Simons Institute for the Theory of Computing
Reunion Workshop for the Program on "Logical Structures in Computation"
Monday - Thursday December 11 - 14, 2017

Concurrency and Probability:

Removing Confusion, Compositionally

Ugo Montanari
Dipartimento di Informatica, University of Pisa

Joint work with Roberto Bruni and Hernán Melgratti

Information Society Technologies

Roadmap

- Concurrency: a useful abstraction level
- Equivalent computations may have different decision points and different probabilities
- Petri occurrence nets with confusion
- Our result: compiling a net with confusion into one without confusion
- Additional causal links for transmitting negative conditions
- The resulting net is a net with persistence for handling OR causality
- Conclusion and future work

Roadmap

- Concurrency: a useful abstraction level
- Equivalent computations may have different decision points and different probabilities
- Petri occurrence nets with confusion
- Our result: compiling a net with confusion into one without confusion
- Additional causal links for transmitting negative conditions
- The resulting net is a net with persistence for handling OR causality
- Conclusion and future work

Concurrency Theory, I

- a useful widespread abstraction
  - for the design and use of a variety of systems
- concurrent computations
  - equivalence classes of execution sequences
  - pairs of concurrent events can be executed in any order
- sequences in the same class are indistinguishable
  - for the current purpose of interest
- behavior independent on
  - time
  - speed of processors
- causal dependencies between events
- nondeterminism via mutual exclusion of events

Roadmap

- Concurrency: a useful abstraction level
- Equivalent computations may have different decision points and different probabilities
- Petri occurrence nets with confusion
- Our result: compiling a net with confusion into one without confusion
- Additional causal links for transmitting negative conditions
- The resulting net is a net with persistence for handling OR causality
- Conclusion and future work

Concurrency Theory, II

- inadequate when modeling explicit choice points
  - equivalent sequences behave very differently
  - alternatives can be created/deleted by concurrent events
  - => the confusion problem
- hard when combined with probabilities
  - nondeterminism vs. probability/stochastic distributions
  - exponential distributions for process races
  - nondeterminism for distributed decisionsß
  - schedulers for optimal control
- time can hardly be ruled out

concurrency is too coarse an abstraction?

Petri nets as a touchstone

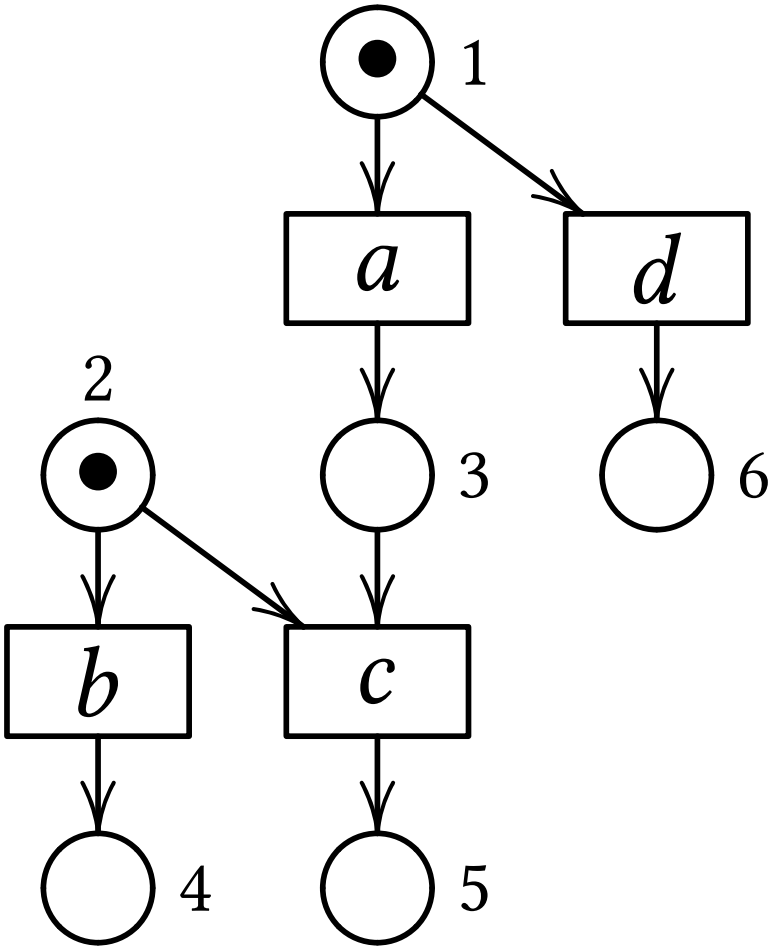
Roadmap

- Concurrency: a useful abstraction level
- Equivalent computations may have different decision points and different probabilities
- Petri occurrence nets with confusion
- Our result: compiling a net with confusion into one without confusion
- Additional causal links for transmitting negative conditions
- The resulting net is a net with persistence for handling OR causality
- Conclusion and future work

