Generalising Mazurkiewicz Traces

Maciej Koutny
Newcastle University, UK

with

Ryszard Janicki (McMaster University, Canada)
Jetty Kleijn (Leiden University, The Netherlands)
Łukasz Mikulski (Nicolaus Copernicus University, Poland)

IFIP WG 2.2 meeting, Munich, September 2014

Outline

- Mazurkiewicz traces
- Traces consisting of step sequences
- Order structures for generalised traces
- Axiomatisation
- Saturation and closure
- Consistency results

Mazurkiewicz traces

- Basic ingredients

  concurrency alphabet (Σ, ind)

  alphabet of actions Σ *finite*

  independence relation on actions ind *irreflexive* *symmetric*

  equations $ab = ba$ *if* $(a,b) \in ind$

- Equivalent sequences of actions

  $wabu \approx wbau$ if $ab = ba$

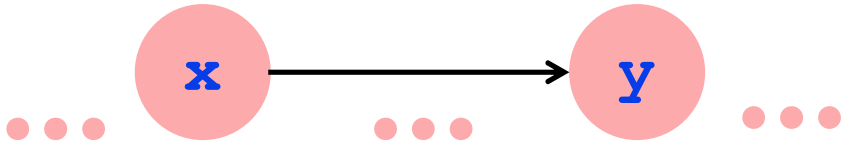
  $x \equiv y$ if $x \approx z \approx \ldots \approx v \approx y$

- (Mazurkiewicz) trace

  equivalence class $[x]$ of $\equiv$

Traces and orders

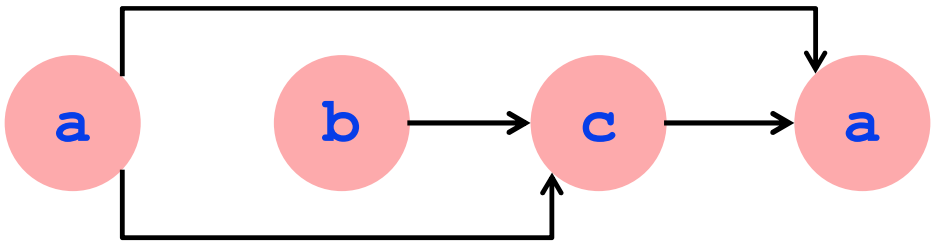
- Traces can be represented by (causal) partial orders
- Extracting dependencies from a sequence `...x...y...`



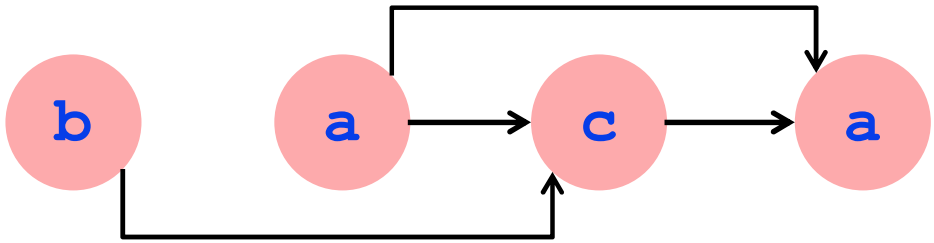
$(x,y) \notin ind$

- Example acyclic dependence graphs

$(a,b) \in ind$ $\quad [abca] = \{abca, baca\}$



the same



Observations

- Acyclic relations are fundamental as they can represent:
  execution orders *as well as* causal orders

- Execution orders are saturated acyclic relations:
  all execution orderings are determined

- Causal orders are saturation closed:
  all direct and indirect dependencies are present
  nothing can be added without excluding some executions

- Execution orders = total partial orders

- Causal orders = partial orders

Facts

Any acyclic relation can be linearised

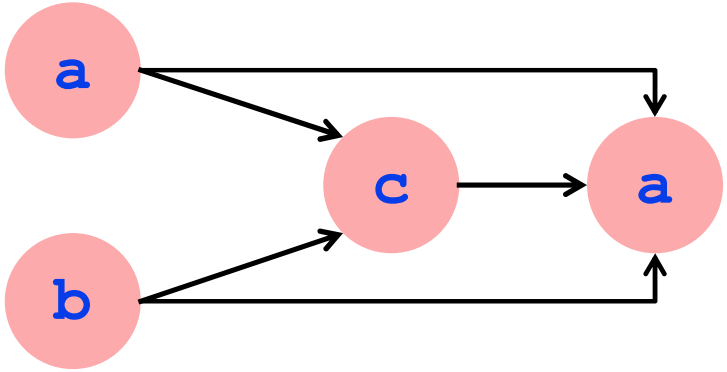
Any acyclic relation can be extended to a unique partial order which has the same total order extensions, through transitive closure

Partial orders are exactly those acyclic relations which can be derived from their total order extensions, through intersection

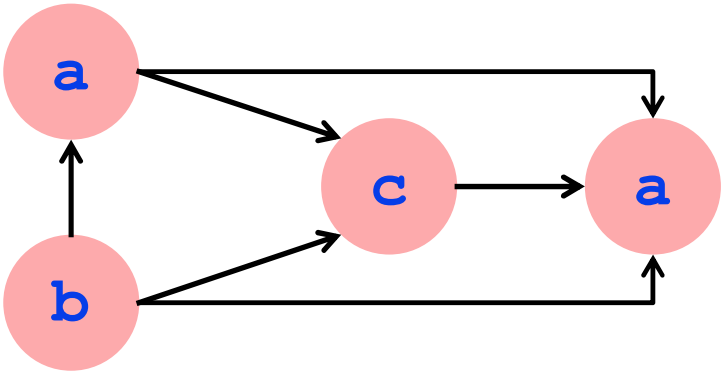
[Szpilrajn 1935]

