E \times E$ is the shrinking relation s.t.
  $\forall e, e', e'' \in E.\ e' \triangleleft [e \rightarrow e''] \implies e \rightarrow e''$

only existing dependencies can be dropped

dropped cause (w.r.to H): $\mathsf{dc}(H, e) = \{e' \mid \exists d \in H.\ d \triangleleft [e' \rightarrow e]\}$

trace: sequence of events $\rho = e_1, \ldots, e_n$ such that

- $\overline{\rho} = \{e_1, \ldots, e_n\}$ is conflict free
- for each $i \leq n$. $\mathsf{ic}(e_i) \setminus \mathsf{dc}(\overline{\rho_{i-1}}, e_i) \subseteq \overline{\rho_{i-1}}$

Shrinking causality

Shrinking event structure: some causal relation may be dropped

$\gamma = (E, \rightarrow, \#, \lhd)$ where

- $(E, \rightarrow, \#)$ is a prime event structure and
- $\lhd \subseteq E \times E \times E$ is the shrinking relation s.t.
  $\forall e, e', e'' \in E.\ e' \lhd [e \rightarrow e''] \implies e \rightarrow e''$

$$[b \rightarrow c] \ \lhd \ a$$

traces: b, bc, a, ab, ac, abc

in ac the dependency of c from b is dropped by a

Growing causality

Growing event structure: some causal relation may be added

$\gamma = (E, \rightarrow, \#, \blacktriangleright)$ where

- $(E, \rightarrow, \#)$ is a PES and
- $\blacktriangleright \subseteq E \times E \times E$ is the growing relation s.t.
  $\forall e, e', e'' \in E.\ e' \blacktriangleright [e \rightarrow e''] \implies \neg(e \rightarrow e'')$

Growing causality

Growing event structure: some causal relation may be added

$\gamma = (E, \rightarrow, \#, \blacktriangleright)$ where

- $(E, \rightarrow, \#)$ is a PES and
- $\blacktriangleright \subseteq E \times E \times E$ is the growing relation s.t. $\forall e, e', e'' \in E.\ e' \blacktriangleright [e \rightarrow e''] \Longrightarrow \neg(e \rightarrow e'')$

only non existing dependencies can be added

growing causes (w.r.to H) ($\mathrm{ac}(H, e)$): $\{e' \mid \exists d \in H.\ d \blacktriangleright [e' \rightarrow e]\}$

trace: sequence of events $\rho = e_1, \ldots, e_n$ such that

- $\overline{\rho} = \{e_1, \ldots, e_n\}$ is conflict free
- for each $i \leq n$. $\mathrm{ic}(e_i) \cup \mathrm{ac}(\overline{\rho_{i-1}}, e_i) \subseteq \overline{\rho_{i-1}}$

Growing causality

Growing event structure: some causal relation may be added

$\gamma = (E, \rightarrow, \#, \blacktriangleright)$ where

- $(E, \rightarrow, \#)$ is a PES and
- $\blacktriangleright \subseteq E \times E \times E$ is the growing relation s.t. $\forall e, e', e'' \in E.\ e' \blacktriangleright [e \rightarrow e''] \implies \neg(e \rightarrow e'')$

$$d \blacktriangleright [e \rightarrow f]$$

traces: $e, f, d, de, ef, def, edf$

in def and edf the dependency of f from e is added by d

we do not have dfe

Dynamic Event Structure

with growing and shrinking causality:

$\Sigma = (E, \rightarrow, \#, \lhd, \blacktriangleright)$

- $(E, \rightarrow, \#, \lhd)$ is a shrinking event structure
- $(E, \rightarrow, \#, \blacktriangleright)$ is a growing event structure, and
- $\forall e, e' \in E.\ \nexists m, a \in E.\ a \blacktriangleright [e \rightarrow e'] \lhd m$

the dependency $e \rightarrow e'$ cannot be added and removed

trace: sequence of events $\rho = e_1, \ldots, e_n$ such that

- $\overline{\rho} = \{e_1, \ldots, e_n\}$ is conflict free
- for each $i \leq n$. $((\mathrm{ic}(e_i) \cup \mathrm{ac}(\overline{\rho_{i-1}}, e_i)) \setminus \mathrm{dc}(\overline{\rho_{i-1}}, e_i)) \subseteq \overline{\rho_{i-1}}$

Representing dependencies

Just some examples of the relations proposed in literature

Representing dependencies

Just some examples of the relations proposed in literature

Static:

Representing dependencies

Just some examples of the relations proposed in literature

Static:

- with a partial order $\leq$

Representing dependencies

Just some examples of the relations proposed in literature

Static:

- with a partial order $\leq$: $\quad e \leq e'$

Representing dependencies

Just some examples of the relations proposed in literature

Static:

- with a partial order $\leq$: $e \leq e'$
- with an enabling relation $\vdash$

Representing dependencies

Just some examples of the relations proposed in literature

Static:

- with a partial order $\leq$: $e \leq e'$
- with an enabling relation $\vdash$: $X \vdash Y$

Representing dependencies

Just some examples of the relations proposed in literature

Static:

- with a partial order $\leq$: $e \leq e'$
- with an enabling relation $\vdash$: $X \vdash Y$

Dynamic:

Representing dependencies

Just some examples of the relations proposed in literature

Static:

- with a partial order $\leq$: $e \leq e'$
- with an enabling relation $\vdash$: $X \vdash Y$

Dynamic:

- with an enabling/disabling relation $\vdash\!\!\multimap$

Representing dependencies

Just some examples of the relations proposed in literature

Static:

- with a partial order $\leq$: $e \leq e'$
- with an enabling relation $\vdash$: $X \vdash Y$

Dynamic:

- with an enabling/disabling relation $\vdash\!\!\!\multimap$: $\vdash\!\!\!\multimap(\{e\}, e', A)$

Representing dependencies

Just some examples of the relations proposed in literature

Static:

- with a partial order $\leq$: $e \leq e'$
- with an enabling relation $\vdash$: $X \vdash Y$

Dynamic:

- with an enabling/disabling relation $\vdash\!\!\circ$: $\vdash\!\!\circ(\{e\}, e', A)$
- with modifiers

Representing dependencies

Just some examples of the relations proposed in literature

Static:

- with a partial order $\leq$: $e \leq e'$
- with an enabling relation $\vdash$: $X \vdash Y$

Dynamic:

- with an enabling/disabling relation $\vdash\!\!\circ$: $\vdash\!\!\circ(\{e\}, e', A)$
- with modifiers: $e \blacktriangleright [e'' \to e']$ or $e \lhd [e'' \to e']$

The variety of approaches is driven by the variety of situations to be modeled

Configurations

a key concept is the one of configuration

a subset of events representing the state of the system

it contains all the information needed to establish when an event can be added

it can be used to compare various notions of dependency

the various relations representing dependencies act on configurations

Context

The idea is that an event can be added if there is a context which justifies this

We introduce a new relation $\gg \subseteq \mathbf{2}^{A} \times E$ where

- $A \subseteq \mathbf{2}^{E}_{fin} \times \mathbf{2}^{E}_{fin}$, and
- $Z \gg e$ is such that $Z \neq \emptyset$

the intuition is that if $(X, Y) \in Z$ then X is the context and Y is the authorizing part

we call this relation context dependent relation (CD-relation)

How to use the new relation

$Z = \{(X_1, Y_1), \ldots, (X_n, Y_n)\} \gg e$

$\textsc{Cxt}(Z) = \bigcup_{i=1}^{|Z|} X_i$

Let $C \subseteq E$ be a subset of events and $e \notin C$ an event

e is enabled at C ($C[e\rangle$) if for each $Z \gg e$ there is a pair $(X_i, Y_i) \in Z$ such that

- $\textsc{Cxt}(Z) \cap C = X_i$, and
- $Y_i \subseteq C$

Examples

Plain causality: $a \leq b$

$$\begin{tikzcd} \emptyset \arrow[r] & \{a\} \arrow[r] & \{a, b\} \end{tikzcd}$$

$\{(\emptyset, \emptyset)\} \gg a$

and

$\{(\emptyset, \{a\})\} \gg b$ or $\{(\{a\}, \emptyset)\} \gg b$

Examples

Or causality: $\{a, b\} \vdash c$

$$\begin{tikzcd} & \{a\} \arrow[r] & \{a, c\} \\ \emptyset \arrow[ur] \arrow[dr] & & \\ & \{b\} \arrow[r] & \{b, c\} \end{tikzcd}$$

$\{(\emptyset, \emptyset)\} \gg a$, $\qquad \{(\emptyset, \emptyset)\} \gg b$

and

$\{(\{a\}, \emptyset), (\{b\}, \emptyset)\} \gg c$

Examples

Growing causality: $a \blacktriangleright [b \to c]$

$$\begin{tikzcd}
 & \{a\} \arrow[r] & \{a, b\} \arrow[dr] & \\
\emptyset \arrow[ur] \arrow[r] \arrow[dr] & \{b\} \arrow[ur] \arrow[dr] & \{a, c\} \arrow[r] & \{a, b, c\} \\
 & \{c\} \arrow[ur] \arrow[r] & \{b, c\} \arrow[ur] &
\end{tikzcd}$$

$\{(\emptyset, \emptyset)\} \gg a$, $\{(\emptyset, \emptyset)\} \gg b$ and

$\{(\{a\}, \{b\}), (\emptyset, \emptyset)\} \gg c$

Examples

Shrinking causality: $a \lhd [b \to c]$

$$\begin{tikzcd}
 & \{a\} \arrow[r] \arrow[dr] & \{a, c\} \arrow[dr] & \\
\emptyset \arrow[ur] \arrow[dr] & & \{a, b\} \arrow[r] & \{a, b, c\} \\
 & \{b\} \arrow[r] \arrow[ur] & \{b, c\} \arrow[ur] &
\end{tikzcd}$$

