

A structure for examining properties of decentralized DES



(One structure to join them all...)

Notation (wrt transition function δ)

- Finite state machine:

$$M = (Q, \Sigma, \delta, q_0, F)$$

- System requiring attention denoted

$$M_L = (Q, \Sigma, \delta, q_0, F_L)$$

- Specification denoted by

$$M_K = (Q, \Sigma, \delta, q_0, F_K)$$

Notation (wrt transition set T)

- Finite state machine:

$$M = (Q, \Sigma, T, q_0, F^T)$$

- System requiring attention denoted

$$M_L = (Q, \Sigma, T, q_0, F_L^T)$$

- Specification denoted by

$$M_K = (Q, \Sigma, T, q_0, F_K^T)$$

Introduction of the structure

S.L. Ricker and B. Caillaud. “Mind the Gap: Expanding communication options for decentralized discrete-event control”. In *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, LA, pp. 5924–29.

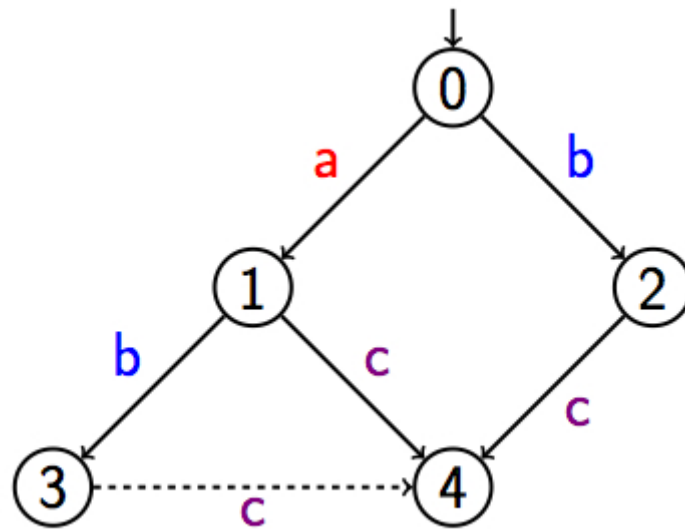
S.L. Ricker. “Asymptotic Minimal Communication for Decentralized Discrete-Event Control”. In *Proceedings of the 9th International Workshop on Discrete-Event Systems*, Gotenberg, Sweden, pp 486–91.

CONTEXT: synchronous communication

What has gone before...

- Rudie and Willems (1995) presented an automaton for verifying the property of co-observability.
- For n decentralized controllers
 - Non-deterministic automaton \mathbf{A} is the product of $n+1$ copies of M_K and 1 copy of M_L
 - Tracks sequences s , s' and s'' plus evolution of plant
 - Alphabet of \mathbf{A} is simply alphabet of M_L

Example: verifying co-observability



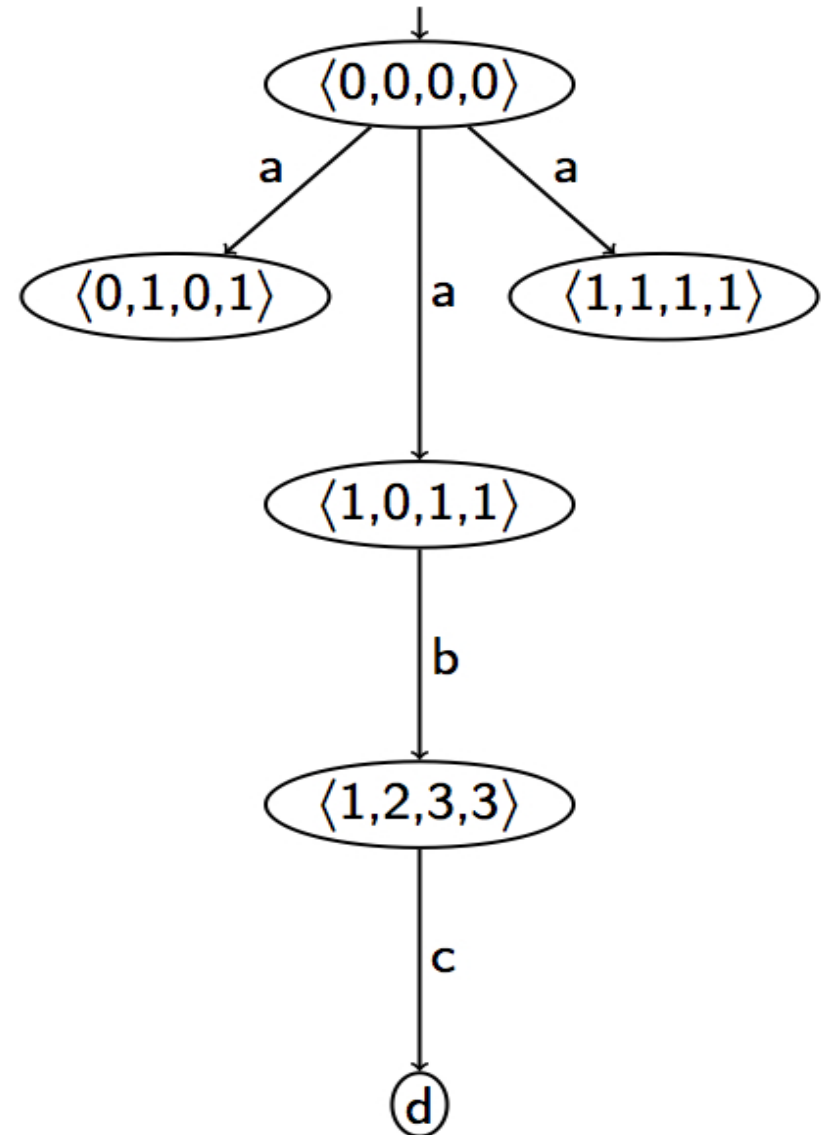
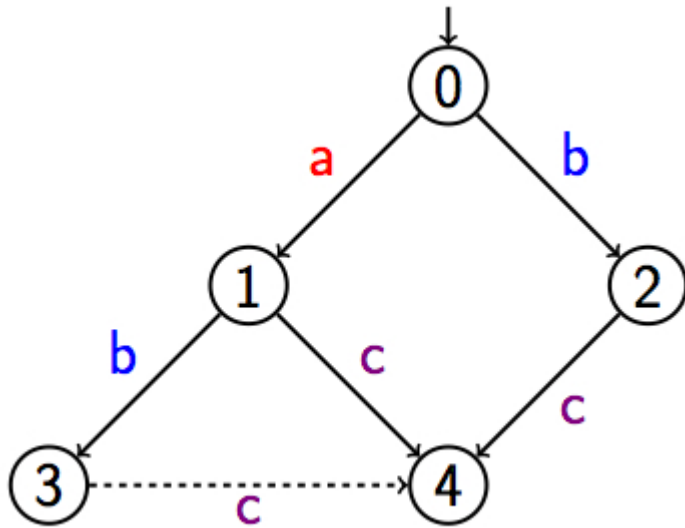
$$\Sigma_{o,1} = \{a\}$$