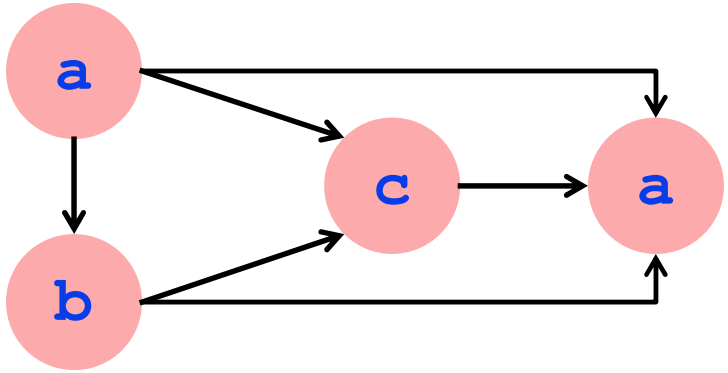
Traces and orders

- Causal partial order is obtained through transitive closure

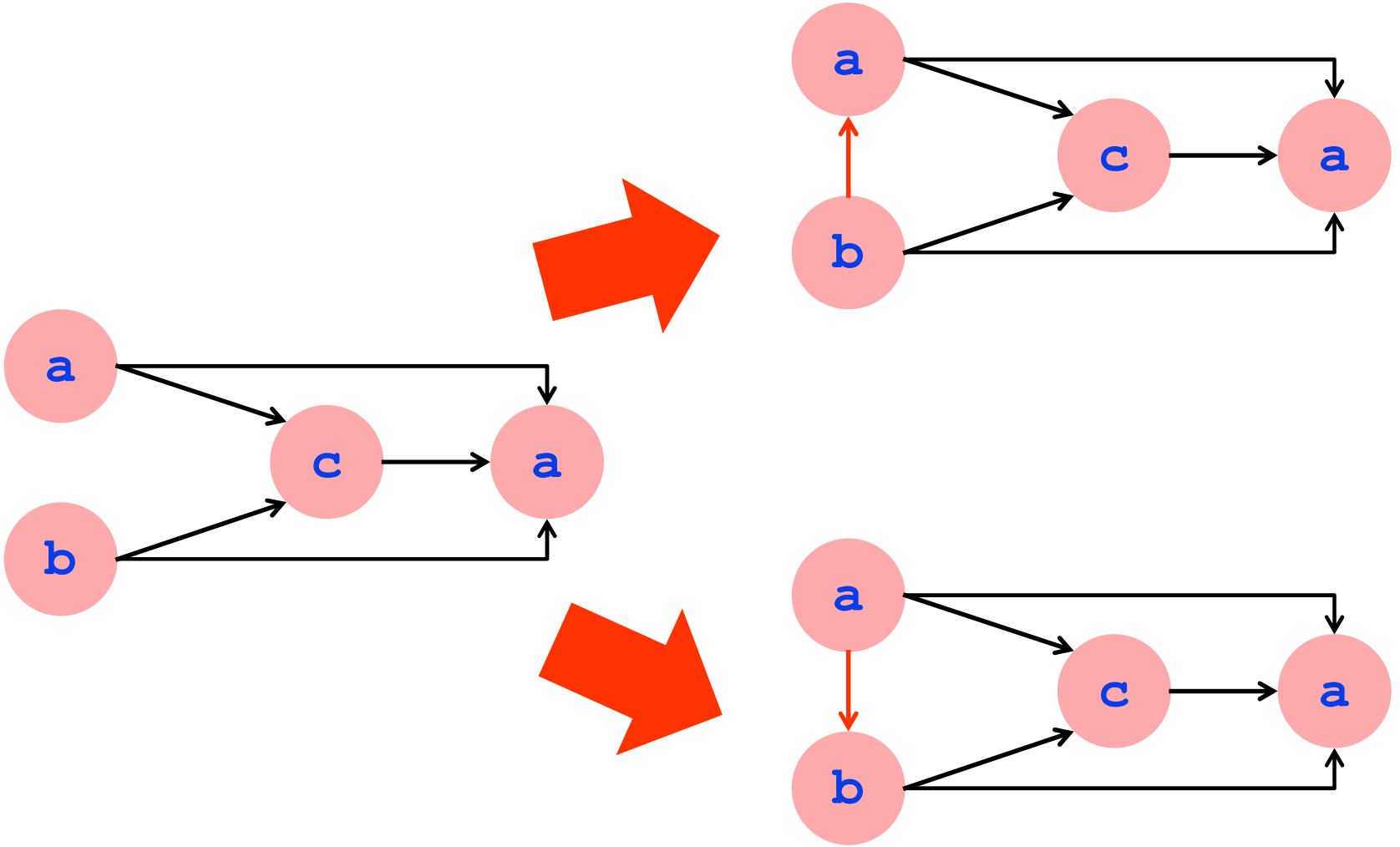


- Sequences of actions correspond to total orders
- Total orders = *saturated* partial orders