$\{(\emptyset, \emptyset)\} \gg a,$ $\{(\emptyset, \emptyset)\} \gg b$ and

$\{(\{a\}, \emptyset), (\{b\}, \emptyset), (\{a, b\}, \emptyset)\} \gg c$

Examples

Resolving conflicts: $\emptyset \vdash X$, with $X \subseteq \{a, b, c\}$ and $X \neq \{a, b\}$ and $\{c\} \vdash \{a, b\}$

$$\begin{tikzcd}
 & \{a\} \arrow[r] & \{a, c\} \arrow[dr] & \\
\emptyset \arrow[ur] \arrow[r] \arrow[dr] & \{c\} \arrow[ur] \arrow[dr] & & \{a, b, c\} \\
 & \{b\} \arrow[r] & \{b, c\} \arrow[ur] &
\end{tikzcd}$$

$\{(\emptyset, \emptyset), (\{c\}, \emptyset), (\{b\}, \{c\})\} \gg a$, $\{(\emptyset, \emptyset), (\{c\}, \emptyset), (\{a\}, \{c\})\} \gg b$

and $\{(\emptyset, \emptyset)\} \gg c$

Examples

inhibition: $\vdash\!\!\!-\!\!\circ\ (\{a\}, c, \{b\})$

$$\begin{tikzcd}
 & \{a\} \arrow[r] & \{a,b\} \arrow[dr] & \\
\emptyset \arrow[ur] \arrow[dr] & & & \{a,b,c\} \\
 & \{c\} \arrow[r] & \{a,c\} \arrow[ur] &
\end{tikzcd}$$

$\{(\emptyset, \emptyset)\} \gg a,$ $\{(\emptyset, \{a\})\} \gg b$ and

$\{(\emptyset, \emptyset), (\{a\}, \{b\})\} \gg c$

Context-dependent event structure: cdes

$\mathcal{E} = (E, \#, \gg)$ where

- E is a set of *events*
- $\# \subseteq E \times E$ is an irreflexive and symmetric relation, called *conflict relation*
- $\gg \subseteq \mathbf{2}^{A} \times E$, where $A \subseteq \mathbf{2}^{E}_{fin} \times \mathbf{2}^{E}_{fin}$, is the *context-dependency* relation (CD-relation)

Each element of the CD-relation $\gg$ is called *entry*.

Configuration of a cdes

C be a subset of E is a configuration iff there exists a sequence of distinct events $\rho = e_1 \cdots e_n \cdots$ over E such that

- $\overline{\rho} = C$,
- $\overline{\rho}$ is conflict-free, and
- $\forall i.\ 1 \leq i \leq len(\rho).\ \overline{\rho}_{i-1}[e_i\rangle$

Some consideration on this new relation

It is faithful in representing the conditions under which an event happens

It is rather liberal

The same context may be represented in several ways

It can generalized easily

- enabling of subset of events,
- more structure in the contexts (corresponding to more structure in configurations)

Configuration of a μ event structure

the notion depends on the kind (μ) of event structure

we do not review them

we denote the set of configurations of an μ-event structure X with $\mathsf{Conf}_{\mu}(X)$

$\mapsto_{\mu} \subseteq \mathsf{Conf}_{\mu}(X) \times \mathsf{Conf}_{\mu}(X)$ is the relation describing how to reach a configuration from another

Comparing event structures

We use event automaton

$\mathsf{E} = \langle E, \mathsf{S}, \mapsto, s_0 \rangle$ such that

- $\mathsf{S} \subseteq \mathbf{2}^E$, and
- $\mapsto \subseteq \mathsf{S} \times \mathsf{S}$ is such that $s \mapsto s'$ implies that $s \subset s'$
- $s_0 \in \mathsf{S}$ is the initial state

states can be enriched with other information

Comparing event structures

for each kind of event structure we have

- configurations and
- a way to reach a configuration from another

these two ingredients form an event automaton:

Take X be an event structure of kind μ over the set of events E

$\mathcal{G}_\mu(X) = \langle E, \mathsf{Conf}_\mu(X), \mapsto_\mu, \emptyset \rangle$ is an event automaton

it is enough to compare the corresponding event automata

From event automata to cdes

Fix an event automaton $\mathsf{E} = \langle E, \mathsf{S}, \mapsto, s_0 \rangle$

$e \mathrel{\#_{\mathsf{ea}}} e'$ iff for each $s \in \mathsf{S}$. $\{e, e'\} \not\subseteq s$

$\mathcal{C}(\mathsf{E}, e) = \{s \in \mathsf{S} \mid s \cup \{e\} \in \mathsf{S} \ \wedge \ s \mapsto s \cup \{e\}\}$ are the allowing contexts

$\mathcal{I}(\mathsf{E}, e) = \{s \in \mathsf{S} \mid s \cup \{e\} \not\in \mathsf{S}\}$ are the negative contexts

$\mathsf{E} = \langle E, \mathsf{S}, \mapsto, s_0 \rangle$ be an ea. Then $\mathcal{F}_{\mathsf{ea}}(\mathsf{E}) = (E, \#, \gg)$ is a CDES, where for each $e \in E$ we have

$$\{(X, \emptyset) \mid X \in \mathcal{C}(\mathsf{E}, e)\} \cup \{(X, \{e\}) \mid X \in \mathcal{I}(\mathsf{E}, e)\} \gg e$$

Furthermore $\mathsf{E} \equiv \mathcal{G}_{\text{CDES}}(\mathcal{F}_{\mathsf{ea}}(\mathsf{E}))$.

Petri Nets

Petri nets

A net is the tuple (S, T, F, m_0) where

- S is a set of places and T is a set of transitions ($S \cap T = \emptyset$)
- $F = (F_{pre}, F_{post})$ is a pair of finitary multirelation from places to transitions (the flow relation... each arc has a weight)
- m_0 is a multiset over places, the initial marking

(S, T, F, m_0) may be seen as a bipartite graph (no arc connect a transition to a transition or a place to a place)

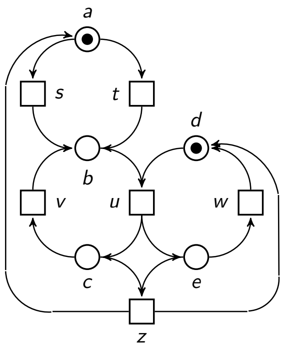
the multiset of transition A is enabled at the marking m if
$F_{pre}(-, A) \subseteq m$ A is the tensor of transitions

when the multiset A is fired at m a new marking is produced:
$m' = m - F_{pre}(-, A) + F_{post}(-, A)$ notation $m\,[A\rangle\, m'$

Petri nets

great success because of:

- graphical notation for nets:



- nice algebra (S and T invariants, siphons and traps) to prove properties of the modeled systems
- extendable (adding times, probabilities, firing under conditions...)

What were Petri's ideas

Petri's PhD thesis is entitled Kommunikation mit Automaten and (a part of) the abstract was:

> *This work is about the comprehensible foundations of a theory of communication...*

in the view of Petri nets as rewrite systems this aspect is somehow forgotten

In Petri's view a net is still the tuple we saw, but the focus is on

- distributed state (places reside on different spaces)
- independence of many activities: concurrency and locality of evolution
- exchange of informations (tokens) and synchronization

Petri focussed on physical aspects, not only *mathematical* ones

Petri nets and logic

like all kind of automata the relations are obvious...

but Petri was in search of this foundation theory

the theory must face dependencies, conflicts and concurrency among transitions

the executions of a net may be represented as a net itself (Petri's motto: the semantics of a net is a net)

the dependencies among events should lay on a line (chain)

the state of a system is what is produced by events on compatible lines: a maximal antichain

F should have values in $\{0, 1\}$, $F_{pre}(s, t)$ means there is an arc from s to t, $F_{post}(s, t)$ means there is an arc from t to s,

the net should be acyclic

Semantics as deduction: causal net

F is a relation (over $S \times T$ and $T \times S$)

${}^\bullet x = \{y \mid F(y,x)\}$ and $x^\bullet = \{y \mid F(x,y)\}$

A *causal net* $C = (B, E, F, m)$ is a safe net satisfying the following restrictions:

- $\forall b \in m,\ {}^\bullet b = \emptyset$,
- $\forall b \in B.\ \exists b' \in m$ such that $b' \leq_C b$,
- $\forall b \in B.\ {}^\bullet b$ is either empty or a singleton,
- for all $e \in E$ the set $\{e' \in E \mid e' \leq_C e\}$ is finite, and
- $\#$ is an irreflexive and symmetric relation defined as follows:
  - $e \#_r e'$ iff $e, e' \in E$, $e \neq e'$ and ${}^\bullet e \cap {}^\bullet e' \neq \emptyset$,
  - $x \# x'$ iff $\exists y, y' \in E$ such that $y \#_r y'$ and $y \leq_C x$ and $y' \leq_C x'$.