$$\Sigma_{o,2} = \{b\}$$

$$\Sigma_{uo} = \{c\}$$

$$\Sigma_{c,1} = \Sigma_{c,2} = \{c\}$$

Example: verifying co-observability



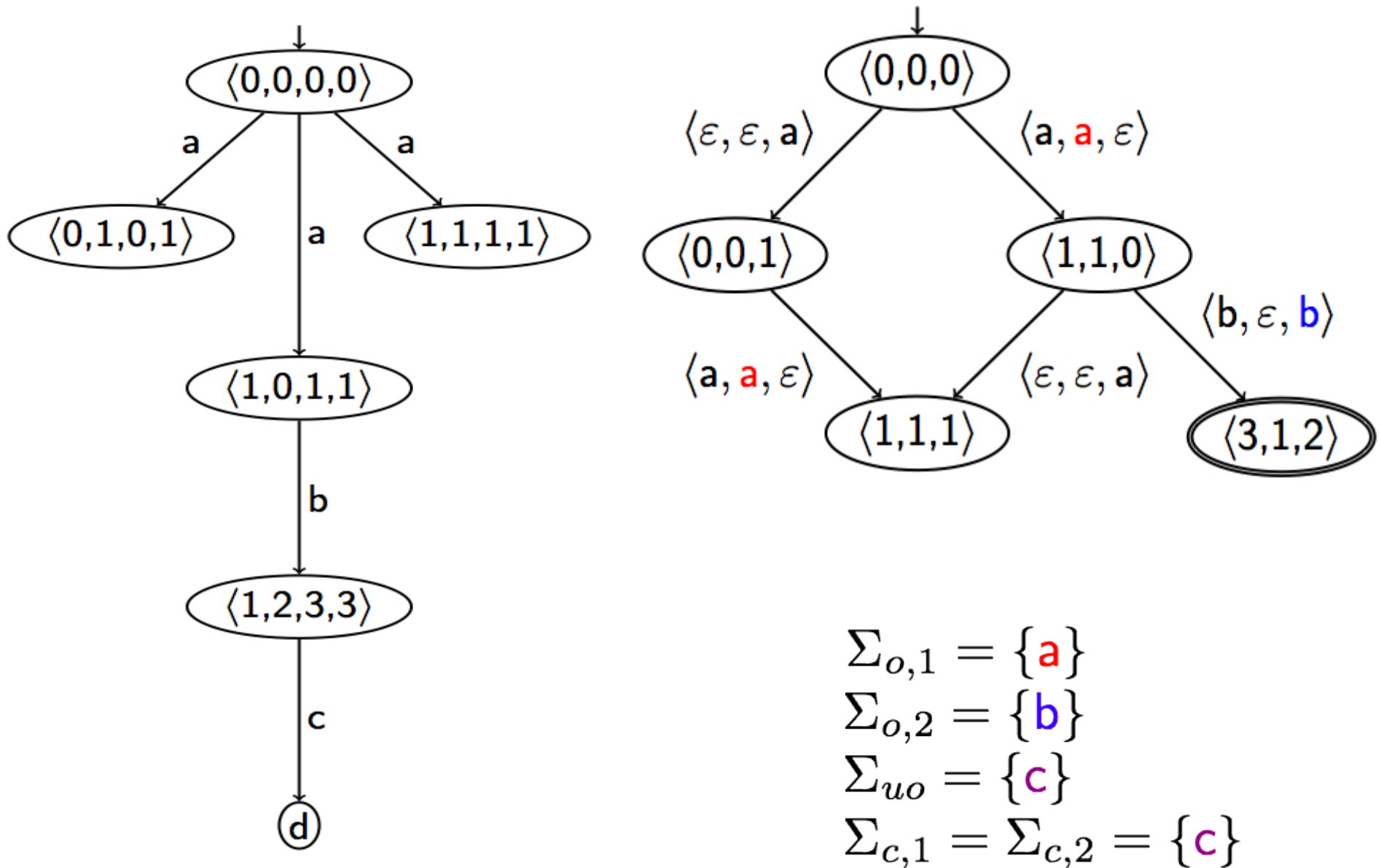
$$\Sigma_{o,1} = \{a\}$$

$$\Sigma_{o,2} = \{b\}$$

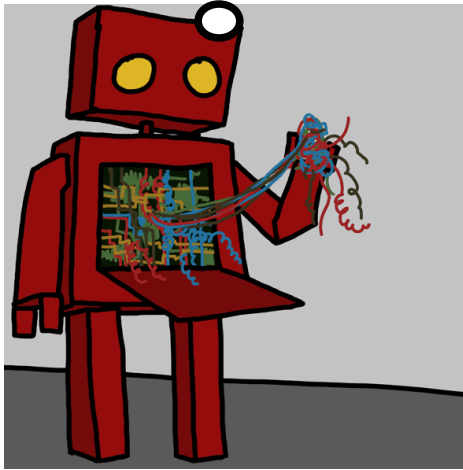