Confusion

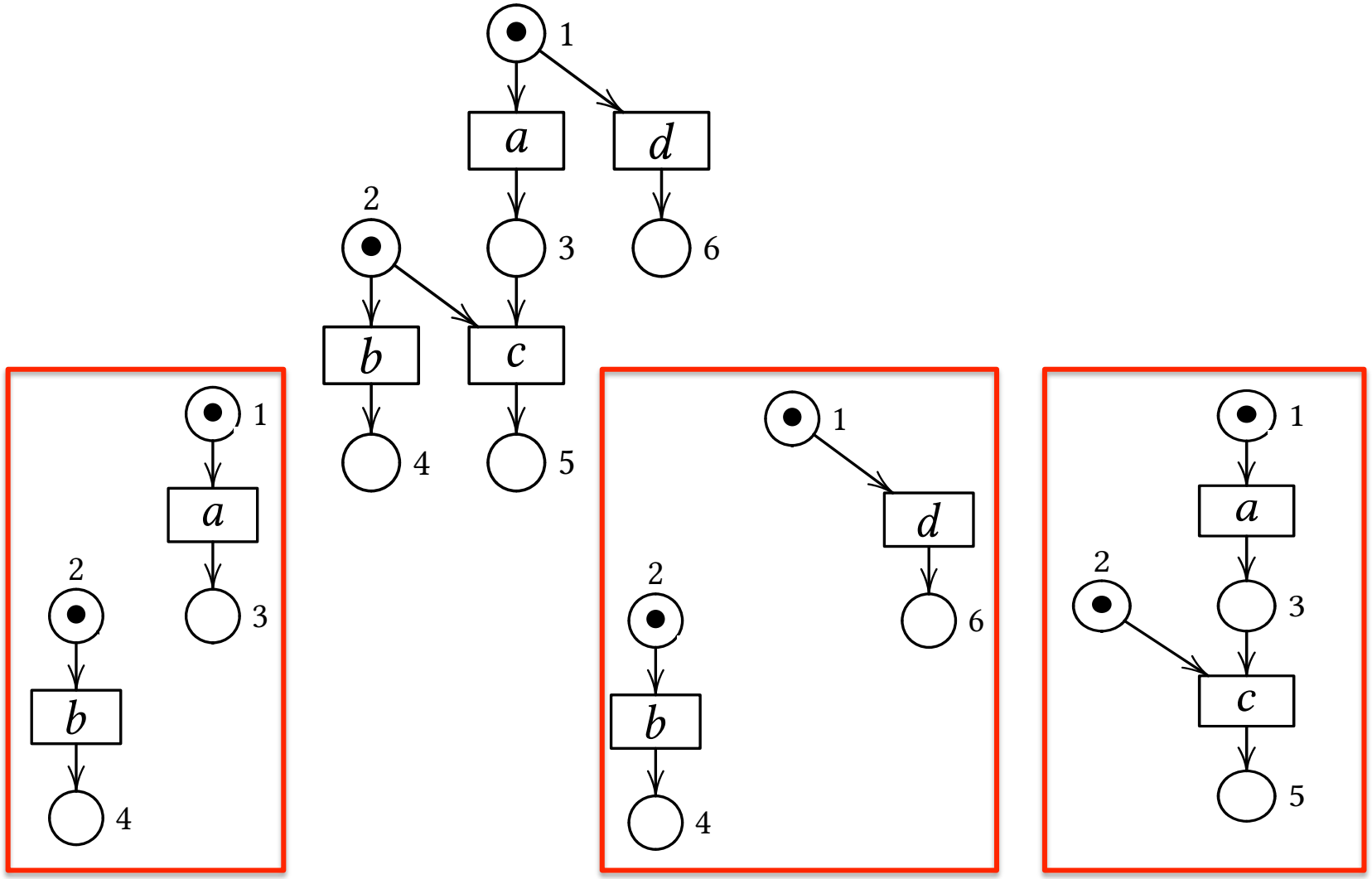
- ordinary automata
  - single point of decision:
  - probabilities attached to arcs leaving the same state
- Petri nets
  - states and decisions are distributed:
  - what is a decision point?
- easy for special nets
  - free-choice nets
    - presets of any two transitions either disjoint or equal,
  - confusion-free nets
    - no alternatives created/deleted by concurrent transitions

Occurrence Nets: An Example

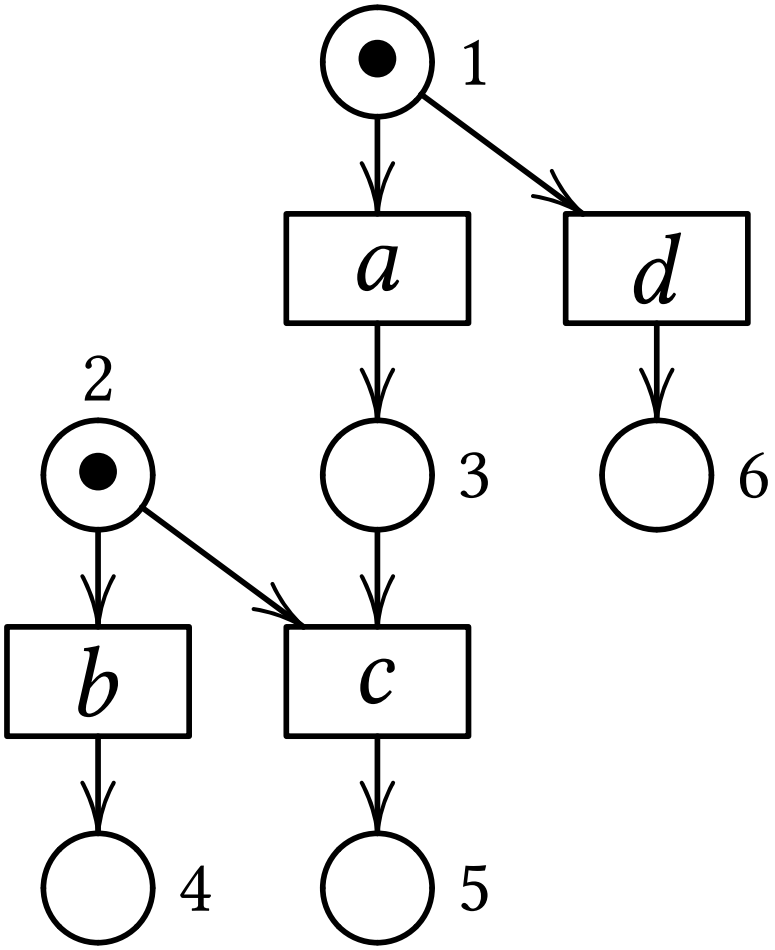


- ON are unfoldings of cyclic nets
- places have at most one input arc
- multiple output arcs from places represent choices
- 1-safe: at most one token per place
- nondeterministic behavior