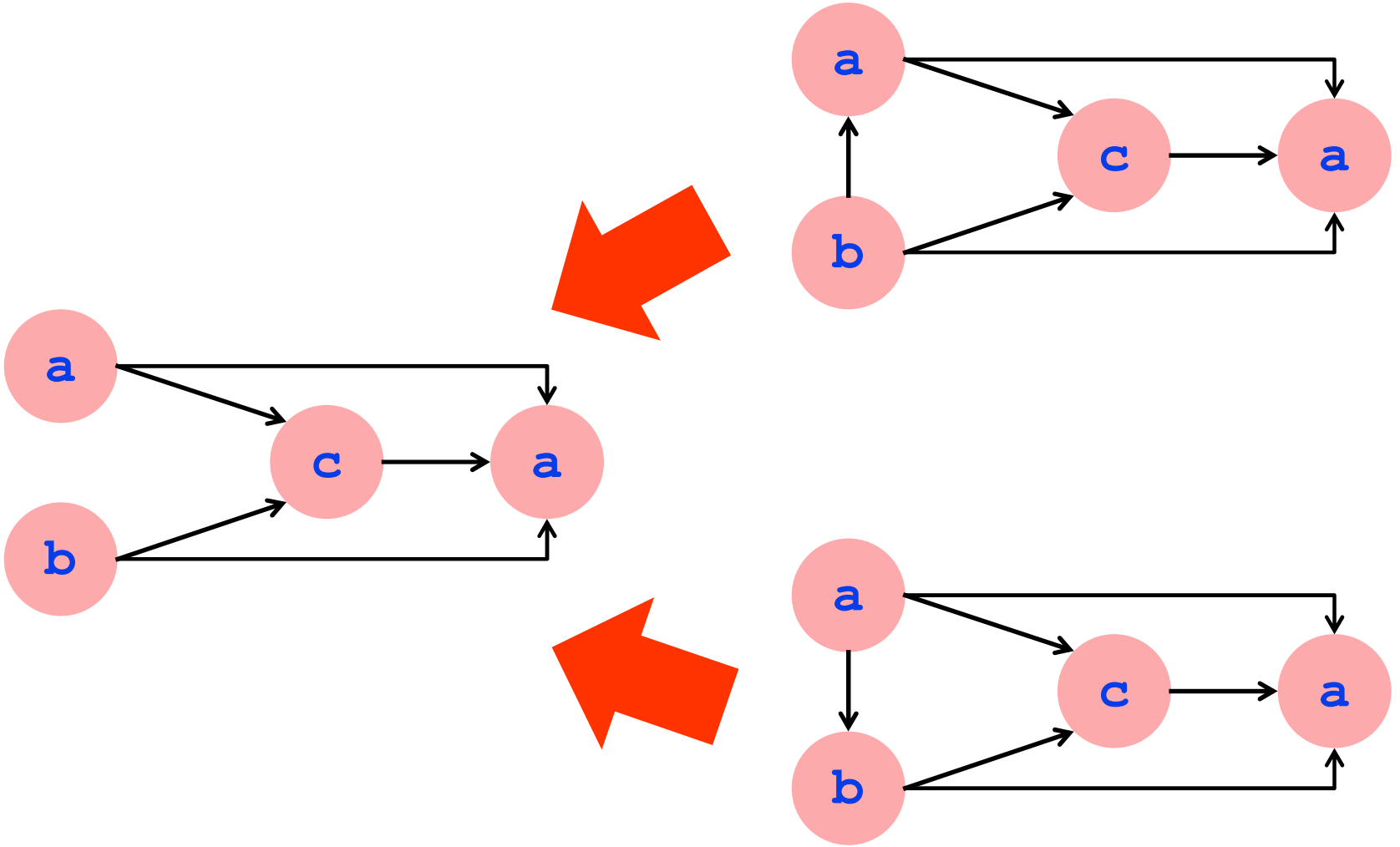




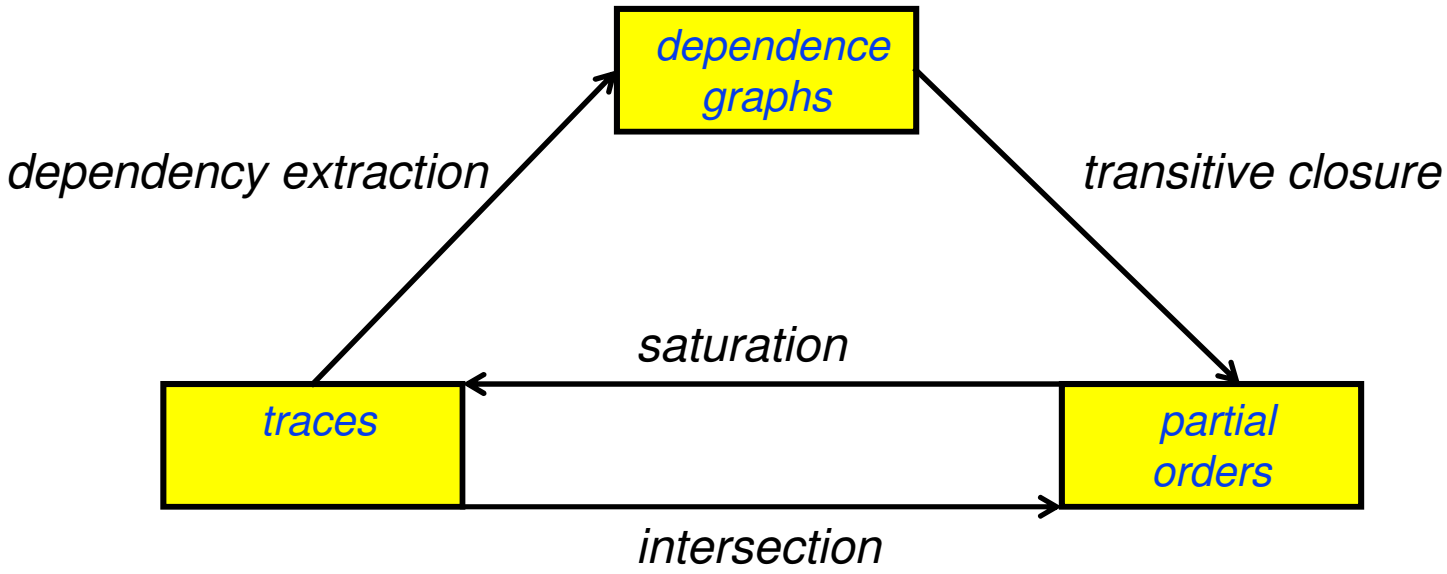
Saturation



Intersection



Consistency



Extending traces

- (sequences of) actions → (sequences of) sets of actions
- Steps can be swapped → **AB = BA**
- Not interesting enough (essentially no change)!
- Introduce serialisation equations
- Steps can be split → **C = DE**