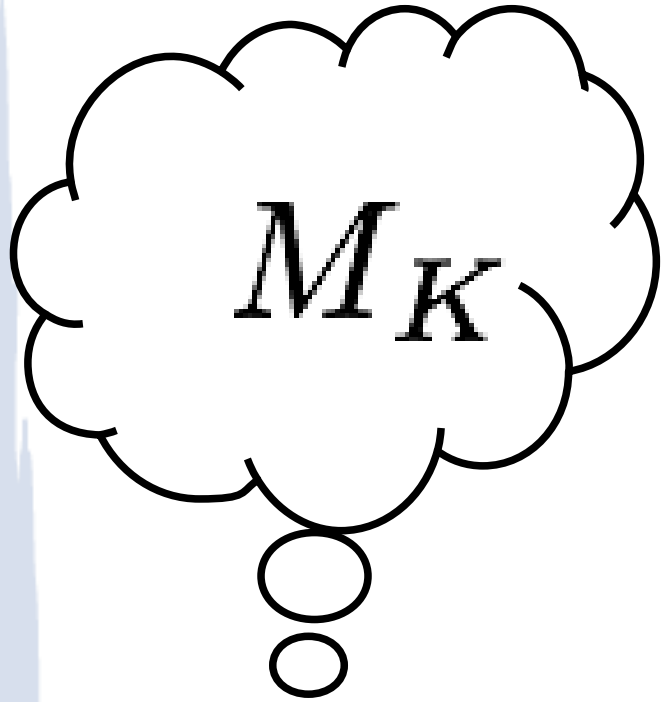
$$\Sigma_{uo} = \{c\}$$

$$\Sigma_{c,1} = \Sigma_{c,2} = \{c\}$$

Sneak Peak: Old vs. U



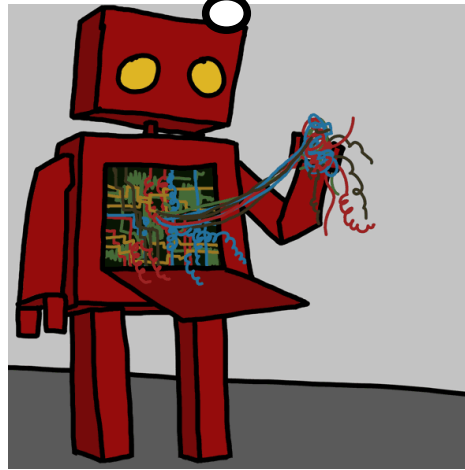
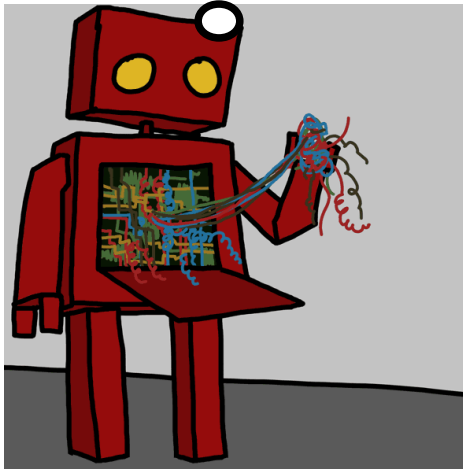
Constructing U