Deterministic Processes



Confusion: An Example



- a and b are concurrent
- ab and ba are equivalent

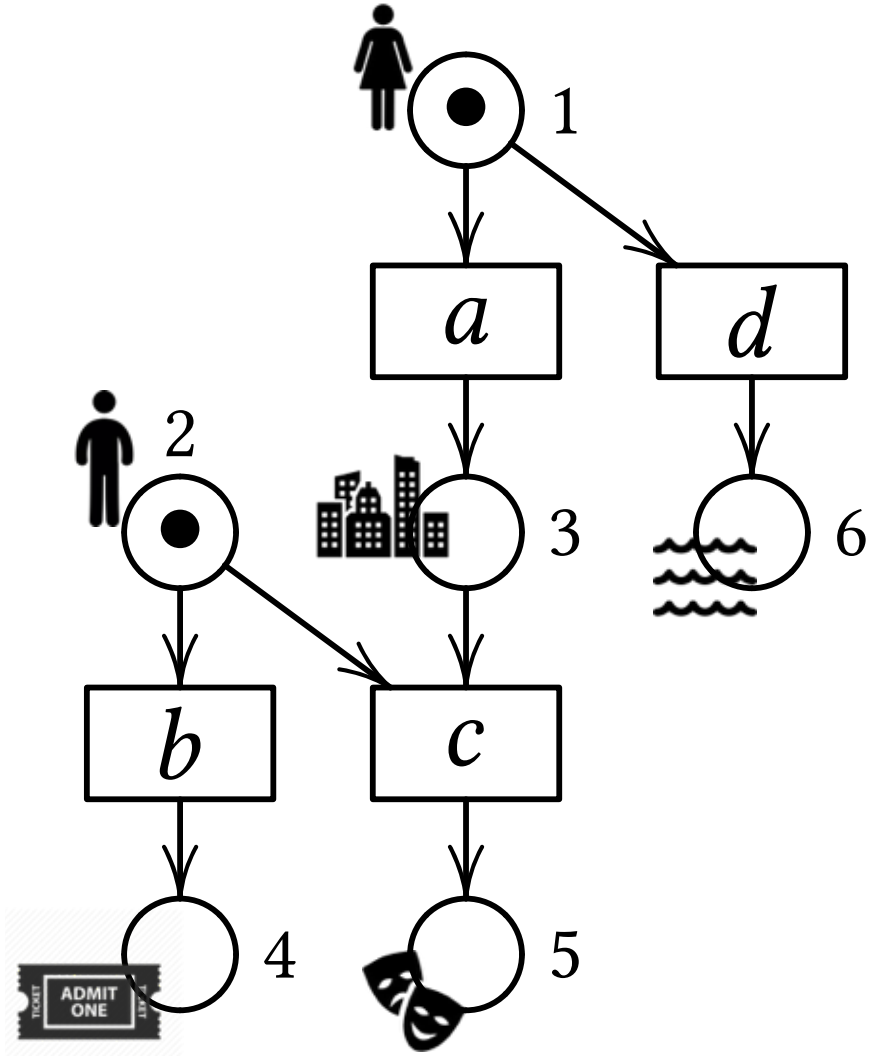
- but:

- ab choses
  - a over d
- c becomes executable
  - b over c

- ba choses
- no choice for b
  - a over d

- ?! ba forbidden?

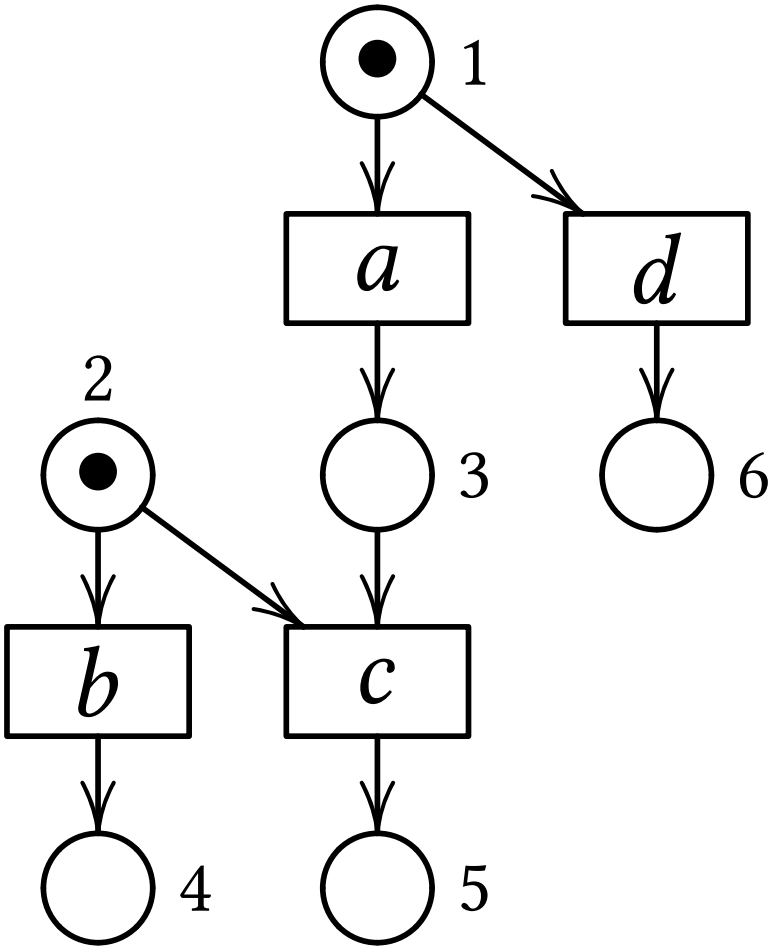
Confusion: The Solution



b is **not** concurrent w.r.t. a and d

the decision to fire b better be postponed after a or d

Abbes & Benveniste Executions



- partially ordered branching cells
  - transitive closure of transitions wrt.
    - causality, mutual exclusion
  - equivalence classes are BC
- => decision points
- new cells may appear