Extending traces

- Three irreflexive relations on actions

  sim inl ser

- simultaneity (symmetric) **sim** defines legal steps

  A $\in \mathcal{S}$ if $\mathbf{A \times A - id_\Sigma \subseteq sim}$

- interleaving (symmetric) **inl** defines equations

  AB = BA if $\mathbf{A \times B \subseteq inl}$

- serialisability **ser** defines equations

  C = DE if $\mathbf{D \times E \subseteq ser}$ $\mathbf{C = D \cup E}$

Assumed: $\mathbf{inl \cap sim = \varnothing}$ $\mathbf{ser \subseteq sim}$

Extending traces

- Take

  $A \times B \subseteq ser$ and $B \times A \subseteq ser$

- Then

  $A \cup B = AB$ and $B \cup A = BA$

- Leading to interleaving equation

  $AB = BA$

- Hence, more generally (motivation: Petri nets)

  $AB = BA$

  whenever for all $a \in A$ and $b \in B$

  $(a,b) \in inl$ or

  $(a,b) \in ser$ and $(b,a) \in ser$

Concurrency alphabets

- Fundamental concurrency alphabet:

  $(\Sigma, \mathtt{sim}, \mathtt{inl}, \mathtt{ser})$

- Two relations suffice!

- Generalised concurrency alphabet:

  $(\Sigma, \mathtt{sim}, \mathtt{seq})$

- Sequentialisability $\mathtt{seq}$ defines equations over S

  $\mathbf{AB} = \mathbf{BA}$ if $\mathbf{A} \times \mathbf{B} \subseteq \mathtt{seq} \cap \mathtt{seq}^{-1}$

  $\mathbf{A} \cup \mathbf{B} = \mathbf{AB}$ if $\mathbf{A} \times \mathbf{B} \subseteq \mathtt{seq} \cap \mathtt{sim}$

System derived relationships

- *How to define fundamental concurrency alphabet ?*

  (Σ, sim, inl, ser)

- For transitions in elementary net systems (*a priori* semantics):

  sim disjointness of input/output places
  ser read arcs allowing concurrent access
  inl mutex arcs disallowing concurrent access

Mazurkiewicz alphabets

How to model (Σ, ind) *?*

- Fundamental concurrency alphabet:
  $(\Sigma, \varnothing, ind, \varnothing)$

- Generalised concurrency alphabet:
  $(\Sigma, \varnothing, ind)$

Concurrency alphabets

Going between two types of extension:

$$\mathbf{seq} = \mathbf{inl} \cup \mathbf{ser}$$

$$\mathbf{inl} = (\mathbf{seq} \cap \mathbf{seq}^{-1}) - \mathbf{sim}$$

$$\mathbf{ser} = \mathbf{seq} \cap \mathbf{sim}$$

leads to the same equations and traces

Derived relationships

- strong simultaneity

  $\mathtt{ssi} = \mathtt{sim} - (\mathtt{seq} \cup \mathtt{seq}^{-1}) = \mathtt{sim} - (\mathtt{ser} \cup \mathtt{ser}^{-1})$

- concurrency

  $\mathtt{con} = \mathtt{seq} \cap \mathtt{seq}^{-1} \cap \mathtt{sim} = \mathtt{ser} \cap \mathtt{ser}^{-1}$

- semi-serialisability

  $\mathtt{sse} = (\mathtt{seq} - \mathtt{seq}^{-1}) \cap \mathtt{sim} = \mathtt{ser} - \mathtt{ser}^{-1}$

- weak dependence

  $\mathtt{wdp} = (\mathtt{seq}^{-1} - \mathtt{seq}) \cap \mathtt{sim} = \mathtt{ser}^{-1} - \mathtt{ser}$

- rigid ordering

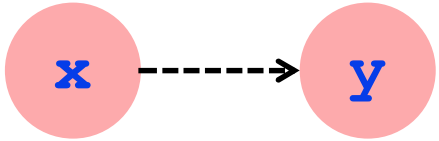
  $\mathtt{rig} = \Sigma\times\Sigma - ((\mathtt{seq} \cap \mathtt{seq}^{-1}) \cup \mathtt{sim})$

  $= \Sigma\times\Sigma - (\mathtt{sim} \cup \mathtt{inl})$

Together with $\mathtt{inl}$ ***they partition*** $\Sigma\times\Sigma$

Orderings for generalised traces

- *What relationships fit generalised traces ?*
- Two relations:

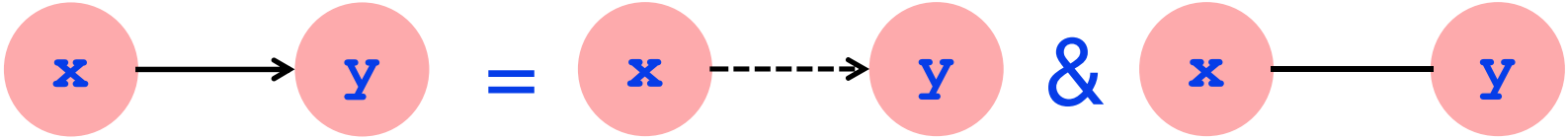
weak causality 

x before or together with **y** (not after)

mutex 

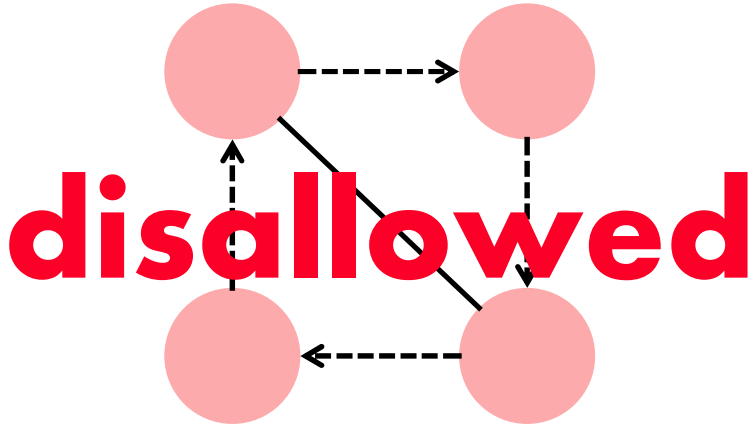
x before or after **y** (not together)

- Causality / execution precedence (derived):

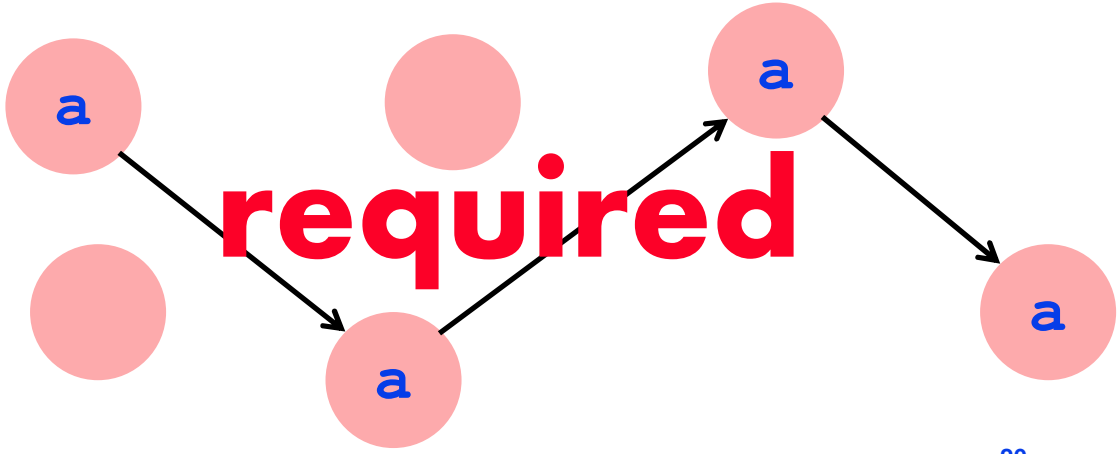


Order structures (“acyclic relations”)