Constructing U

M_K

M_K

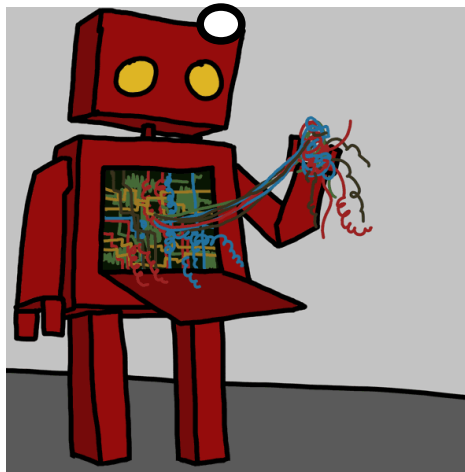


Constructing U

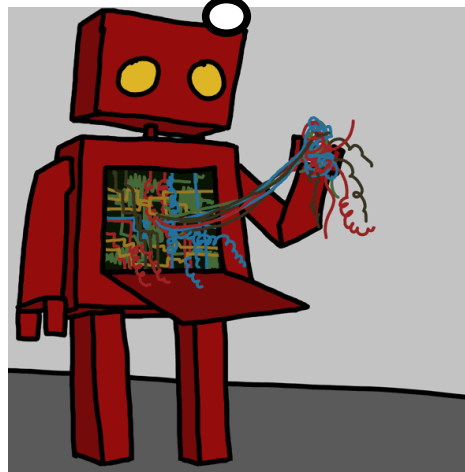
M_K

M_K

M_K

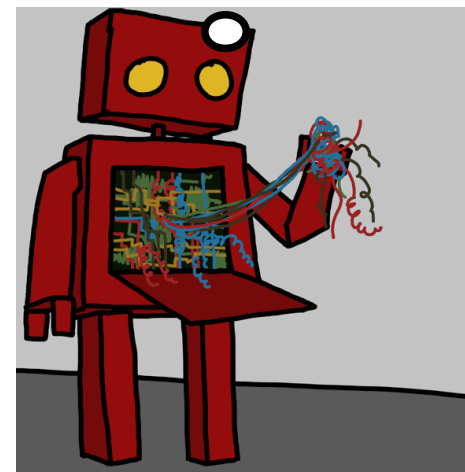


1



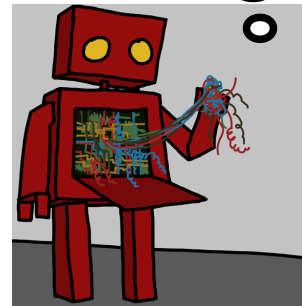
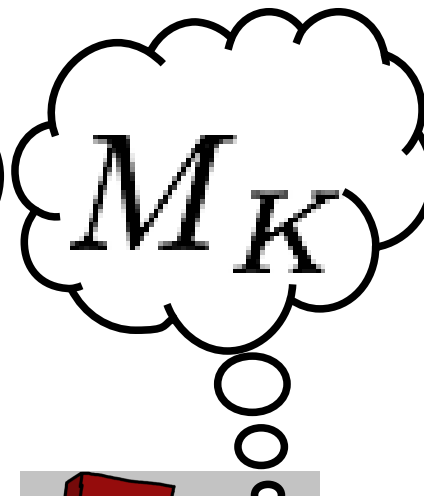
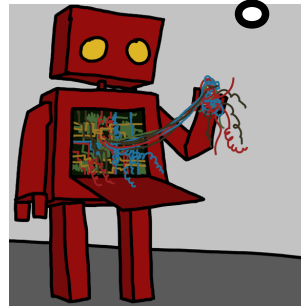
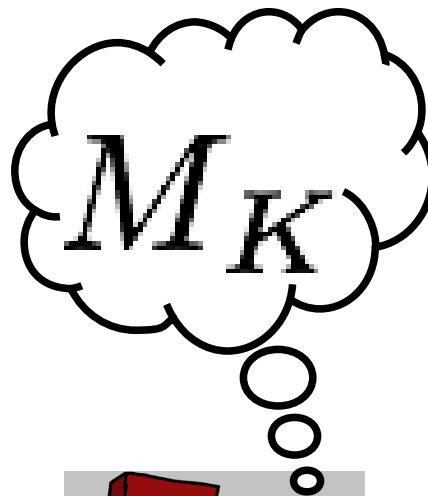
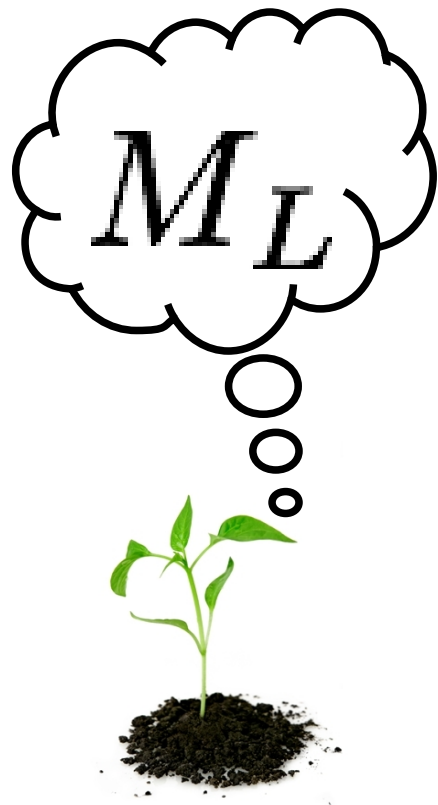
2

...

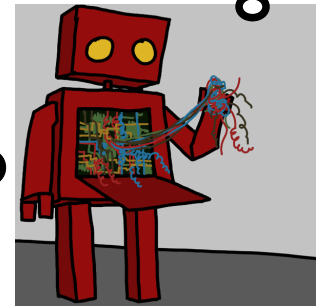
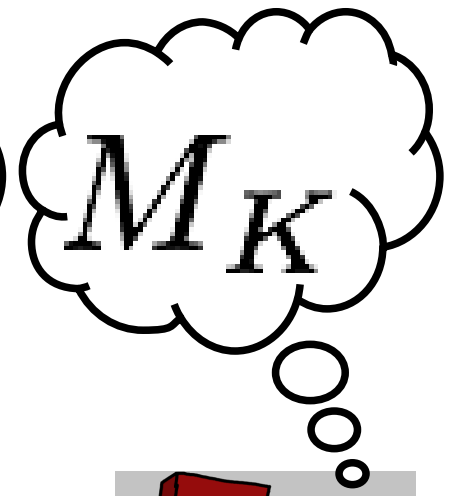


n

Constructing U



...