Semantics as deduction: causal net

An information system (IS) is a triple $(A, Con, \vdash)$ where $Con = \{X \subseteq A \mid X \text{ is finite}\}$ is a consistency predicate and $\vdash \subseteq Con \setminus \emptyset \times A$ is an entailment relation such that

- $X \subseteq Y \in Con$ implies $X \in Con$
- $\forall a \in A.\ \{a\} \in Con$
- $X \vdash a$ implies $X \cup \{a\} \in Con$
- $a \in X \in Con$ implies $X \vdash a$
- $X, Y \in Con$, $\forall b \in Y.\ X \vdash b$ and $Y \vdash c$ then $X \vdash c$

the elements of an IS are the subsets $\mathcal{A}$ of A that are finitely consistent (each finite subset of $\mathcal{A}$ is in Con) and deductively closed (if $X \vdash a$ and $X \subseteq \mathcal{A}$ then $a \in \mathcal{A}$)

Semantics as deduction: causal net

the elements of an information system ordered by inclusion are a Scott domain

continuous functions on Scott domains have a fixed point

it says how to effectively compute as approximations of increasing amount of information

Semantics as deduction: causal net

We start from a causal net $C = (B, E, F, m)$

Define: Con as $\{X \subseteq_{fin} E \mid \forall e, e' \in X \ \neg(e \# e')\}$

Define: $\lfloor e \rfloor = \{e' \in E \mid e' \leq_C e\}$

Define: $X \vdash' e$ iff $X \in Con$ and $\lfloor e \rfloor \subseteq X$ and saturate it adding $X \vdash e$ iff $e \in X$

We have that $(E, Con, \vdash)$ is an information system

Petri nets as models of logic

Petri nets explicitly manipulate resources

resource aware logics like linear logic have strong relationships with Petri nets

the idea is the usual one: tokens are propositions, transitions are deductions, and the structure obtained when a net correspond to a linear logic theory is that the causal net is not only a Scott domain but also a quantale

Priorities

The liberty in deciding which transitions may be executed at a given distributed state is reduced by imposing ordering in the firing of transitions

Priorities

The liberty in deciding which transitions may be executed at a given distributed state is reduced by imposing ordering in the firing of transitions

the implication "if $m\,[t\rangle\, m'$ and $m \subseteq m''$ then there exists a marking m''' such that $m''\,[t\rangle\, m'''$" does not hold in general as at m'' there may be another transition to execute first

Priorities

The liberty in deciding which transitions may be executed at a given distributed state is reduced by imposing ordering in the firing of transitions

the implication "if $m\,[t\rangle\, m'$ and $m \subseteq m''$ then there exists a marking m''' such that $m''\,[t\rangle\, m'''$" does not hold in general as at m'' there may be another transition to execute first

but to decide which transition I have to fire I have to check the entire space of possibilities: the locality of changes is somehow lost

Priorities and inhibitor arcs

The capability of checking for absence of tokens in given places add expressivity to Petri nets

Priorities and inhibitor arcs

The capability of checking for absence of tokens in given places add expressivity to Petri nets

A net is now (S, T, F, I, m_0) where $I \subseteq S \times T$ and the enabling at the marking m is changed: to fire t the place s such that $(s, t) \in I$ must be empty, i.e. $m(s) = 0$

Priorities and inhibitor arcs

The capability of checking for absence of tokens in given places add expressivity to Petri nets

A net is now (S, T, F, I, m_0) where $I \subseteq S \times T$ and the enabling at the marking m is changed: to fire t the place s such that $(s, t) \in I$ must be empty, i.e. $m(s) = 0$

the usual implication does not hold

Priorities and inhibitor arcs

The capability of checking for absence of tokens in given places add expressivity to Petri nets

A net is now (S, T, F, I, m_0) where $I \subseteq S \times T$ and the enabling at the marking m is changed: to fire t the place s such that $(s, t) \in I$ must be empty, i.e. $m(s) = 0$

the usual implication does not hold

with inhibitor arcs you may simulate a two counter machine

Reset arcs

usually the number of tokens consumed from a given place is fixed and independent from the marking

reset arcs have the capability of emptying the places they are connected to:

A net is now (S, T, F, R, m_0) where $R \subseteq S \times T$, the enabling at the marking m is not changed but the firing of t the place s such that $(s, t) \in R$ is emptied, i.e. $m'(s) = 0$, where m' is the reached marking

what is changed is the uniformity of changes that is relevant in Petri nets

reset arcs add expressivity to Petri nets

Lending arcs

transitions are executed only if the resources are available (in places)

relax this requirement: transitions may fire even if some resources are not available (in a controlled way)

resources may be taken on credit, and technically this is done creating debit tokens

A net is now (S, T, F, L, m_0) where $L \colon S \times T \to \mathbb{N}$, the enabling at the marking m is not changed but the firing of t causes changes also in the place s such that $L(s, t) > 0$, as if tokens are available, then they are used, otherwise the proper number of debit tokens is created in that place

when a token is produced in a place with a debit token, both are annihilated

Lending arcs

with lending arcs inhibitor arcs may be simulated

Lending Petri nets have been used as a model for a logic dealing with contracts

marking reachability equivalent to provability in that logic

Maximal firing rule

If all the enabled transitions (with the proper multiplicity) are executed together at a given marking then Petri nets are again Turing powerful

Recap

to increase expressivity either add new kind of arcs or modify dramatically the firing rule

new arcs: the kind of automata are different, firing rule the idea of distributed space is forgotten

Petri's idea of communicating automata is somehow lost

Apparently adding expressivity means departing from the initial intuitions

can we obtain some add expressivity without departing too much?

Back to languanges

- How to equip (reversible) process calculi with a truly concurrent semantics
- Usually the semantics is provided in term of labelled transitions systems
- In the 90ies a lot of work has been done to provide a non sequential semantics to process calculi
- we figure out to apply these works to the reversible setting

The 90ies

Various approach to give a truly concurrent semantics to CCS.
Mainly in terms of

- suitable nets (e.g. occurrence, flow, . . .)
- suitable event structures (e.g. prime, bundle, . . .)

often exploiting the relationships among these models

What about (reversible) (R)CCS

$$\begin{array}{lll}
\text{(Actions)} & \alpha & ::= a \mid \overline{a} \mid \tau \\
\text{(CCS Processes)} & P, Q & ::=^{\mathsf{co}} \sum_{i \in I} \alpha_i.P_i \mid (P \parallel Q) \mid P\backslash a
\end{array}$$

the coinductive definition gives us recursion

$$\begin{array}{lll}
\text{(RCCS Processes)} & R, S & ::= m \triangleright P \mid (R \parallel S) \mid R\backslash a \\
\text{(Memories)} & m & ::= \langle *, \alpha, Q \rangle \cdot m \mid \langle m, \alpha, Q \rangle \cdot m' \mid \\
 & & \langle 1 \rangle \cdot m \mid \langle 2 \rangle \cdot m \mid \langle \rangle
\end{array}$$

Interleaving semantics

$\mathcal{M}$ are the possible memories, $\hat{\mathcal{M}} = \mathcal{M} \cup \mathcal{M}^2$ and $\mathcal{P}_R$ are the RCCS processes

As usual defined by a set of rules (forward and backward)

$$(\mathcal{P}_R, \hat{\mathcal{M}} \times \mathtt{Act}_\tau, \rightarrow)$$

$$(\mathcal{P}_R, \hat{\mathcal{M}} \times \mathtt{Act}_\tau, \rightsquigarrow)$$

and

$$(\mathcal{P}_R, \hat{\mathcal{M}} \times \mathtt{Act}_\tau, \rightarrow \cup \rightsquigarrow)$$

Unravel net

The non sequential semantics is defined using unravel nets

$N = \langle S, T, F, \mathrm{m} \rangle$ causal net ($\textsc{cn}$) when

- $\forall s \in S.\ |{}^\bullet s| \leq 1$ and $|s^\bullet| \leq 1$,
- F^* is acyclic,
- $\forall s \in S\ {}^\bullet s = \emptyset \ \Rightarrow\ m(s) = 1$ and
- each transition is executable

$N = \langle S, T, F, \mathrm{m} \rangle$ unravel net (UN) when

- for each execution $X \in \mathsf{St}(N)$ the subnet $N|_X$ is a $\textsc{cn}$, and
- $\forall t, t' \in T.\ {}^\bullet t = {}^\bullet t' \ \wedge\ t^\bullet = t'^\bullet \ \Rightarrow\ t = t'$.

In UNs places may have more than one incoming arc

Keeping track of executed transitions

we have to keep track of the executed transitions, and the marking does not suffice

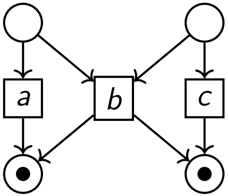
It is an UN but we do not know which transition has been executed!

complete UN $N = \langle S, T, F, \mathrm{m} \rangle$ when

- $\forall t \in T.\ \exists s_t \in S.\ {}^{\bullet}s_t = \{t\} \ \wedge\ s_t{}^{\bullet} = \emptyset$, and $|t^{\bullet}| > 1$.

Keeping track of executed transitions

we have to keep track of the executed transitions, and the marking does not suffice



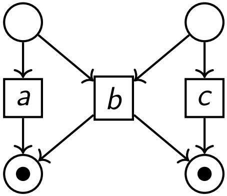
It is an UN but we do not know which transition has been executed!

complete UN $N = \langle S, T, F, \mathrm{m} \rangle$ when

- $\forall t \in T.\ \exists s_t \in S.\ {}^{\bullet}s_t = \{t\} \ \wedge \ s_t{}^{\bullet} = \emptyset$, and $|t^{\bullet}| > 1$.

Keeping track of executed transitions

we have to keep track of the executed transitions, and the marking does not suffice



It is an UN but we do not know which transition has been executed!

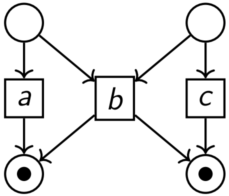
complete UN $N = \langle S, T, F, \mathsf{m} \rangle$ when

- $\forall t \in T.\ \exists s_t \in S.\ {}^\bullet s_t = \{t\} \ \wedge\ {s_t}^\bullet = \emptyset$, and $|t^\bullet| > 1$.

$\mathcal{K}_T$ are the *key*-places and $|\mathcal{K}_T| = |T|$

Keeping track of executed transitions

we have to keep track of the executed transitions, and the marking does not suffice



It is an UN but we do not know which transition has been executed!

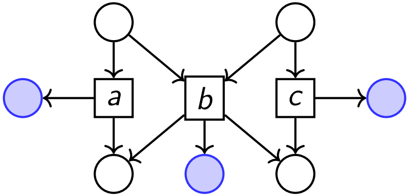
complete UN $N = \langle S, T, F, \mathrm{m} \rangle$ when

- $\forall t \in T.\ \exists s_t \in S.\ {}^\bullet s_t = \{t\} \ \wedge\ {s_t}^\bullet = \emptyset$, and $|t^\bullet| > 1$.