Separability: weak causality cycles and mutex do not mix

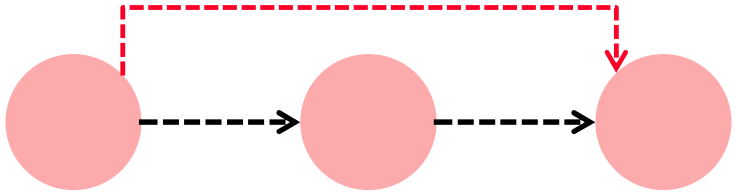


Label linearity: similarly labelled events are totally ordered

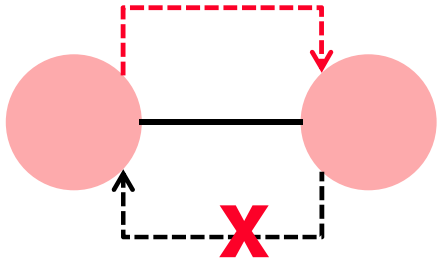


Saturated order structures

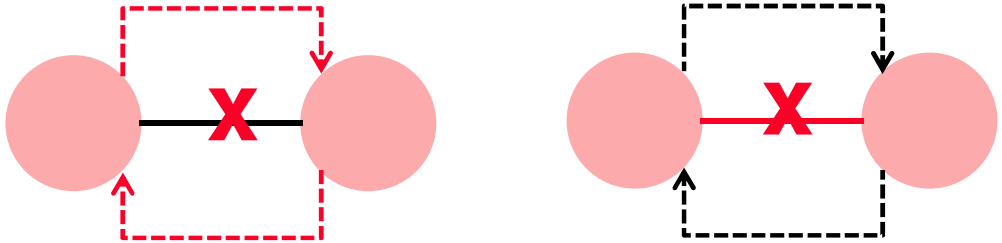
L1



L2



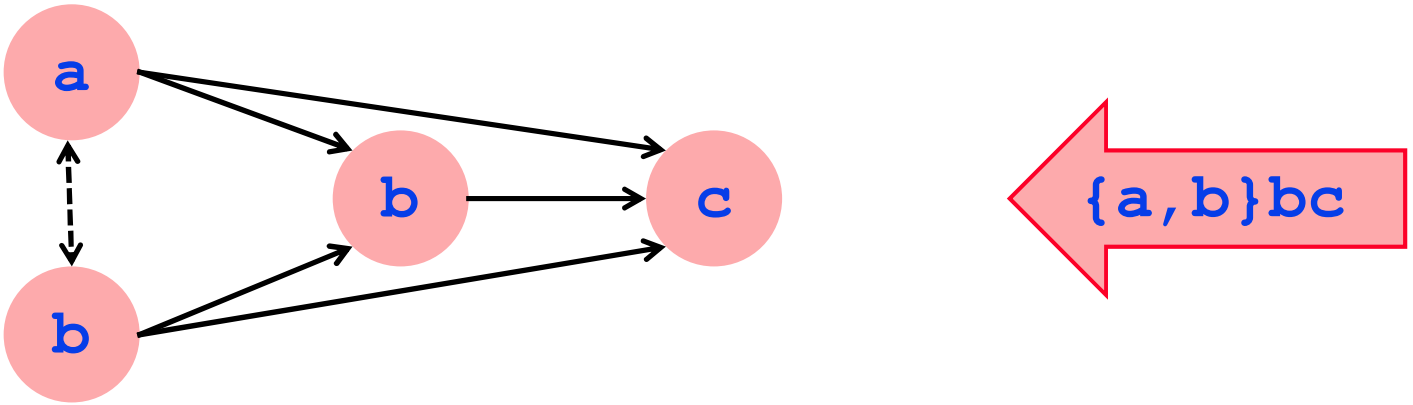
L3



L4

Saturated order structures

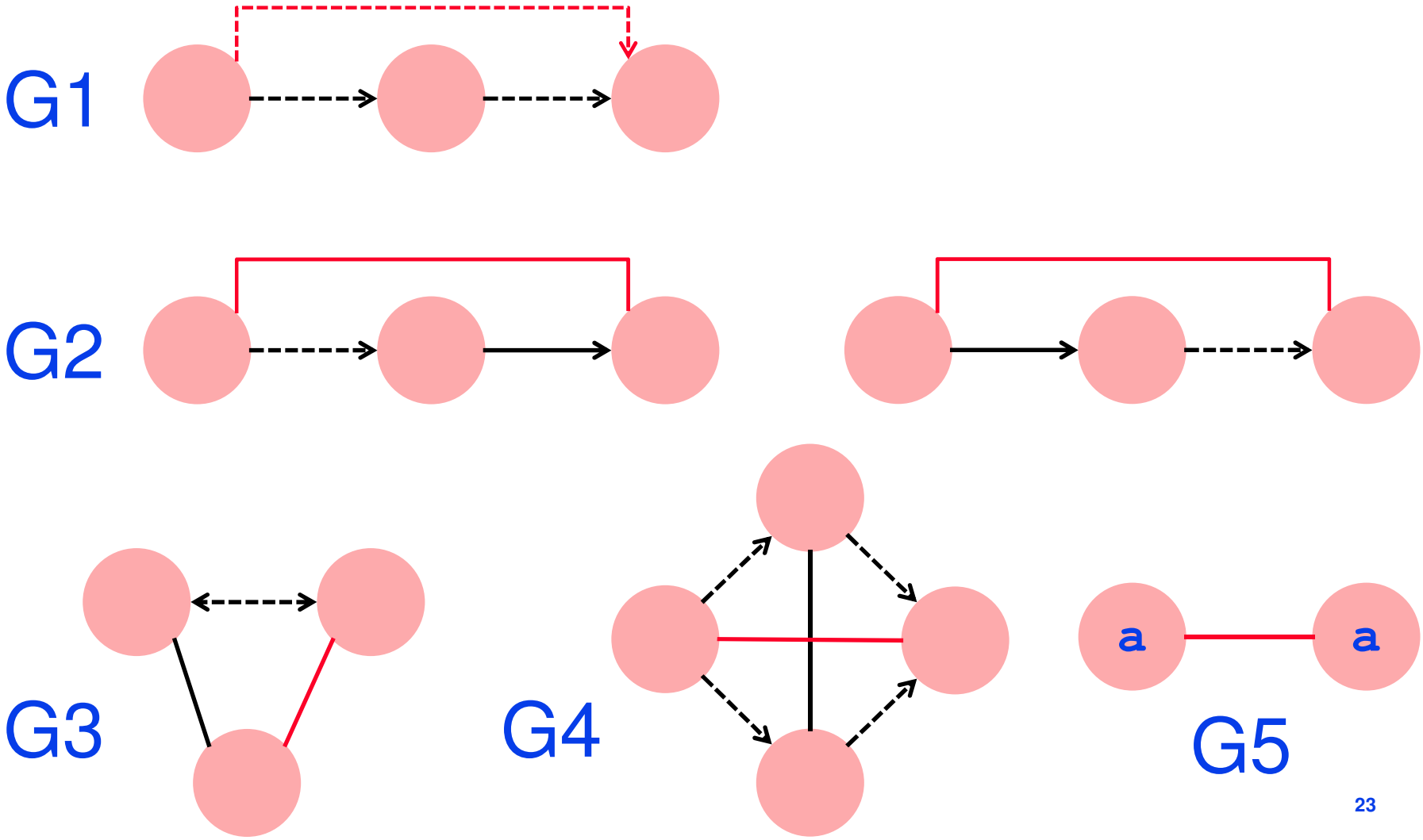
Saturated order structures consist of simultaneus *layers*