State Structure of U

- State space

$$Q^U \subseteq Q^{n+1}$$

- We avoid the computation of $n+2$ copies of Q because of the alphabet we choose for reasoning about decentralized properties of interest
 - Synchronization vectors (A. Arnold, 1994)

Alphabet of U: synchronization vectors

- Construct basic vectors to capture observable and unobservable events (assume $n = 2$ here)

- Unobservable events

$$\sigma \in \Sigma_{uo} \Rightarrow \langle \sigma, \varepsilon, \varepsilon \rangle$$

$$\sigma \in \Sigma_{uo,1} \Rightarrow \langle \varepsilon, \sigma, \varepsilon \rangle$$

$$\sigma \in \Sigma_{uo,2} \Rightarrow \langle \varepsilon, \varepsilon, \sigma \rangle$$

- Observable events

$$\sigma \in \Sigma_{o,1} \setminus \Sigma_{o,2} \Rightarrow \langle \sigma, \sigma, \varepsilon \rangle$$

$$\sigma \in \Sigma_{o,2} \setminus \Sigma_{o,1} \Rightarrow \langle \sigma, \varepsilon, \sigma \rangle$$

$$\sigma \in \Sigma_{o,1} \cap \Sigma_{o,2} \Rightarrow \langle \sigma, \sigma, \sigma \rangle$$

Alphabet of U: synchronization vectors

- Also need to introduce the empty vector

$$\langle \varepsilon, \varepsilon, \varepsilon \rangle$$

- Can define a partial order on the vectors
 - Least upper bound

$$\forall i \in I_0, (\ell_1 \vee \ell_2)(i) = \begin{cases} \ell_1(i), & \text{if } \ell_1(i) \neq \varepsilon; \\ \ell_2(i), & \text{if } \ell_2(i) \neq \varepsilon; \\ \varepsilon, & \text{otherwise.} \end{cases}$$

- Compatible vectors

$$\ell_1 \uparrow \ell_2, \text{ iff } \forall i \in I_0, \ell_1(i) = \varepsilon \text{ or } \ell_2(i) = \varepsilon \text{ or } \ell_1(i) = \ell_2(i)$$

Alphabet of U : synchronization vectors

- Alphabet of U contains
 - Set of basic vectors
 - Empty vector
 - Least upper bound of all compatible vectors

Alphabet of U : synchronization vectors

$$\Sigma_{uo} = \Sigma_{uo,1} = \Sigma_{uo,2} = \{c\}$$

$$\Sigma_{o,1} = \{a\}$$

$$\Sigma_{o,2} = \{b\}$$

- Set of basic vectors

$$\langle c, \varepsilon, \varepsilon \rangle, \langle \varepsilon, c, \varepsilon \rangle, \langle \varepsilon, \varepsilon, c \rangle, \langle \varepsilon, b, \varepsilon \rangle, \langle \varepsilon, \varepsilon, a \rangle$$
$$\langle a, a, \varepsilon \rangle, \langle b, \varepsilon, b \rangle$$

- Least upper bound of all compatible vectors

$$\langle c, c, \varepsilon \rangle, \langle \varepsilon, c, c \rangle, \langle c, \varepsilon, c \rangle, \langle c, c, c \rangle, \langle b, c, b \rangle, \langle b, b, b \rangle,$$
$$\langle \varepsilon, b, c \rangle, \langle c, b, \varepsilon \rangle, \langle c, b, c \rangle, \langle a, a, c \rangle, \langle a, a, a \rangle, \langle \varepsilon, b, a \rangle,$$
$$\langle \varepsilon, c, a \rangle, \langle c, \varepsilon, a \rangle, \langle c, c, a \rangle, \langle c, b, a \rangle$$

- Empty vector

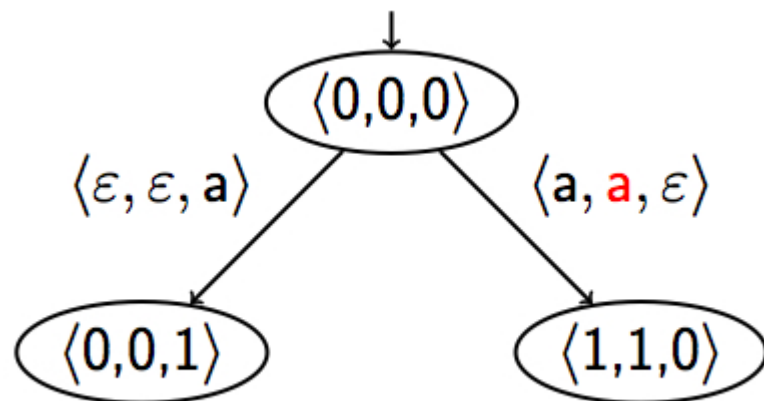
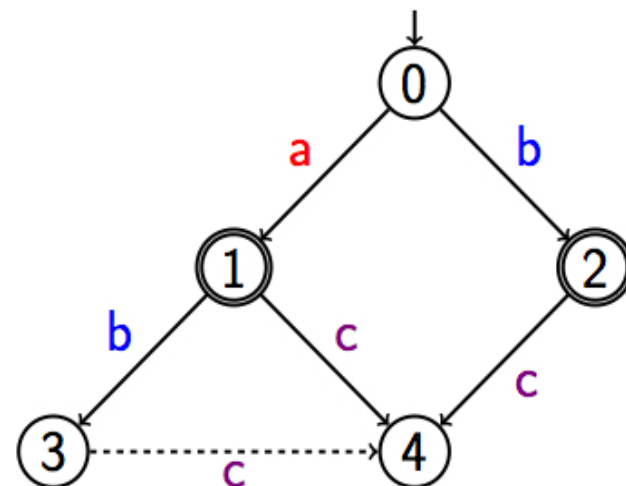
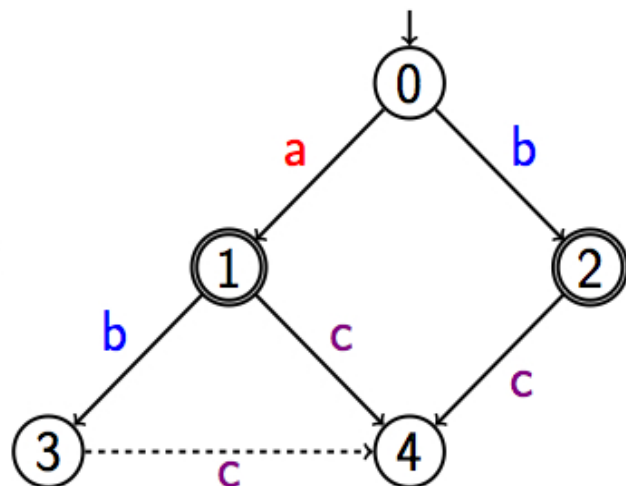
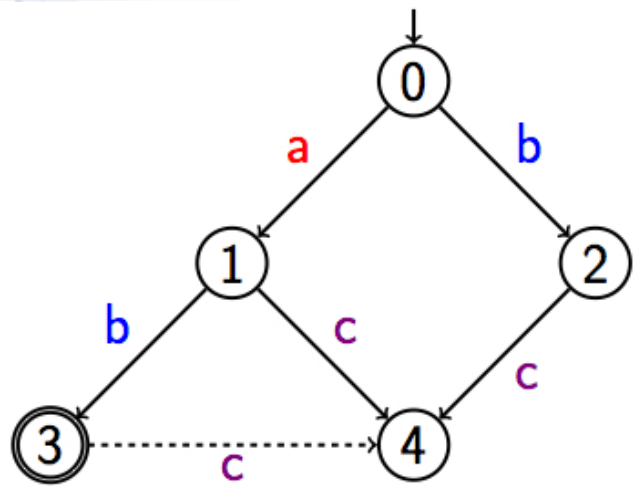
$$\langle \varepsilon, \varepsilon, \varepsilon \rangle$$