- {a,d} ⊑ {b,c}
- {b,c} cannot be executed
- if a is chosen,
  - cell {b,c} is left
- if d is chosen
  - a and c disappear
  - new cell {b} appears

Roadmap

- Concurrency: a useful abstraction level
- Equivalent computations may have different decision points and different probabilities
- Petri occurrence nets with confusion
- Our result: compiling a net with confusion into one without confusion
- Additional causal links for transmitting negative conditions
- The resulting net is a net with persistence for handling OR causality
- Conclusion and future work

Our Aim

- pure probabilistic model: no nondeterminism, no optimal scheduler
- speed independence: no stochastic component
- concurrent choices: they must be independent
- complete concurrency: all and only the linearizations of the partial ordering of causes are executable
- concurrency is a correct abstraction: probability of a concurrent deterministic computation is independent from the order of execution
- probabilities sum to 1: the sum of the probabilities assigned to all deterministic processes is 1

Our Contribution I

- generic occurrence net => confusion-free net
- modular construction in three phases
  - build structural branching cells (s-cells)
    - static, hierarchical, compositional vs. A&B dynamic
  - from s-cells to dynamic nets
    - certain transitions are dynamically generated
  - from dynamic nets to nets with persistence
    - certain places, when full, cannot become empty
- recover A&B, but: they interpret, we compile

Our Contribution II

Dynamic nets:

- Asperti & Busi
- certain transitions are dynamically generated

Nets with persistency

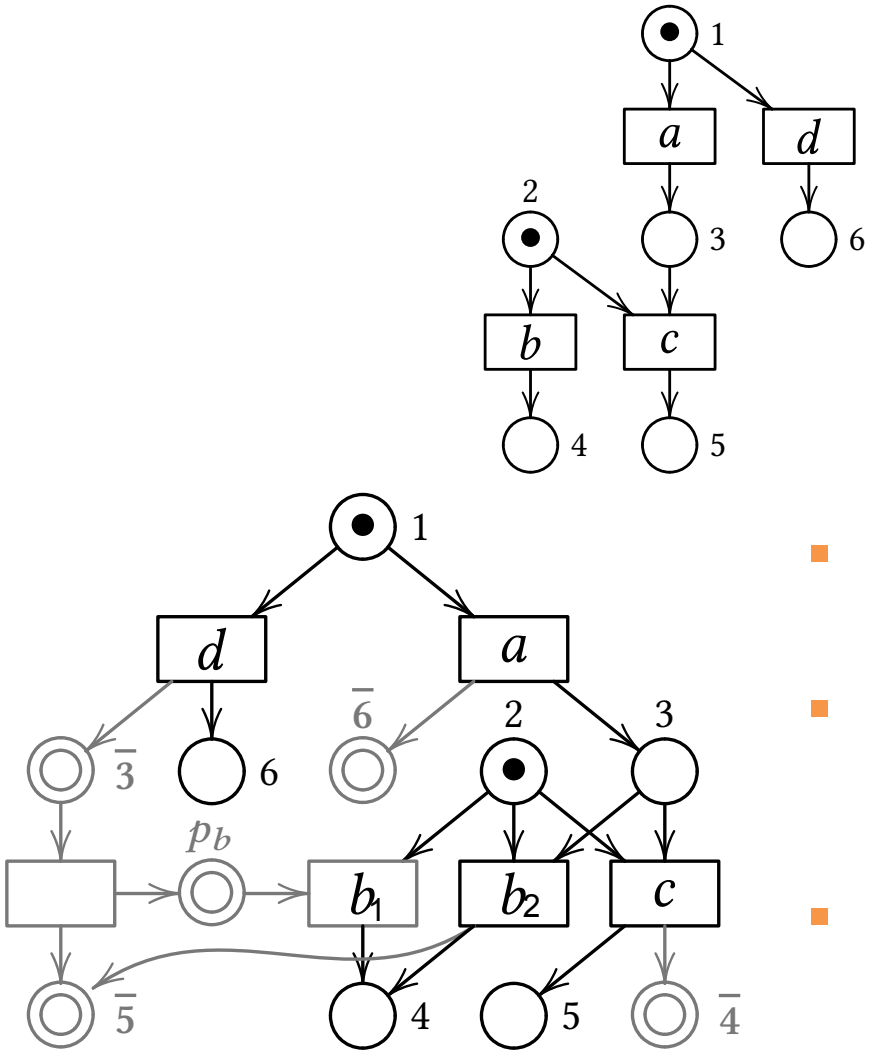
- Crazzolara & Winskel
  - tokens in a persistent place
    - are indistinguishable one from the other (collective)
    - cannot be consumed
    - a token carries infinite weight

- dynamics nets: a commodity
- nets with persistency: a necessity

Roadmap

- Concurrency: a useful abstraction level
- Equivalent computations may have different decision points and different probabilities
- Petri occurrence nets with confusion
- Our result: compiling a net with confusion into one without confusion
- Additional causal links for transmitting negative conditions
- The resulting net is a net with persistence for handling OR causality
- Conclusion and future work