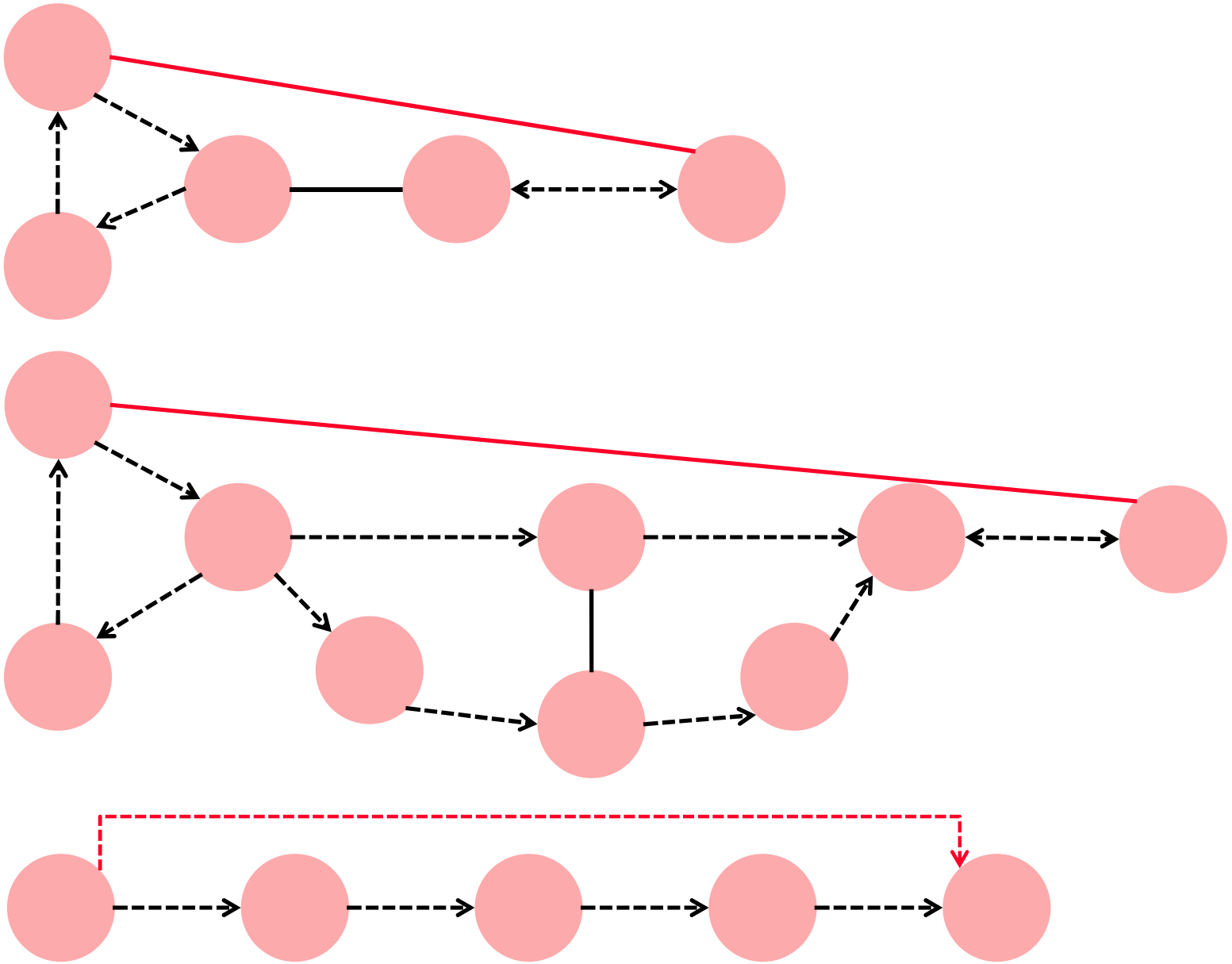
and are in one-to-one correspondence with step sequences

Invariant order structures

Both relations are irreflexive, and mutex is also symmetric



Invariant closure



Facts

Any acyclic relation can be linearised

Any acyclic relation can be extended to a unique partial order which has the same total order extensions, through transitive closure

Partial orders are exactly those acyclic relations which can be derived from their total order extensions, through intersection

[Szpilrajn 1935]

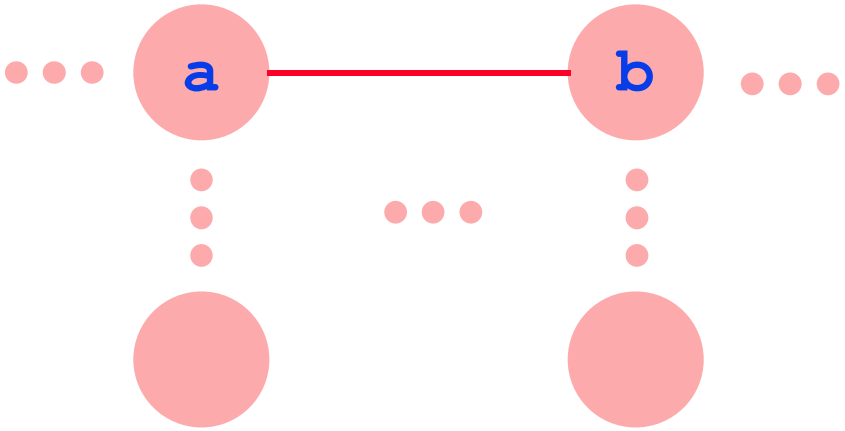
Facts

Any order structure can be saturated

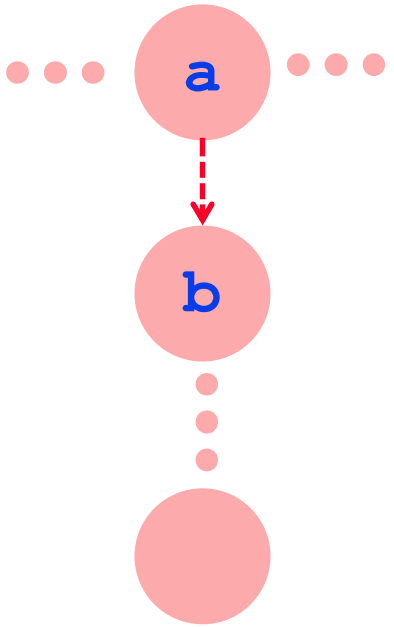
Any order structure can be extended to a unique invariant order structure which has the same saturations, through invariant closure