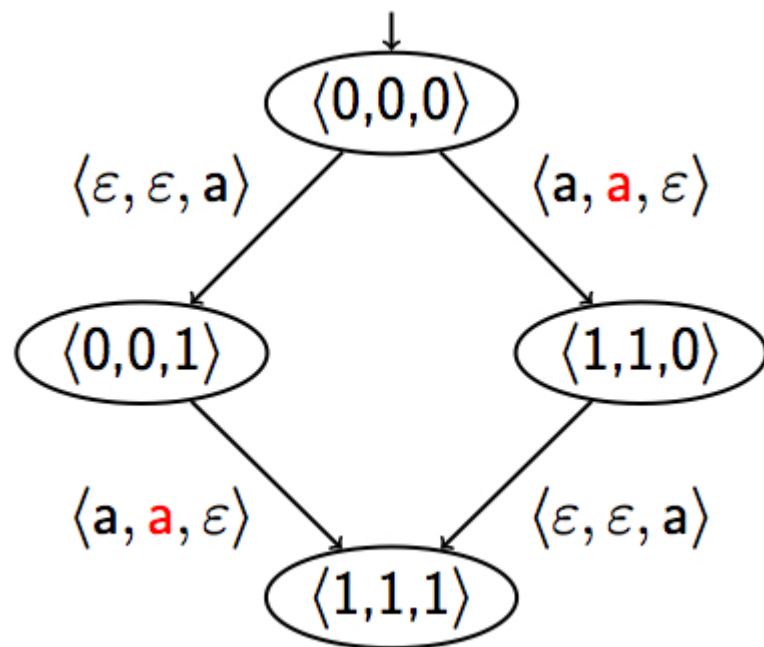
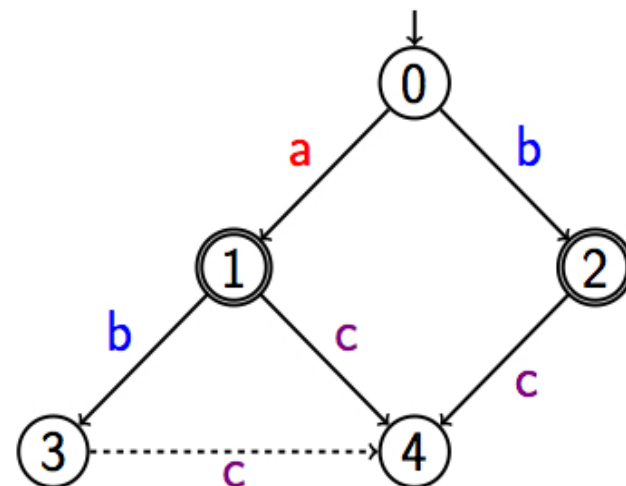
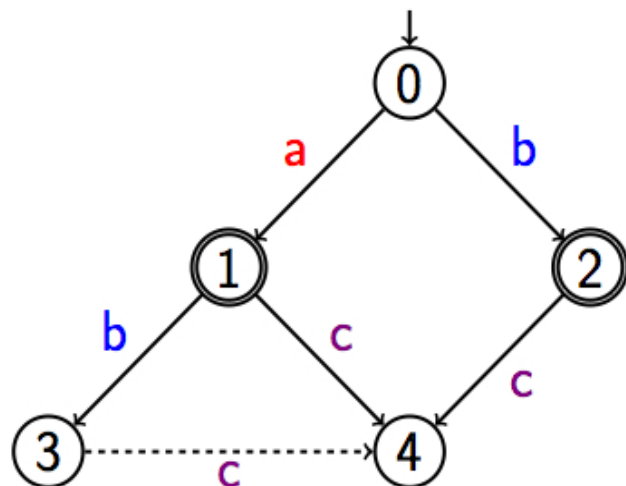
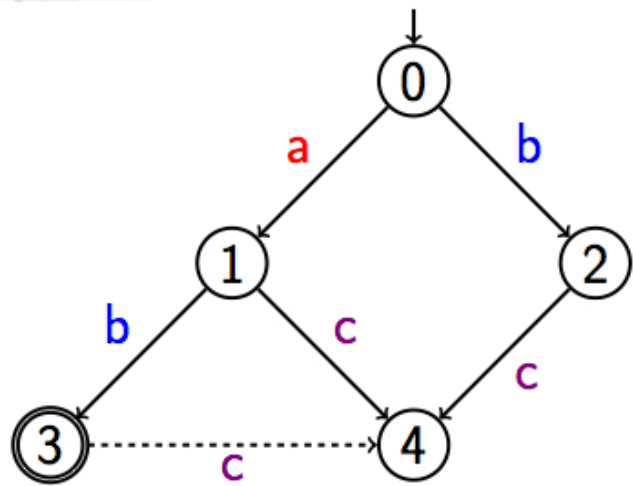
Identifying violations of co-observability with U