Our Contribution III



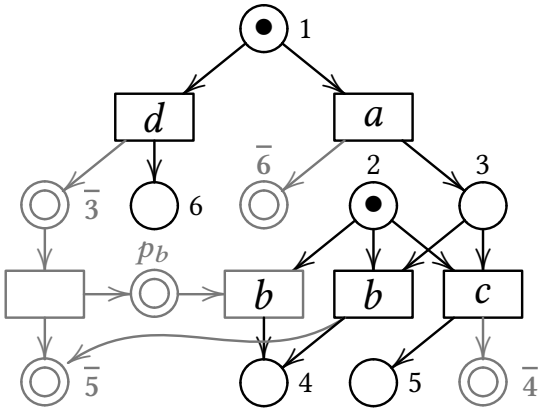
forbid unwanted transitions

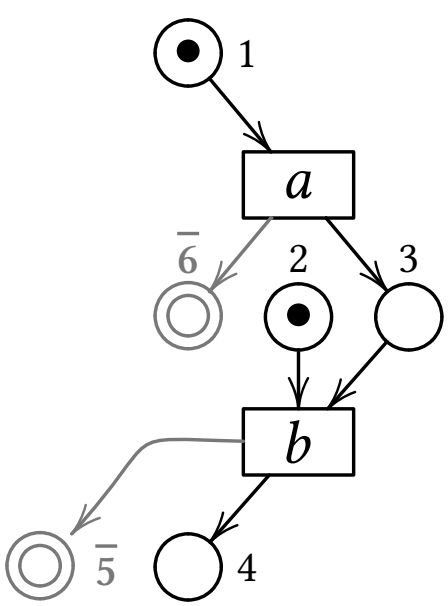
additional causal links

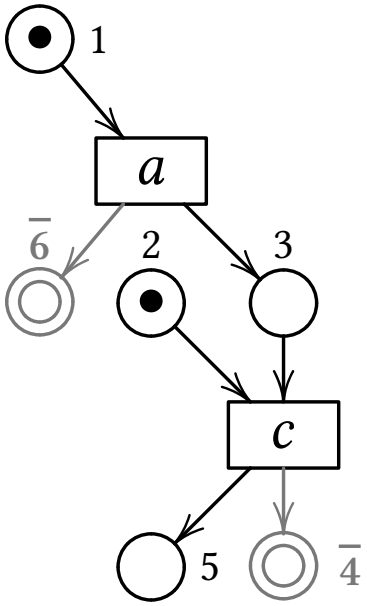
places w. negative information

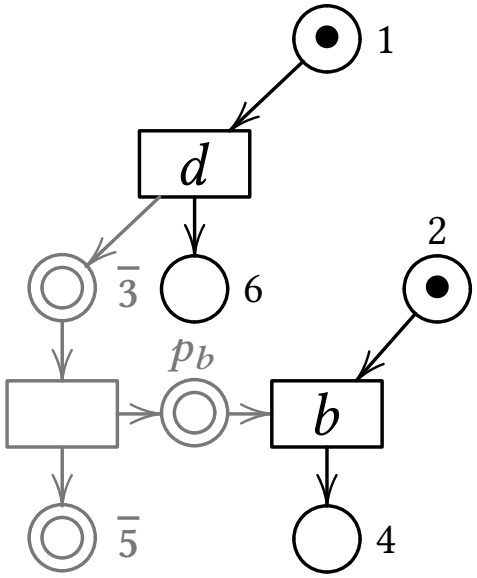
- at the beginning b_1 and b_2 are not enabled
- {a,d} cell: if d is executed
  - ¬3 is activated
  - b_1 is enabled via p_b: no alternatives
- if a is executed
  - b_2 and c are both enabled (exclusively)
  - here b_2 is an alternative to c