$\mathcal{K}_T$ are the *key*-places and $|\mathcal{K}_T| = |T|$

Keeping track of executed transitions

we have to keep track of the executed transitions, and the marking does not suffice



It is an UN but we do not know which transition has been executed!

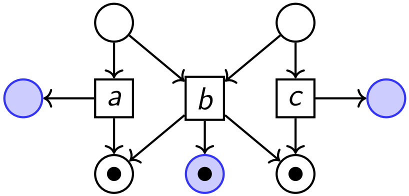
complete UN $N = \langle S, T, F, \mathrm{m} \rangle$ when

- $\forall t \in T.\ \exists s_t \in S.\ {}^\bullet s_t = \{t\} \ \wedge\ s_t{}^\bullet = \emptyset$, and $|t^\bullet| > 1$.

$\mathcal{K}_T$ are the place in blue

Keeping track of executed transitions

we have to keep track of the executed transitions, and the marking does not suffice



It is an UN but we do not know which transition has been executed!

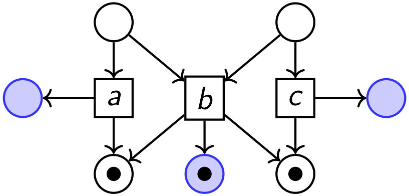
complete UN $N = \langle S, T, F, \mathrm{m} \rangle$ when

- $\forall t \in T.\ \exists s_t \in S.\ {}^\bullet s_t = \{t\} \ \wedge\ s_t{}^\bullet = \emptyset$, and $|t^\bullet| > 1$.

$\mathcal{K}_T$ are the place in blue

Keeping track of executed transitions

we have to keep track of the executed transitions, and the marking does not suffice



It is an UN but we do not know which transition has been executed!

complete UN $N = \langle S, T, F, \mathrm{m} \rangle$ when

- $\forall t \in T.\ \exists s_t \in S.\ {}^\bullet s_t = \{t\} \ \wedge\ s_t{}^\bullet = \emptyset$, and $|t^\bullet| > 1$.

$\mathcal{K}_T$ are used to reverse

Reversible Unravel net

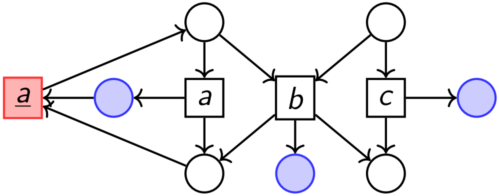
reversible UN (rCN) $N = \langle S, T, U, F, \mathsf{m} \rangle$ when

- $U \subseteq T$ and $\forall u \in U.\ \exists!\ t \in T \setminus U$ such that ${}^\bullet u = t^\bullet$ and $u^\bullet = {}^\bullet t$, and
- $N|_{T \setminus U}$ is a complete unravel net and $\langle S, T, F, \mathsf{m} \rangle$ is a safe one.

Reversible Unravel net

reversible UN (rCN) $N = \langle S, T, U, F, \mathsf{m} \rangle$ when

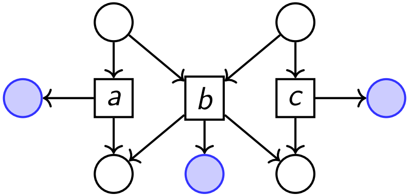
- $U \subseteq T$ and $\forall u \in U.\ \exists!\ t \in T \setminus U$ such that ${}^\bullet u = t^\bullet$ and $u^\bullet = {}^\bullet t$, and
- $N|_{T \setminus U}$ is a complete unravel net and $\langle S, T, F, \mathsf{m} \rangle$ is a safe one.



Reversible Unravel net

reversible UN (rCN) $N = \langle S, T, U, F, \mathrm{m} \rangle$ when

- $U \subseteq T$ and $\forall u \in U.\ \exists!\ t \in T \setminus U$ such that ${}^{\bullet}u = t^{\bullet}$ and $u^{\bullet} = {}^{\bullet}t$, and
- $N|_{T \setminus U}$ is a complete unravel net and $\langle S, T, F, \mathrm{m} \rangle$ is a safe one.



From UN to rcn

Just add to an UN all the reversing transitions

already done for occurrence nets (unfoldings)

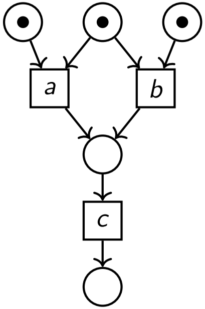
nothing really new, beside the use of key places

From UN to rcn

what we do is adding the reverse transition for each forward one:

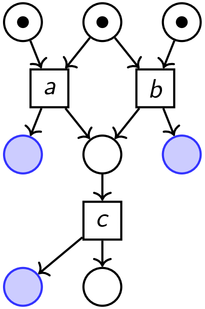
From UN to rcn

what we do is adding the reverse transition for each forward one:



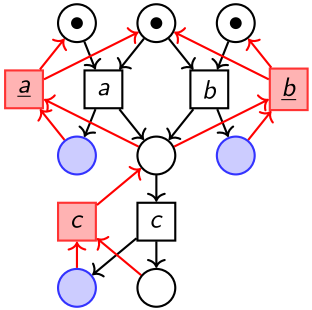
From UN to rcn

what we do is adding the reverse transition for each forward one:



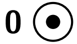
From UN to rcn

what we do is adding the reverse transition for each forward one:



The encoding of 0

The net $\mathcal{N}(\mathbf{0}) = \langle \{\mathbf{0}\}, \emptyset, \emptyset, \{\mathbf{0}\} \rangle$ is the net associated to $\mathbf{0}$ and it is called *zero*



The encoding of $\alpha.P$

Start with $\mathcal{N}(P) = \langle S_P, T_P, F_P, \mathrm{m}_P \rangle$

$\mathcal{N}(\alpha.P)$ is $\langle S_{\alpha.P}, T_{\alpha.P}, F_{\alpha.P}, \mathrm{m}_{\alpha.P} \rangle$ where

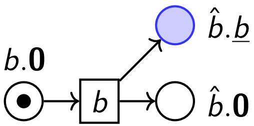
$$\begin{array}{rcl}
S_{\alpha.P} & = & \{\alpha.P, \hat{\alpha}.\underline{\alpha}\} \cup \hat{\alpha}.S_P \\
T_{\alpha.P} & = & \{\alpha\} \cup \hat{\alpha}.T_p \\
F_{\alpha.P} & = & \{(\alpha.P, \alpha), (\alpha, \hat{\alpha}.\underline{\alpha})\} \cup \{(\alpha, \hat{\alpha}.b) \mid b \in \mathrm{m}_P\} \cup \hat{\alpha}.F_P \\
\mathrm{m}_{\alpha.P} & = & \{\alpha.P\}
\end{array}$$

The encoding of $\alpha.P$

Start with $\mathcal{N}(P) = \langle S_P, T_P, F_P, \mathsf{m}_P \rangle$

$\mathcal{N}(\alpha.P)$ is $\langle S_{\alpha.P}, T_{\alpha.P}, F_{\alpha.P}, \mathsf{m}_{\alpha.P} \rangle$ where

$$\begin{array}{rcl}
S_{\alpha.P} & = & \{\alpha.P, \hat{\alpha}.\underline{\alpha}\} \cup \hat{\alpha}.S_P \\
T_{\alpha.P} & = & \{\alpha\} \ \cup \hat{\alpha}.T_p \\
F_{\alpha.P} & = & \{(\alpha.P, \alpha), (\alpha, \hat{\alpha}.\underline{\alpha})\} \cup \{(\alpha, \hat{\alpha}.b) \mid b \in \mathsf{m}_P\} \ \cup \hat{\alpha}.F_P \\
\mathsf{m}_{\alpha.P} & = & \{\alpha.P\}
\end{array}$$



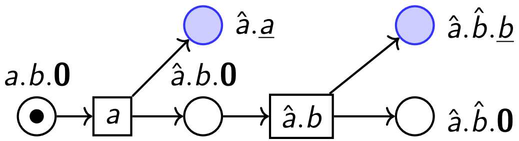
when P is $\mathbf{0}$

The encoding of $\alpha.P$

Start with $\mathcal{N}(P) = \langle S_P, T_P, F_P, \mathrm{m}_P \rangle$

$\mathcal{N}(\alpha.P)$ is $\langle S_{\alpha.P}, T_{\alpha.P}, F_{\alpha.P}, \mathrm{m}_{\alpha.P} \rangle$ where

$$\begin{array}{rcl}
S_{\alpha.P} & = & \{\alpha.P, \hat{\alpha}.\underline{\alpha}\} \cup \hat{\alpha}.S_P \\
T_{\alpha.P} & = & \{\alpha\} \ \cup \hat{\alpha}.T_p \\
F_{\alpha.P} & = & \{(\alpha.P, \alpha), (\alpha, \hat{\alpha}.\underline{\alpha})\} \cup \{(\alpha, \hat{\alpha}.b) \mid b \in \mathrm{m}_P\} \ \cup \hat{\alpha}.F_P \\
\mathrm{m}_{\alpha.P} & = & \{\alpha.P\}
\end{array}$$



when P is $b.\mathbf{0}$

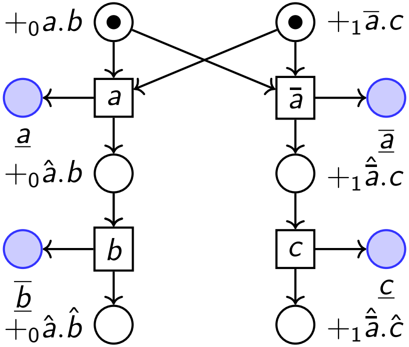
The encoding of $\sum_{i \in I} \alpha_i.P_i$