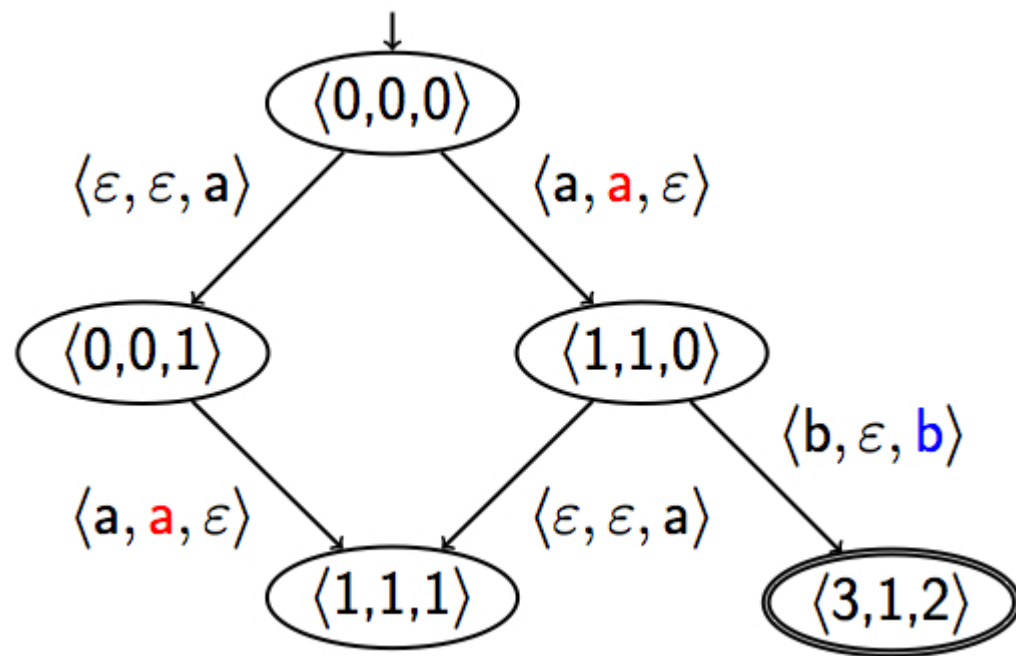
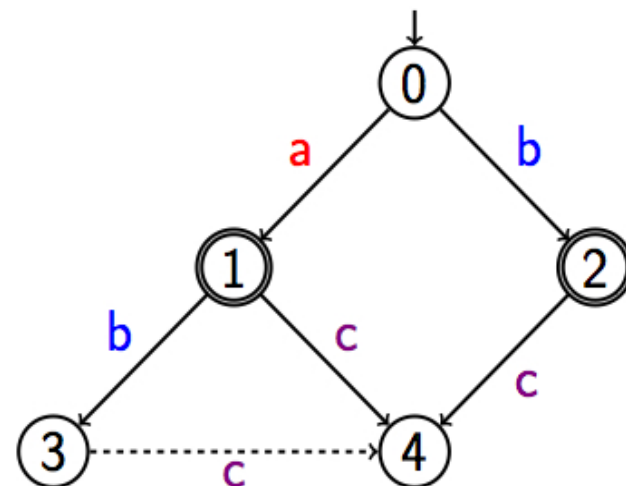
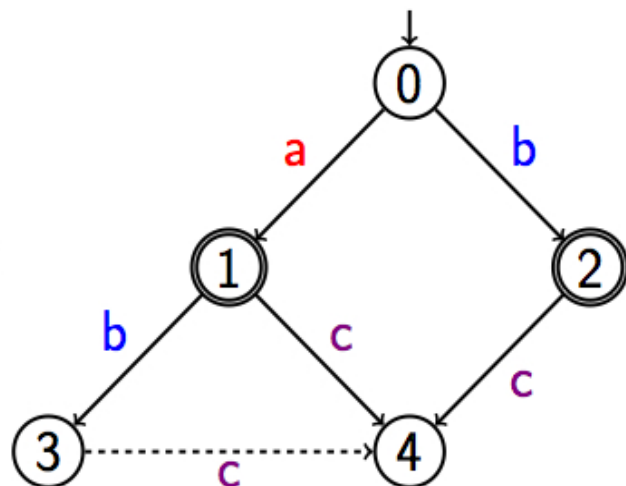
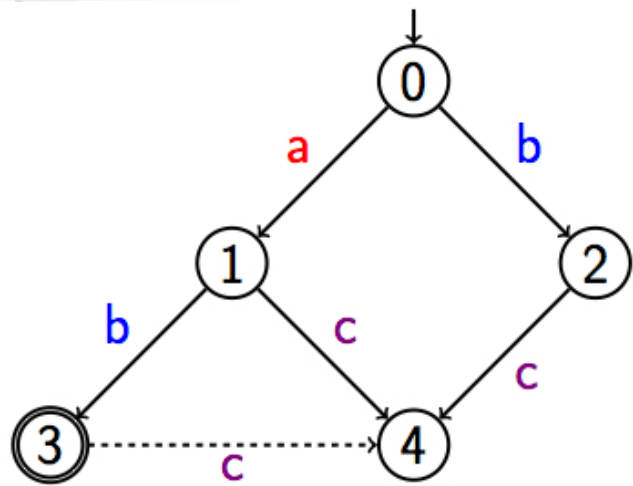
- To deal with co-observability wrt a single event
 - Mark states in M_L where the event must be disabled
 - For all agents that control the event
 - Mark states in their copies of M_K where the event must be enabled
 - Otherwise
 - Mark all states in an agent's copy of M_K