The Deterministic Processes





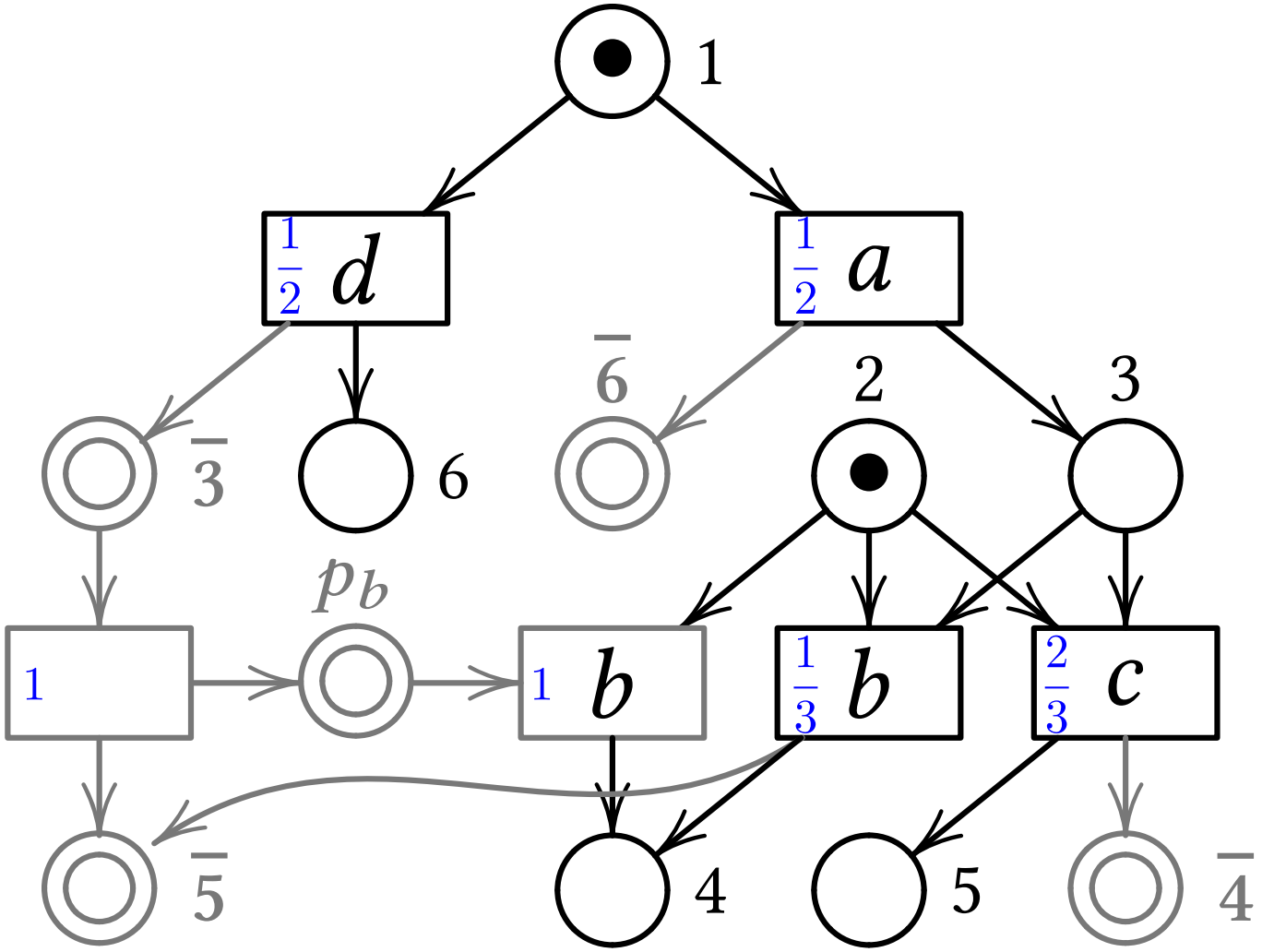




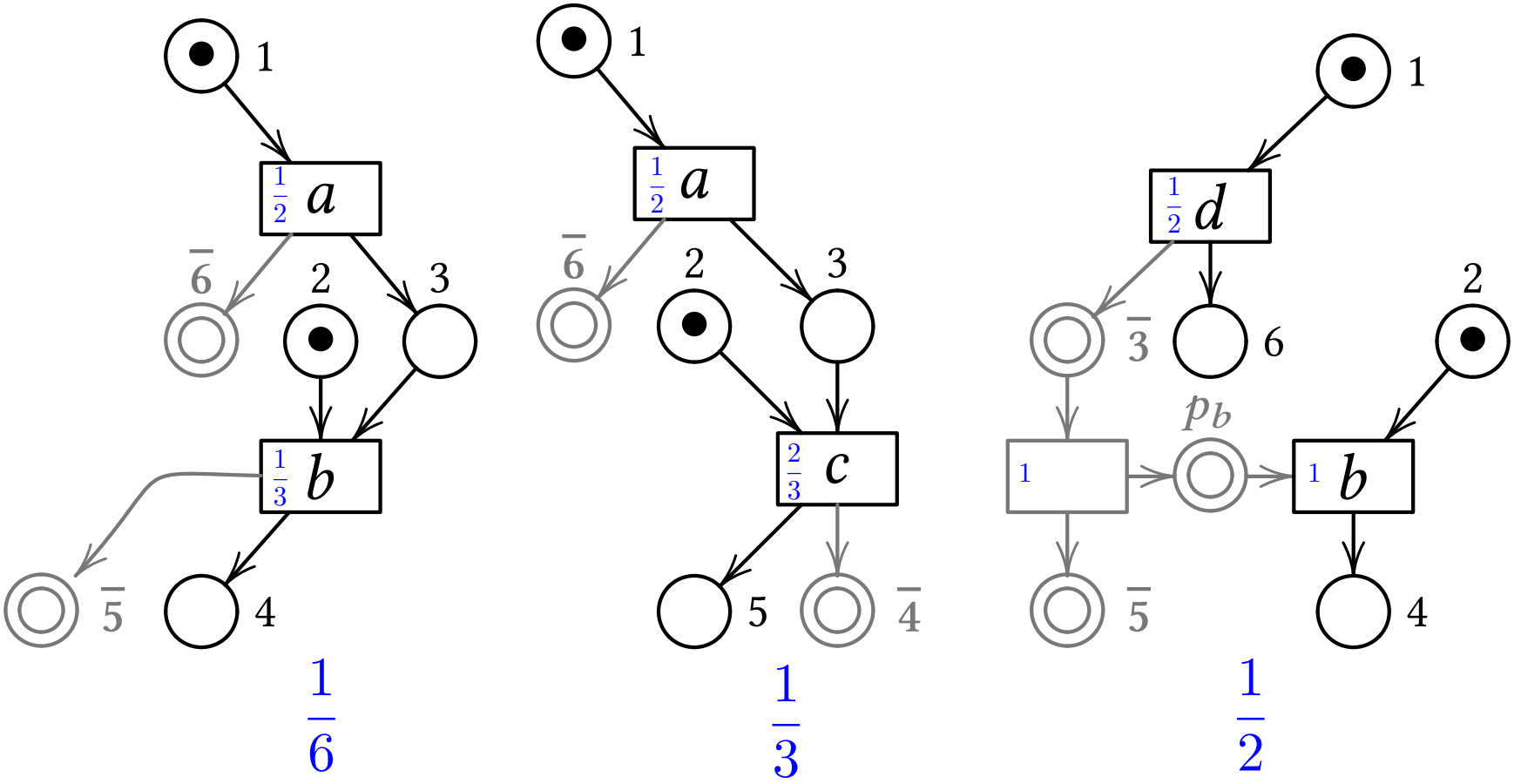
Probability I

- assign arbitrary probability distributions to decision arcs outgoing the same non persistent places
- transitions
  - auxiliary: probability 1
  - ordinary: product of probabilities on incoming arcs
    - normalized w.r.t. all alternatives in the same s-cell
- probability of a process: product of its transition probabilities

Example I


1
$\frac{1}{2}$ d
$\frac{1}{2}$ a
$\overline{6}$
2
3
$\overline{3}$
6
p_b
1
1 b
$\frac{1}{3}$ b
$\frac{2}{3}$ c
$\overline{5}$
4
5
$\overline{4}$