Given $\mathcal{N}(P_i)$ we get $+_{i \in I} P_i = \langle S_{+_{i \in I} P_i}, T_{+_{i \in I} P_i}, F_{+_{i \in I} P_i}, \mathsf{m}_{+_{i \in I} P_i} \rangle$

$$\begin{array}{rcl} S_{+_{i \in I} P_i} & = & \cup_{i \in I} +_{\mathsf{i}} S_{P_i} \\ T_{+_{i \in I} P_i} & = & \cup_{i \in I} +_{\mathsf{i}} T_{P_i} \\ F_{+_{i \in I} P_i} & = & \{(+_{\mathsf{i}} x, +_{\mathsf{i}} y) \mid (x, y) \in F_{P_i}\} \\ & & \cup \{(+_{\mathsf{j}} s, +_{\mathsf{i}} t) \mid s \in \mathsf{m}_{P_j} \wedge {}^{\bullet}t \in \mathsf{m}_{P_i} \wedge i \neq j\} \\ \mathsf{m}_{+_{i \in I} P_i} & = & \cup_{i \in I} +_{\mathsf{i}} \mathsf{m}_{P_i}. \end{array}$$

The encoding of $\sum_{i \in I} \alpha_i.P_i$

Given $\mathcal{N}(P_i)$ we get $+_{i\in I} P_i = \langle S_{+_{i\in I}P_i}, T_{+_{i\in I}P_i}, F_{+_{i\in I}P_i}, \mathrm{m}_{+_{i\in I}P_i} \rangle$



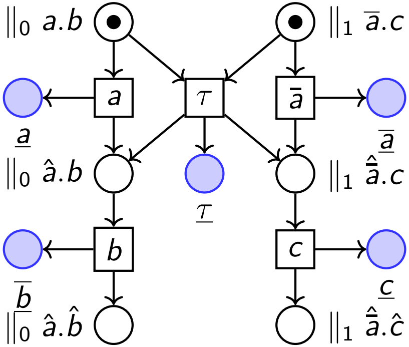
The encoding of $P \parallel Q$

Given $\mathcal{N}(P_1)$ and $\mathcal{N}(P_2)$ we get
$\mathcal{N}(P_1 \| P_2) = \langle S_{P_1\|P_2}, T_{P_1\|P_2}, F_{P_1\|P_2}, \mathsf{m}_{P_1\|P_2} \rangle$

$$\begin{array}{rcl}
S_{P_1\|P_2} & = & \|_0 S_{P_1} \cup \|_1 S_{P_2} \cup \{ s_{\{t,t'\}} \mid t \in T_{P_1} \wedge t' \in T_{P_2} \wedge \overline{\ell(t)} = \ell(t') \} \\
T_{P_1\|P_2} & = & \|_0 T_{P_1} \cup \|_1 T_{P_2} \cup \{ \{t, t'\} \mid t \in T_{P_1} \wedge t' \in T_{P_2} \wedge \overline{\ell(t)} = \ell(t') \} \\
F_{P_1\|P_2} & = & \|_0 F_{P_1} \cup \|_1 F_{P_2} \cup \{ (\{t, t'\}, s_{\{t,t'\}}) \mid t \in T_{P_1} \wedge t' \in T_{P_2} \wedge \overline{\ell(t)} = \ell(t') \} \\
& & \cup \{ (\|_{\mathsf{i}} s, \{t_1, t_2\}) \mid (s, t_i) \in F_{P_i} \} \\
& & \cup \{ (\{t_1, t_2\}, \|_{\mathsf{i}} s) \mid (t_i, s) \in F_{P_i} \;\wedge\; s \notin \mathcal{K}_{T_{P_i}} \} \\
\mathsf{m}_{P_1\|P_2} & = & \|_0 \mathsf{m}_{P_0} \cup \|_1 \mathsf{m}_{P_1}
\end{array}$$

The encoding of $P \parallel Q$

Given $\mathcal{N}(P_1)$ and $\mathcal{N}(P_2)$ we get
$\mathcal{N}(P_1 \| P_2) = \langle S_{P_1\|P_2}, T_{P_1\|P_2}, F_{P_1\|P_2}, \mathrm{m}_{P_1\|P_2} \rangle$



Next steps

Given a complete UN $N = \langle S, T, F, \mathrm{m} \rangle$ and a set of transition to be reversed $U \subseteq T$, we contruct $\overleftarrow{N} = \langle S', T', U', F', \mathrm{m}' \rangle$

- $S = S'$,
- $U' = U \times \{\mathtt{r}\}$, $T' = (T \times \{\mathtt{f}\}) \ \cup \ U'$,
- $$\begin{array}{rl} F' = & \{(s, (t, \mathtt{f})) \mid (s, t) \in F\} \ \cup \ \{((t, \mathtt{f}), s) \mid (t, s) \in F\} \ \cup \\ & \{(s, (t, \mathtt{r})) \mid (t, s) \in F\} \ \cup \ \{((t, \mathtt{r}), s) \mid (s, t) \in F\} \end{array}$$
  and
- $\mathrm{m}' = \mathrm{m}$

$\overleftarrow{N}$ is an rCN

Next steps

Start with a coherent RCCS process R, we calculate the ancestor $\rho(R)$

$$\langle \_, \alpha_z^{z}, \sum_{i \in I \setminus \{z\}} \alpha_i.P_i \rangle \cdot m \triangleright P \rightharpoondown m \triangleright \sum_{i \in I} \alpha_i.P_i \quad \text{(Act)}$$

$$\frac{R \preceq R' \qquad R' \rightharpoondown S' \qquad S' \preceq S}{R \rightharpoondown S} \text{(Pre)} \qquad \frac{R \rightharpoondown R'}{R \parallel S \rightharpoondown R' \parallel S} \text{(Par)}$$

$$\frac{R \rightharpoondown R'}{R \backslash a \rightharpoondown R' \backslash a} \text{(Res)} \qquad \frac{R \rightharpoondown^{*} \langle \rangle \triangleright P}{\rho(R) = P} \text{(Init)}$$

Next steps

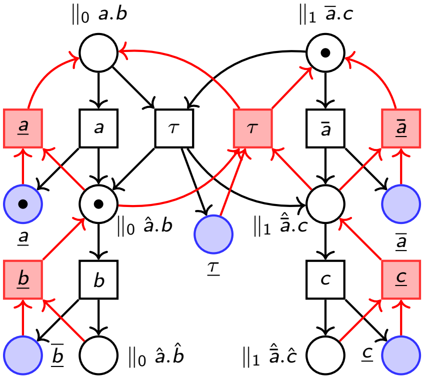
We have to calculate the marking, and we use the memories, using a mapping $\mu(R)$

$$\mu(R \parallel S) = \mu(R) \bowtie \mu(S) \qquad \mu(R\backslash a) = \backslash_{\mathsf{a}}\mu(R)$$
$$\mu(m \cdot \alpha^{i} m_1 \mathbf{0} \cdot \langle\rangle \triangleright P) = \{\alpha, m_1\} \cup \hat{\alpha}.\mu(m \cdot \langle\rangle \triangleright P)$$
$$\mu(m \cdot \alpha^{i} m_1 Q \cdot \langle\rangle \triangleright P) = \{+_{\mathsf{i}}\alpha, m_1\} \cup +_{\mathsf{i}}\hat{\alpha}.\mu(m \cdot \langle\rangle \triangleright P)$$
$$\mu(m \cdot \alpha^{i} * \mathbf{0} \cdot \langle\rangle \triangleright P) = \{\hat{\alpha}.\underline{\alpha}\} \cup \hat{\alpha}.\mu(m \cdot \langle\rangle \triangleright P)$$
$$\mu(m \cdot \alpha^{i} * Q \cdot \langle\rangle \triangleright P) = \{+_{\mathsf{i}}\hat{\alpha}.\underline{\alpha}\} \cup +_{\mathsf{i}}\hat{\alpha}.\mu(m \cdot \langle\rangle \triangleright P)$$
$$\mu(m \cdot \langle i\rangle \cdot \langle\rangle \triangleright P) = \parallel_{\mathsf{i-1}}\mu(m \cdot \langle\rangle \triangleright P)$$
$$\mu(\langle\rangle \triangleright P) = \{P\}$$

Next steps

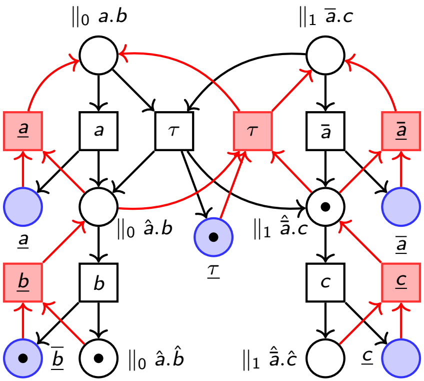
Consider R with $\rho(R) = P$. Then $\overleftarrow{\mathcal{N}(R)}$ is the net $\langle S, T, F, \mu(R)\rangle$ where $\mathcal{N}(P) = \langle S, T, F, \mathrm{m}\rangle$.