Invariant order structures are exactly those order structures which can be derived from their saturations, through intersection

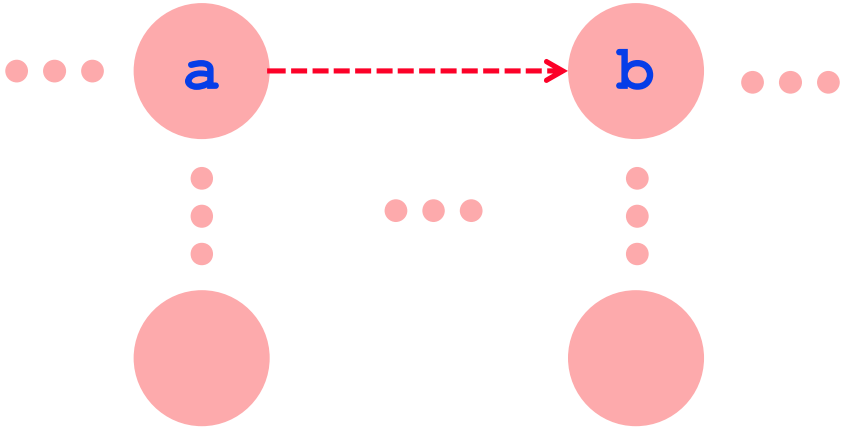
Dependencies in step sequences



$(a,b) \notin sim \cap seq$

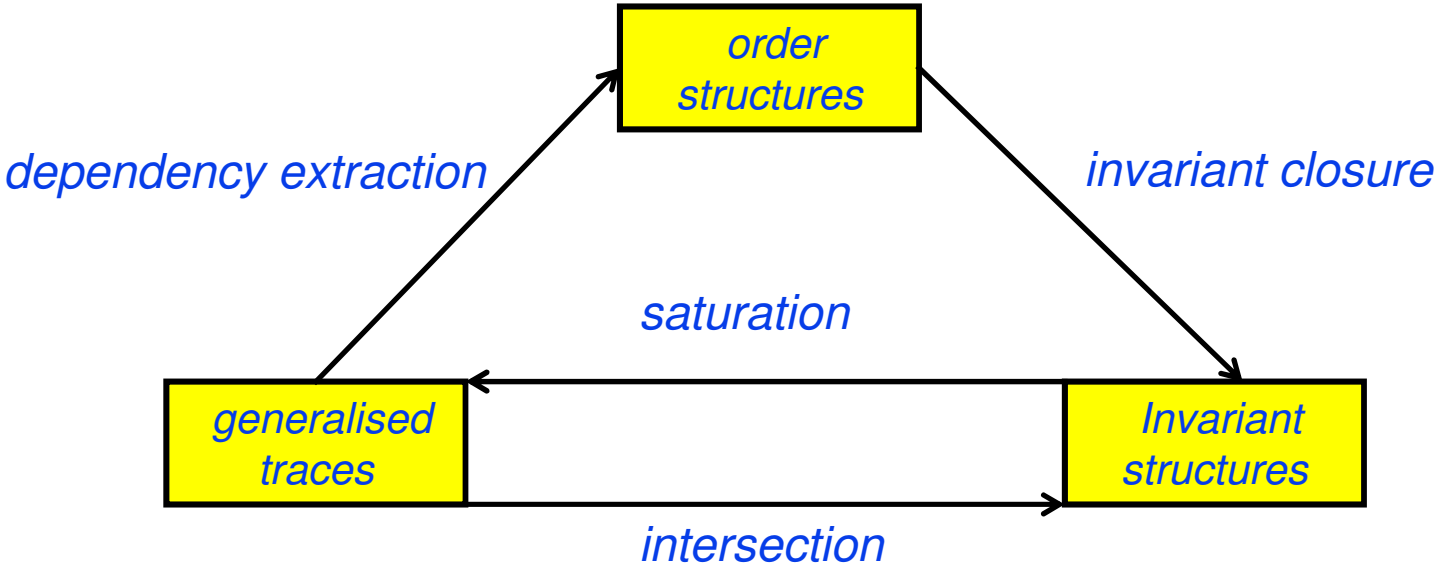


$(b,a) \in sim\text{-}seq$



$(a,b) \notin seq \cap seq^{-1}$

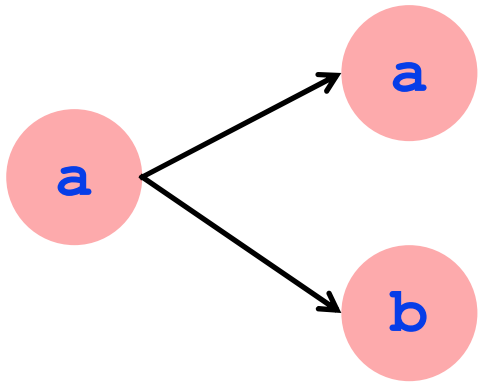
Consistency



Do we need order structures?

YES
Every order structure with injective labelling is generated by a step sequence over some generalised concurrency alphabet

BUT
Not every order structure can be generated by a step sequence over some generalised concurrency alphabet



Conclusion

- We obtained the following:

sequences	step sequences
acyclic relations	order structures
total orders	saturated order structures
partial orders	invariant order structures
transitive closure	invariant closure

- Open problem: strengthen label linearity to filter out order structures which are not generated by generalised traces

Thank You!