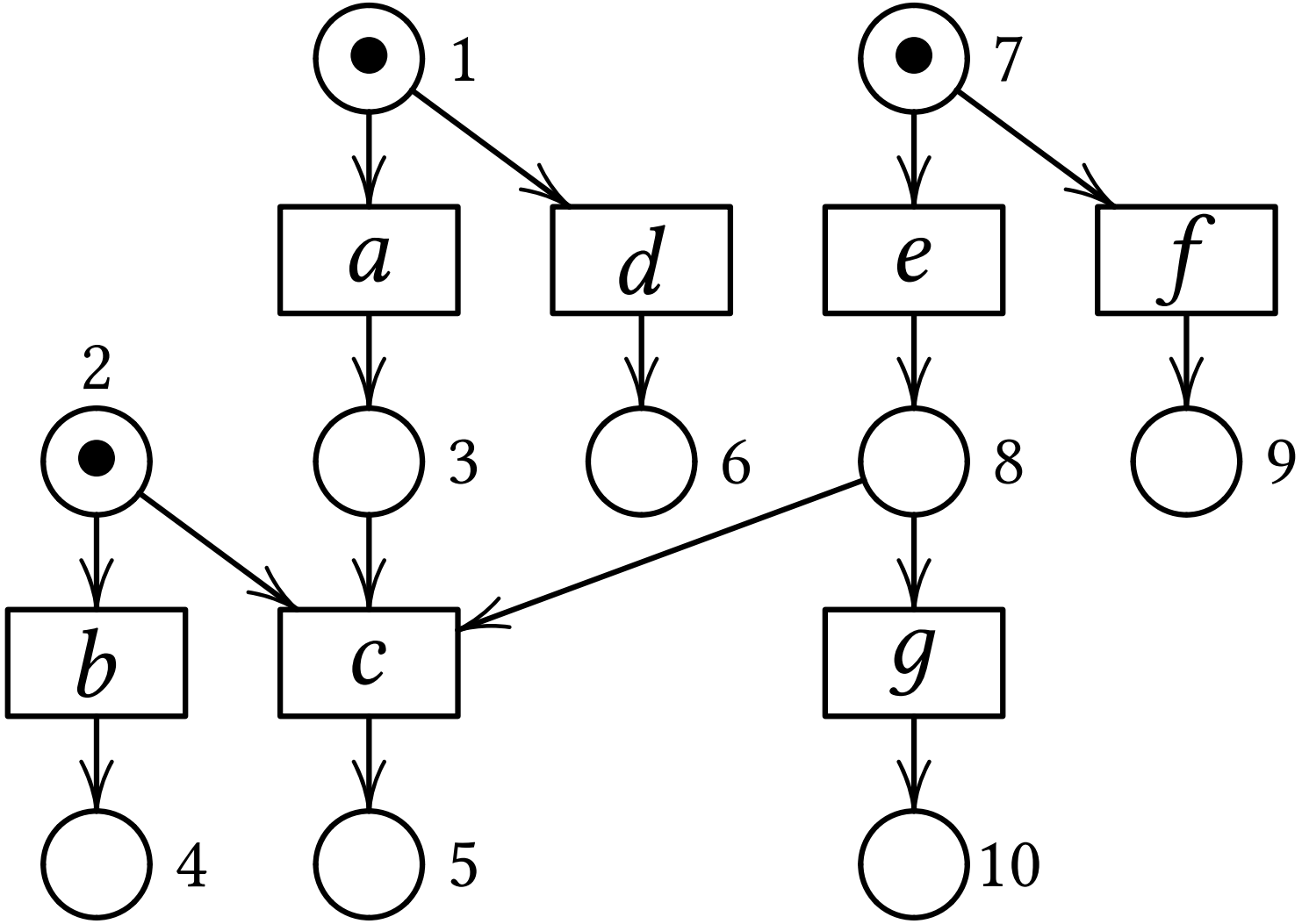
Example II


1
$\frac{1}{2}$ a
$\overline{6}$
2
3
$\frac{1}{3}$ b
$\overline{5}$
4
$\frac{1}{6}$
1
$\frac{1}{2}$ a
$\overline{6}$
2
3
$\frac{2}{3}$ c
5
$\overline{4}$
$\frac{1}{3}$
1
$\frac{1}{2}$ d
$\overline{3}$
6
2
p_b
1
1 b
$\overline{5}$
4
$\frac{1}{2}$

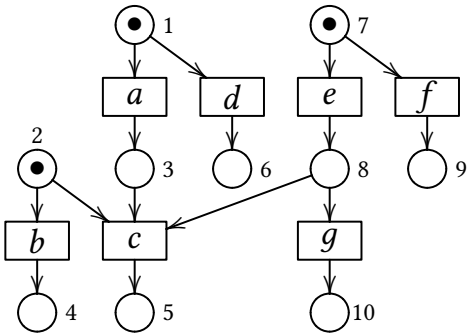
Roadmap

- Concurrency: a useful abstraction level
- Equivalent computations may have different decision points and different probabilities
- Petri occurrence nets with confusion
- Our result: compiling a net with confusion into one without confusion
- Additional causal links for transmitting negative conditions
- The resulting net is a net with persistence for handling OR causality
- Conclusion and future work