Examples



$\|_0\ a.b$
$\|_1\ \overline{a}.c$
$\underline{a}$
a
τ
τ
$\overline{a}$
$\underline{\overline{a}}$
$\underline{a}$
$\|_0\ \hat{a}.b$
$\|_1\ \hat{\overline{a}}.c$
$\underline{\overline{a}}$
$\underline{\tau}$
$\underline{b}$
b
c
$\underline{c}$
$\underline{b}$
$\|_0\ \hat{a}.\hat{b}$
$\|_1\ \hat{\overline{a}}.\hat{c}$
$\underline{c}$

Examples



$\|_0\ a.b$
$\|_1\ \overline{a}.c$
$\underline{a}$
a
τ
τ
$\overline{a}$
$\underline{\overline{a}}$
$\|_0\ \hat{a}.b$
$\|_1\ \hat{\overline{a}}.c$
$\underline{a}$
$\underline{\overline{a}}$
$\underline{\tau}$
$\underline{b}$
b
c
$\underline{c}$
$\overline{b}$
$\|_0\ \hat{a}.\hat{b}$
$\|_1\ \hat{\overline{a}}.\hat{c}$
$\underline{c}$

The main result

$$\langle\rangle \triangleright P \sim \overleftarrow{\mathcal{N}(P)}$$

Event structure and reversibility

some events $F \subseteq E$ are reversible and their removal is subject to some relations: $\prec$ (reverse causality) and $\lhd$ (prevention)

Event structure and reversibility

some events $F \subseteq E$ are reversible and their removal is subject to some relations: $\prec$ (reverse causality) and $\triangleleft$ (prevention)

sets of events ($X \subseteq E$) that can stay together represent the states of the systems (configurations)

Event structure and reversibility

some events $F \subseteq E$ are reversible and their removal is subject to some relations: $\prec$ (reverse causality) and $\triangleleft$ (prevention)

sets of events ($X \subseteq E$) that can stay together represent the states of the systems (configurations)

a natural generalization

Event structure and reversibility

some events $F \subseteq E$ are reversible and their removal is subject to some relations: $\prec$ (reverse causality) and $\lhd$ (prevention)

sets of events ($X \subseteq E$) that can stay together represent the states of the systems (configurations)

a natural generalization

$$\begin{tikzcd}
a \arrow[r, dash] \arrow[r, dashed, bend left] \arrow[d] & c \arrow[d, dash] \\
\underline{b} \arrow[ur, dashed] & b \arrow[l] \arrow[ul, dashed] \arrow[u, dashed, bend right]
\end{tikzcd}$$

Event structure and reversibility

some events $F \subseteq E$ are reversible and their removal is subject to some relations: $\prec$ (reverse causality) and $\lhd$ (prevention)

sets of events ($X \subseteq E$) that can stay together represent the states of the systems (configurations)

a natural generalization

$$\begin{tikzcd}
a \arrow[r, dash] \arrow[r, dashed, bend left] \arrow[d] & c \arrow[d, dash] \\
\underline{b} \arrow[ur, dashed] & b \arrow[l] \arrow[ul, dashed] \arrow[u, dashed, bend right]
\end{tikzcd}$$

$$\begin{tikzcd}
 & \{b\} \arrow[r] & \{a, b\} \arrow[dl] \arrow[d] \\
\emptyset \arrow[ur] \arrow[dr] & & \\
 & \{a\} & \{a, b, c\}
\end{tikzcd}$$

Summing up

the forward relations describe how events are added to a configuration (forward flow)

the backward relations describe how events are removed from a configuration (backward flow)

PN

suitable classes of Petri nets are used as a model for concurrent systems

PN

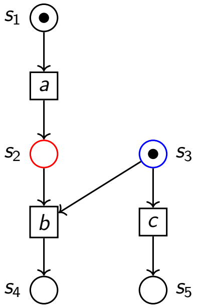
suitable classes of Petri nets are used as a model for concurrent systems

dependencies among transitions are modeled using shared places

PN

suitable classes of Petri nets are used as a model for concurrent systems

dependencies among transitions are modeled using shared places

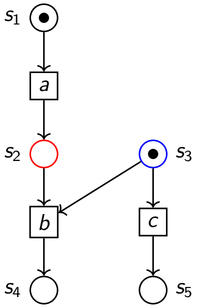


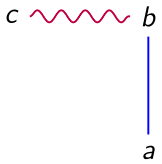
s_2 says that b depends on a and s_3 says that b and c are in conflict

PN

suitable classes of Petri nets are used as a model for concurrent systems

dependencies among transitions are modeled using shared places



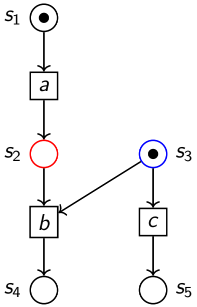


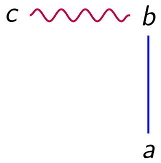
A correspondence between the net and the prime event structures is established and well know

PN

suitable classes of Petri nets are used as a model for concurrent systems

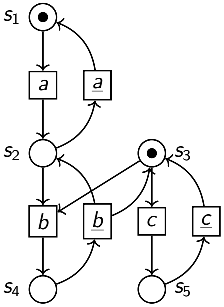
dependencies among transitions are modeled using shared places





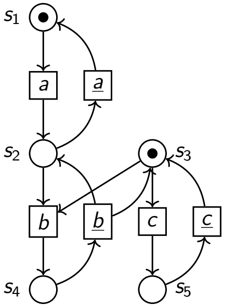
We describe perfectly the forward flow

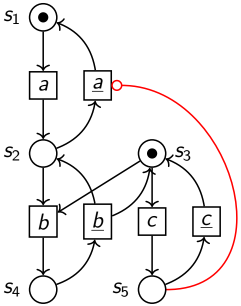
PN: the first intuition on reversing transitions

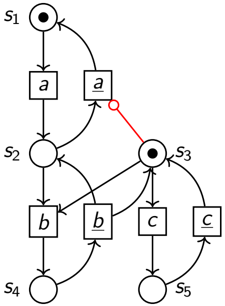


On occurrence nets add the reversing transitions

PN: modeling other backward dependencies



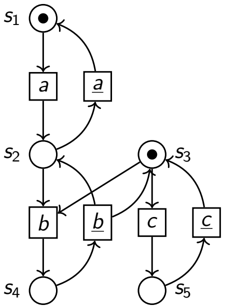


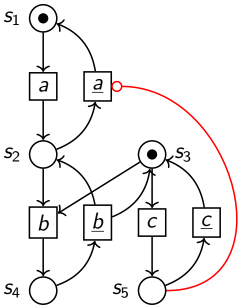


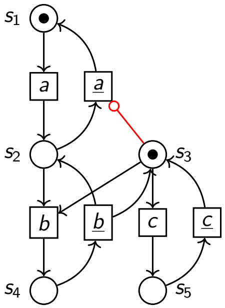
$\underline{a} \triangleleft c$

$c \prec \underline{a}$

PN: modeling other backward dependencies







$\underline{a} \lhd c$

$c \prec \underline{a}$

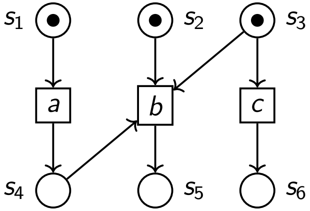
Inhibitor arcs can be used to model backward dependencies

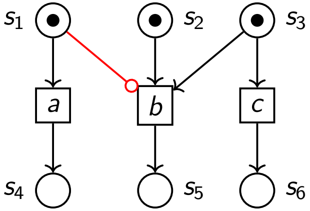
PN: pushing this idea further

Inhibitor arcs can be used to model dependencies

PN: pushing this idea further

Inhibitor arcs can be used to model dependencies





$a < b$ and $b \# c$

PN: pushing this idea further

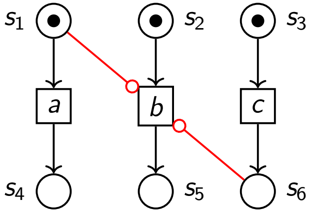
Inhibitor arcs can be used to model dependencies

Recall that conflicts can be asymmetric!

PN: pushing this idea further

Inhibitor arcs can be used to model dependencies

Recall that conflicts can be asymmetric!

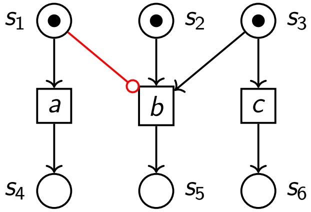


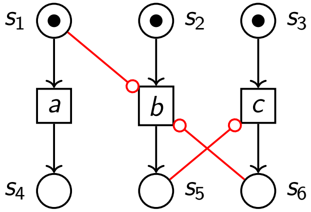
$a < b$ and $c \nearrow b$

PN: pushing this idea further

Inhibitor arcs can be used to model dependencies

Recall that conflicts can be asymmetric!





$a < b$, $b \nearrow c$ and $c \nearrow b$

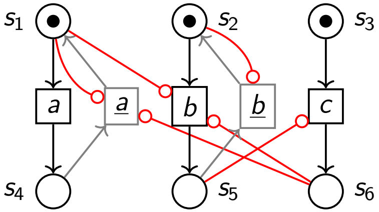
the two asymmetric conflicts $b \nearrow c$ and $c \nearrow b$ model $b \# c$

PN: pushing this idea further

Inhibitor arcs can be used to model dependencies

Recall that conflicts can be asymmetric!

We can apply the same intuition for the reversing transitions:



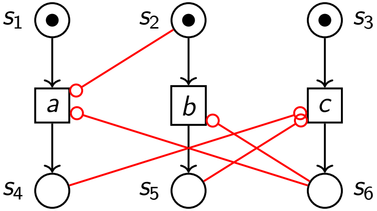
here just a and b are reversed

Causal nets

causal nets (CN) are nets where places have either just one outgoing flow arc or just one incoming flow arc

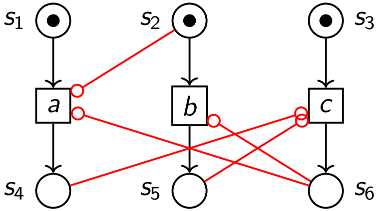
Causal nets

causal nets (CN) are nets where places have either just one outgoing flow arc or just one incoming flow arc



Causal nets

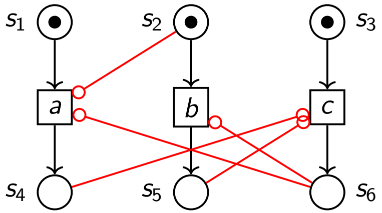
causal nets (CN) are nets where places have either just one outgoing flow arc or just one incoming flow arc



dependencies are all modeled using inhibitor arcs

Causal nets

causal nets (CN) are nets where places have either just one outgoing flow arc or just one incoming flow arc



dependencies are all modeled using inhibitor arcs

Each occurrence net can be transformed into a CN, not the vice versa

Asymmetric Event structure

asymmetric ES (AES)

Asymmetric Event structure

asymmetric ES (AES)

recall: $(E, <, \nearrow)$ where

- E is a countable set of *events*
- $< \subseteq E \times E$ is an irreflexive partial order, called *causality*, defined such that $\forall e \in E.\ \lfloor e \rfloor = \{e' \in E \mid e' \leq e\}$ is finite
- $\nearrow \subseteq E \times E$, called *weak causality*, is defined such that for all $e, e' \in E$
  - $e < e' \ \Rightarrow \ e \nearrow e'$
  - $\nearrow \cap (\lfloor e \rfloor \times \lfloor e \rfloor)$ is acyclic
  - if $e \# e'$ and $e' < e''$ then $e \# e''$.

Asymmetric Event structure

asymmetric ES (AES)

a configuration is a set X of events such that

- $\nearrow$ is well-founded on X (events are not in conflict)
- $\forall e \in X.\ \lfloor e \rfloor \subseteq X$ (it contains all the causes for each event)
- $\forall e \in X.\ \{e' \in X \mid e' \nearrow e\}$ is finite (events have finite sets of weak causes)

Asymmetric Event structure

asymmetric ES (AES)

$$\begin{tikzcd}
 & c & \\
a \arrow[ur, dash] \arrow[ur, dashed, bend left] \arrow[rr, dashed] & & b \arrow[ul, dash] \arrow[ul, dashed, bend right]
\end{tikzcd}$$

$$\begin{tikzcd}
 & \{a\} \arrow[r] & \{a, b\} \arrow[dd] \\
\emptyset \arrow[ur] \arrow[dr] & & \\
 & \{b\} & \{a, b, c\}
\end{tikzcd}$$

Asymmetric Event structure and reversible ones

asymmetric ES (AES) and reversible AES (RAES)

$(E, U, <, \nearrow, \prec, \lhd)$ where E is the set of events, $U \subseteq E$ is the set of *reversible* events, and

- $\nearrow \subseteq E \times E$, called *weak causality*;
- $\lhd \subseteq \underline{U} \times E$, called *prevention*;
- $< \subseteq E \times E$, called *causation*, is an irreflexive relation defined such that for all $e \in E$, $\lfloor e \rfloor_< = \{e' \in E \mid e' \leq e\}$ is finite and $(\nearrow \cup <)$ is acyclic on $\lfloor e \rfloor_<$;
- $\prec \subseteq E \times \underline{U}$, called *reverse causation*, is defined such that $\forall u \in U.\ u \prec \underline{u}$; and for all $u \in U$, $\lfloor u \rfloor_\prec = \{e \in E \mid e \prec \underline{u}\}$ is finite and $(\nearrow \cup <)$ is acyclic on $\lfloor e \rfloor_\prec$;
- for all $e \in E, \underline{u} \in \underline{U}.\ e \prec \underline{u} \ \Rightarrow \neg(\underline{u} \lhd e)$; and

  $(E, \lll, \nearrow)$ with $\lll = < \cap \{(e, e') \mid e \notin U \ or \ \underline{e} \lhd e'\}$ is an AES.

Asymmetric Event structure and reversible ones

asymmetric ES (AES) and reversible AES (RAES)

$A \cup \underline{B}$ is *enabled* at X if $A \cap X = \emptyset$, $B \subseteq X$, $\nearrow$ is acyclic on $A \cup X$ and

- for every $e \in A$, if $e' < e$ then $e' \in X \setminus B$ and if $e \nearrow e'$ then $e' \notin X \cup A$; and
- for every $u \in B$, if $e' \prec \underline{u}$ then $e' \in X \setminus (B \setminus \{u\})$ and if $\underline{u} \lhd e'$ then $e' \notin X \cup A$.

X is a *(reachable) configuration* if there exist a sequences of pairs of sets A_i and B_i, for $i = 1, \ldots, n$, such that

- $A_i \subseteq E$ and $B_i \subseteq U$ for all i, and
- $X_i \xrightarrow{A_i \cup \underline{B_i}} X_{i+1}$ $X_1 = \emptyset$ and $X_{n+1} = X$.

Asymmetric Event structure and reversible ones

asymmetric ES (AES) and reversible AES (RAES)

$$\begin{tikzcd}
a \arrow[r, dash] \arrow[r, dashed, bend left] \arrow[d] & c \arrow[d, dash] \\
\underline{b} \arrow[ur, dashed] & b \arrow[l] \arrow[ul, dashed] \arrow[u, dashed, bend right]
\end{tikzcd}$$

$$\begin{tikzcd}
 & \{b\} \arrow[r] & \{a, b\} \arrow[dl] \arrow[d] \\
\emptyset \arrow[ur] \arrow[r] & \{a\} & \{a, b, c\}
\end{tikzcd}$$

Why asymmetry is desirable/useful

A bit more general: we have also the symmetric version, which is a particular case, so we do not lose anything

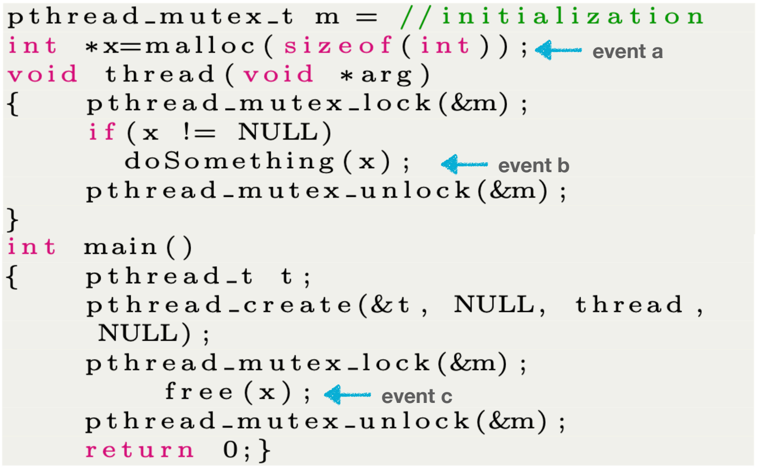
Why asymmetry is desirable/useful

A bit more general: we have also the symmetric version, which is a particular case, so we do not lose anything

The prevention relation is quite similar to an asymmetric conflict

Why asymmetry is desirable/useful

kind of concrete example



```
pthread_mutex_t m = //initialization
int *x=malloc(sizeof(int));
void thread(void *arg)
{    pthread_mutex_lock(&m);
     if(x != NULL)
        doSomething(x);
     pthread_mutex_unlock(&m);
}
int main()
{    pthread_t t;
     pthread_create(&t, NULL, thread,
      NULL);
     pthread_mutex_lock(&m);
          free(x);
     pthread_mutex_unlock(&m);
     return 0;}
```

$\underline{b} \triangleright c$ for the forward flow: c should be reversed before b is reversed

Asymmetry in CN

We consider asymmetric CN

Asymmetry in CN

$C = \langle S, T, F, I, \mathrm{m} \rangle$ is a pre asymmetric (pACN) if the following conditions hold:

- $\forall t, t' \in T.\ t^{\bullet} \cap {}^{\bullet}t' = \emptyset$;
- $\forall s \in S.\ |{}^{\bullet}s| \leq 1\ \wedge\ |s^{\bullet}| \leq 1$ and $\mathrm{m} = {}^{\bullet}T$;
- $\forall t \in T.\ |{}^{\bullet}t| = |t^{\bullet}| = 1$;
- $\lessdot^{+}$ is a partial order (induced by inhibitor arcs);
- $\forall t \in T.\ \lfloor t \rfloor_{\lessdot} = \{t' \in T \mid t' \lessdot^{*} t\}$ is finite and $(\rightsquigarrow \cup \lessdot)$ is acyclic on $\lfloor t \rfloor_{\lessdot}$ ($\rightsquigarrow$ is the weak caualty induced by the inhibitor arcs);
- for all $t, t' \in T.\ t \lessdot^{+} t'$ implies $t \lessdot t'$.

.

Asymmetry in CN

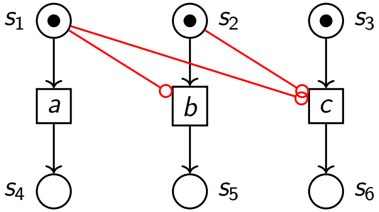
$C = \langle S, T, F, I, \mathrm{m} \rangle$ is a pre asymmetric (pACN) if the following conditions hold:

- $\forall t, t' \in T.\ t^{\bullet} \cap {}^{\bullet}t' = \emptyset$;
- $\forall s \in S.\ |{}^{\bullet}s| \leq 1\ \wedge\ |s^{\bullet}| \leq 1$ and $\mathrm{m} = {}^{\bullet}T$;
- $\forall t \in T.\ |{}^{\bullet}t| = |t^{\bullet}| = 1$;
- $\lessdot^{+}$ is a partial order (induced by inhibitor arcs);
- $\forall t \in T.\ \lfloor t \rfloor_{\lessdot} = \{t' \in T \mid t' \lessdot^{*} t\}$ is finite and $(\leadsto \cup \lessdot)$ is acyclic on $\lfloor t \rfloor_{\lessdot}$ ($\leadsto$ is the weak cauality induced by the inhibitor arcs);
- for all $t, t' \in T.\ t \lessdot^{+} t'$ implies $t \lessdot t'$.

C is an asymmetric causal net (ACN) whenever for all $t, t', t'' \in T$. $t \natural t' \wedge\ t' \lessdot t''$ imply $t \natural t''$ ($\natural$ is the symmetric conflict).