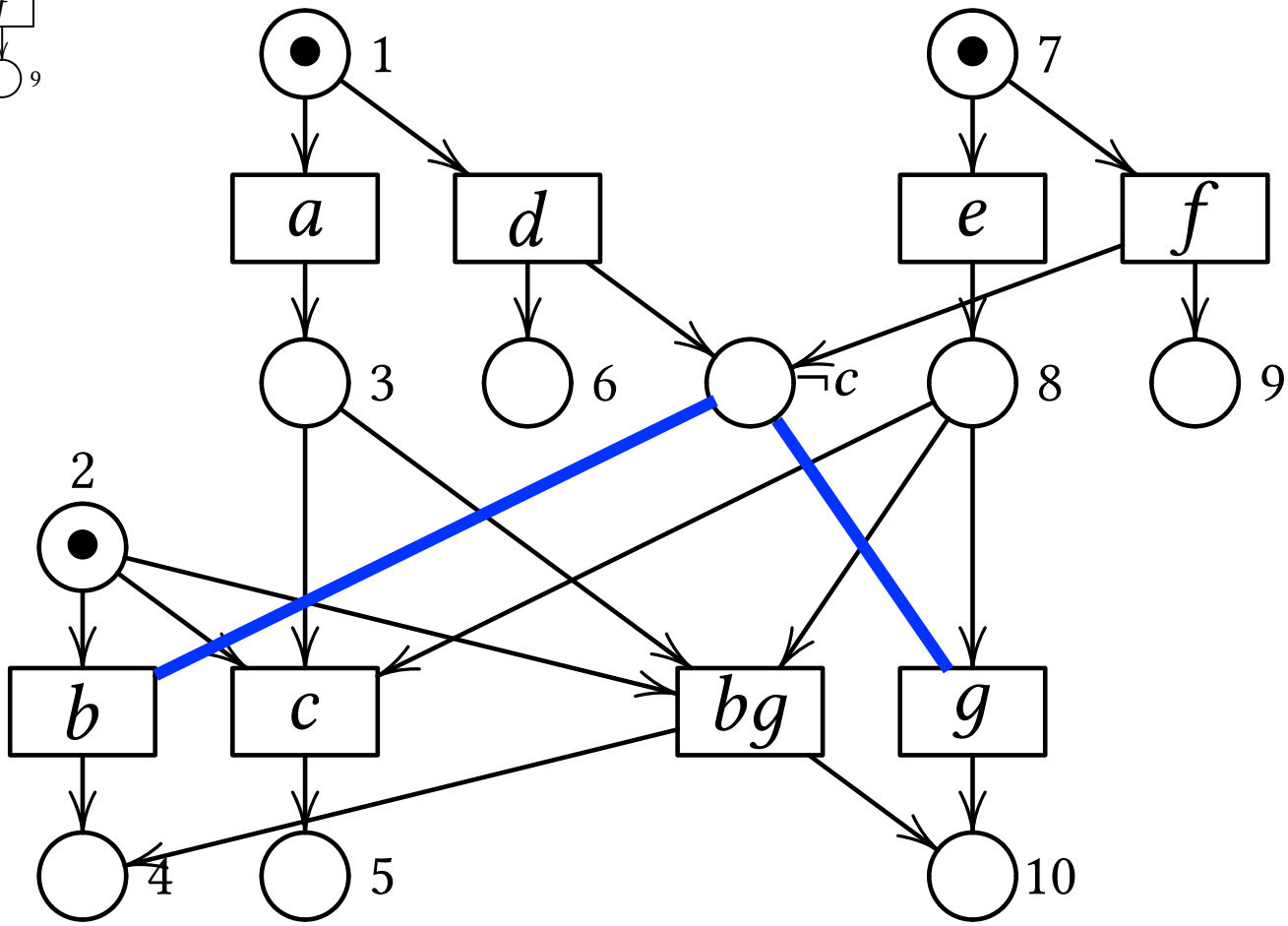
OR-Dependencies I



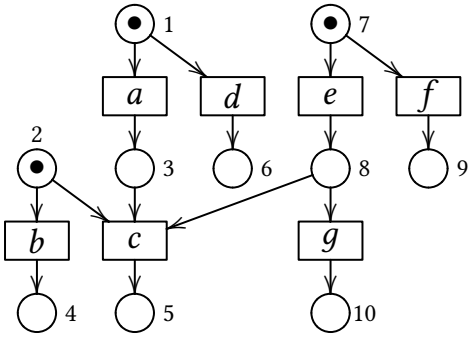
OR-Dependencies III



read arcs: the token is read but not consumed

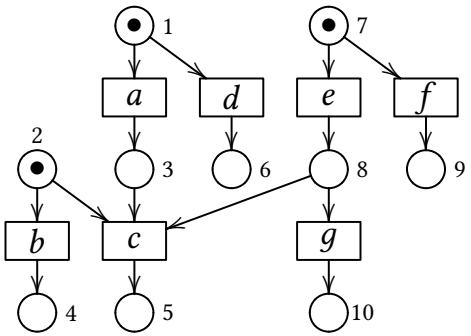


OR-Dependencies IV



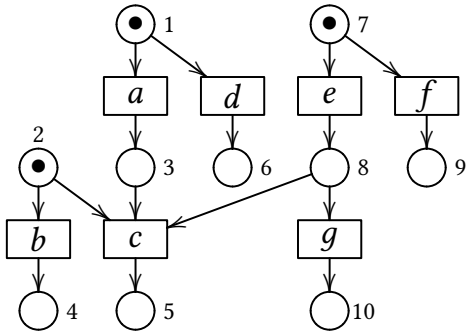
multiple incoming arcs: c **discarded** after the firing of d **or** f

OR-Dependencies V



in a process
b depends
either on d **or** f

OR-Dependencies VI



concurrency exibited by the persistent net:

e.g. after t_f, t_d and t_b can be executed concurrently

Roadmap

- Concurrency: a useful abstraction level
- Equivalent computations may have different decision points and different probabilities
- Petri occurrence nets with confusion
- Our result: compiling a net with confusion into one without confusion
- Additional causal links for transmitting negative conditions
- The resulting net is a net with persistence for handling OR causality
- Conclusion and future work

Conclusion and Future Work I

Our results

- compile an ordinary occurrence net in a statically defined, confusion-free, persistent net exhibiting true concurrency

Future work

- extending the construction to cyclic, non-occurrence nets
- exploiting concurrency in transactions
- complexity analysis
- event structures and domains

Event Structures and Domains for Persistent Nets

- Results in LICS 2017 by Baldan, Corradini and Gadducci about coreflection/equivalence of graph transformations with fusions
- They apply not only to graph fusions but also to fusions of past histories for persistent places of persistent nets
- Functorial relations: nets ↔ event structures ↔ domains are fully extended
  - unfolding persistent nets is a coreflection
  - there is a coreflection between nonprime (OR) connected event structures and persistent occurrence nets
  - configurations are executions in a weak prime domain
  - there is an equivalence between weak prime domains and connected ES