Asymmetry in CN

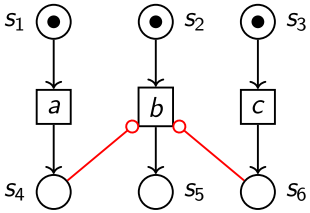
$\lessdot$ dependencies



to execute c we have to execute a and b first

Asymmetry in CN

$\rightsquigarrow$ dependencies ($t' \rightsquigarrow t$ then $t^\bullet \cap {}^\circ t' \neq \emptyset$)

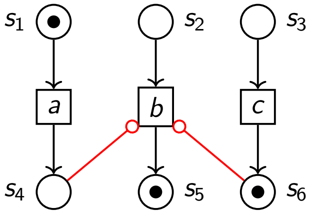


to execute b neither a nor c have been executed

the state of the system is the marking

Asymmetry in CN

$\rightsquigarrow$ dependencies ($t' \rightsquigarrow t$ then $t^\bullet \cap {}^\circ t' \neq \emptyset$)



to execute b neither a nor c have been executed

the state of the system is the marking

b and c have been executed

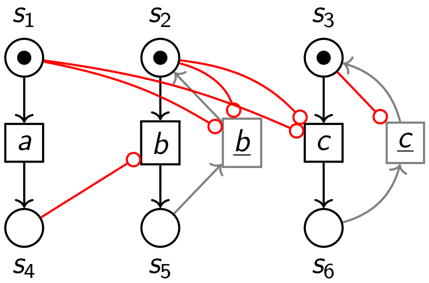
Asymmetry in CN and reversibility

$V = \langle S, T, F, I, \mathrm{m}\rangle$ is a reversible Asymmetric Causal Net (rACN) if there exists a partition $\{\overline{T}, \underline{T}\}$ of T, with $\overline{T}$ the forward transitions and $\underline{T}$ the backward ones, such that:

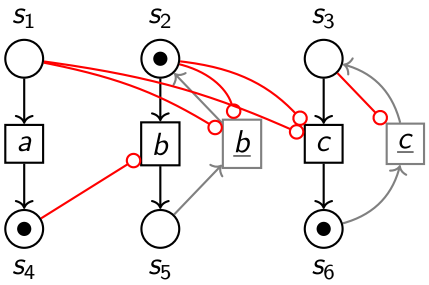
- $V_{\overline{T}} = \langle S, \overline{T}, F_{|\overline{T}\times\overline{T}}, I_{|\overline{T}\times\overline{T}}, \mathrm{m}\rangle$ is a pACN net;
- $\forall \underline{t} \in \underline{T}.\ \exists!\ t \in \overline{T}$ such that $t^\bullet = {}^\bullet\underline{t}$, ${}^\bullet t = \underline{t}^\bullet$, and ${}^\bullet t \subseteq {}^\circ\underline{t}$;
- $\forall \underline{t} \in \underline{T}.\ K_{\underline{t}} = \{t' \in \overline{T} \mid {}^\circ\underline{t} \cap {}^\bullet t' \neq \emptyset\}$ is finite and $\leadsto$ acyclic on $K_{\underline{t}}$;
- $\forall \underline{t} \in \underline{T}.\ \forall t \in \overline{T}$ if ${}^\bullet t \cap {}^\circ\underline{t} \neq \emptyset$ then $t^\bullet \cap {}^\circ\underline{t} = \emptyset$;
- $\forall t, t', t'' \in \overline{T}.\ t \natural t' \ \wedge\ t' \lll t'' \ \Rightarrow\ t \natural t''$ with $\lll$ being the transitive closure of $\lessdot \cap \{(t, t') \mid \underline{t} \notin \underline{T}\ \mathit{or}\ {}^\circ\underline{t} \cap t'^\bullet \neq \emptyset\}$.

code the relations of reverse causality and prevention with inhibitor arcs

Asymmetry in CN and reversibility



Asymmetry in CN and reversibility



the markings determine the state of the system, and it is reached executing first b, then a, then c and reversing b

Results: from rACN to rAES

$V^{\underline{T}} = \langle S, T, F, I, \mathrm{m} \rangle$ be an rACN.

$\mathcal{E}_r(V^{\underline{T}})$ is the tuple $(\overline{T}, U, \lessdot, \lll \cup \rightsquigarrow, \prec, \lhd)$ where
$U = \{t \in \overline{T} \mid \underline{t} \in \underline{T}\}$ are the reversible events and $\lessdot$, $\rightsquigarrow$, $\prec$, $\lhd$ and $\lll$ are the causation, weak causality, reverse causality, prevention and sustained causation induced by F and I

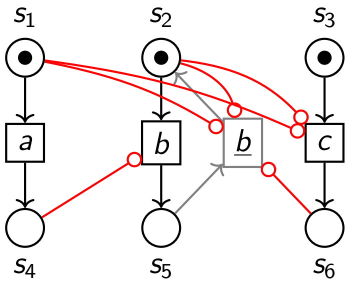
Results: from rACN to rAES

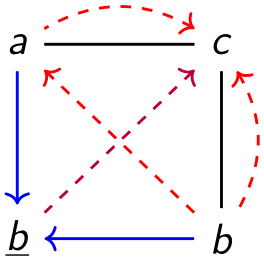
$V^{\underline{T}} = \langle S, T, F, I, \mathrm{m} \rangle$ be an rACN.

$\mathcal{E}_r(V^{\underline{T}})$ is the tuple $(\overline{T}, U, \lessdot, \lll \cup \leadsto, \prec, \lhd)$ where $U = \{t \in \overline{T} \mid \underline{t} \in \underline{T}\}$ are the reversible events and $\lessdot$, $\leadsto$, $\prec$, $\lhd$ and $\lll$ are the causation, weak causality, reverse causality, prevention and sustained causation induced by F and I

$\mathcal{E}_r(V)$ is a reversible asymmetric event structure

Results: from rACN to rAES





Results: from rACN to rAES

If m $\setminus {}^\bullet X \cup X^\bullet$ is a reachable marking of an rACN V then X is also a configuration of $\mathcal{E}_r(V)$

Results: from rAES to rACN

$\mathsf{H} = (E, U, <, \nearrow, \prec, \lhd)$ reversible asymmetric event structure

Results: from rAES to rACN

$\mathsf{H} = (E, U, <, \nearrow, \prec, \lhd)$ reversible asymmetric event structure

$\mathcal{N}_r(\mathsf{H})$ is the net $\langle S, E \cup \{(\underline{u}, \mathtt{r}) \mid u \in U\}, F, I, \mathrm{m}\rangle$ where

$$S = \{(A, B) \mid A, B \subseteq E \ \wedge\ |A| \leq |B| \leq 1 \ \wedge\ A \subseteq B\}$$

$$\begin{aligned} F = {} & \{((\emptyset, \{e\}), e) \mid e \in E\} \cup \{(e, (\{e\}, \{e\})) \mid e \in E\} \cup \\ & \{((\{u\}, \{u\}), (\underline{u}, \mathtt{r})) \mid u \in U\} \cup \{((\underline{u}, \mathtt{r}), (\emptyset, \{u\})) \mid u \in U\} \end{aligned}$$

$$\begin{aligned} I = {} & \{((\emptyset, \{e'\}), e) \mid e' < e\} \cup \{((\{e'\}, \{e'\}), e) \mid e \nearrow e' \wedge \neg(e' < e)\} \\ & \cup \{((\emptyset, \{e\}), (u, \mathtt{r})) \mid e \prec \underline{u}\} \cup \{((\{e\}, \{e\}), (u, \mathtt{r})) \mid \underline{u} \lhd e\} \end{aligned}$$

$$\mathrm{m} = \{(\emptyset, A) \mid (\emptyset, A) \in S\}$$

Results: from rAES to rACN

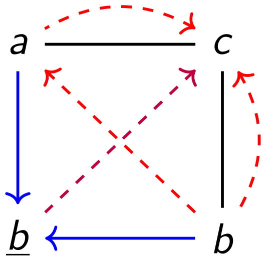
$\mathsf{H} = (E, U, <, \nearrow, \prec, \lhd)$ reversible asymmetric event structure

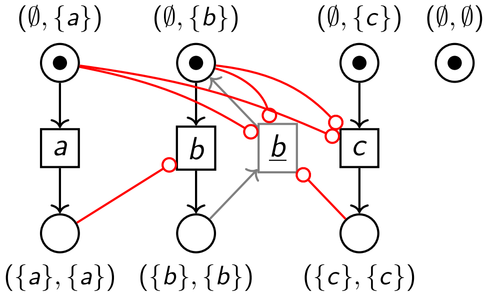
$\mathcal{N}_r(\mathsf{H})$ is the net $\langle S, E \cup \{(\underline{u}, \mathtt{r}) \mid u \in U\}, F, I, \mathrm{m}\rangle$ where

$$
\begin{aligned}
S &= \{(A, B) \mid A, B \subseteq E \ \wedge \ |A| \leq |B| \leq 1 \ \wedge \ A \subseteq B\} \\
F &= \{((\emptyset, \{e\}), e) \mid e \in E\} \cup \{(e, (\{e\}, \{e\})) \mid e \in E\} \cup \\
&\qquad \{((\{u\}, \{u\}), (\underline{u}, \mathtt{r})) \mid u \in U\} \cup \{((\underline{u}, \mathtt{r}), (\emptyset, \{u\})) \mid u \in U\} \\
I &= \ \{((\emptyset, \{e'\}), e) \mid e' < e\} \cup \{((\{e'\}, \{e'\}), e) \mid e \nearrow e' \wedge \neg(e' < e)\} \\
&\qquad \cup \{((\emptyset, \{e\}), (u, \mathtt{r})) \mid e \prec \underline{u}\} \cup \{((\{e\}, \{e\}), (u, \mathtt{r})) \mid \underline{u} \lhd e\} \\
\mathrm{m} &= \{(\emptyset, A) \mid (\emptyset, A) \in S\}
\end{aligned}
$$

$\mathcal{N}_r(\mathsf{H})$ is an reversible causal net

Results: from rAES to rACN





Results: from rAES to rACN

If X is a configuration of an rAES H then $\mathsf{m} \setminus {}^\bullet X \cup X^\bullet$ is a reachable marking of $\mathcal{N}_r(\mathsf{H})$