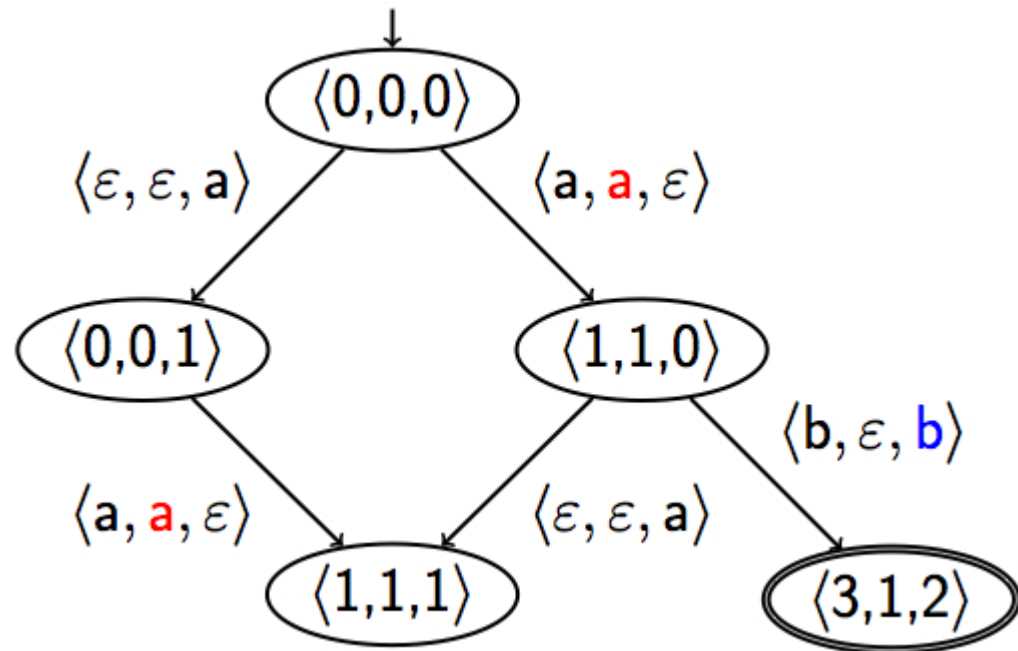
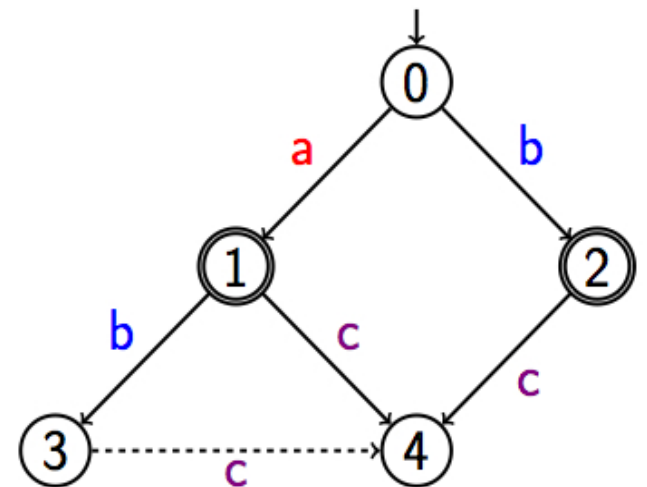
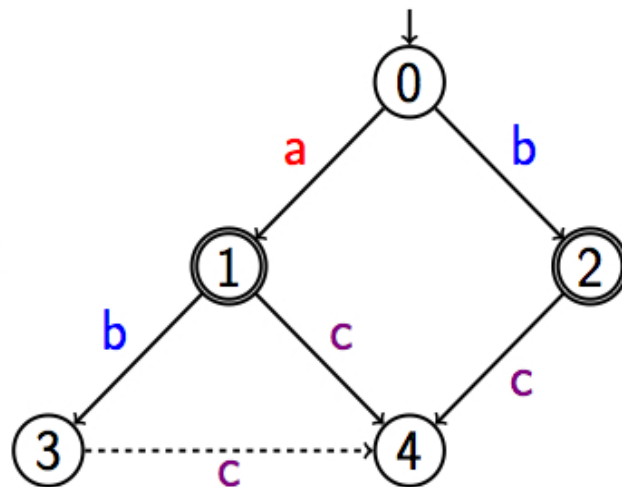
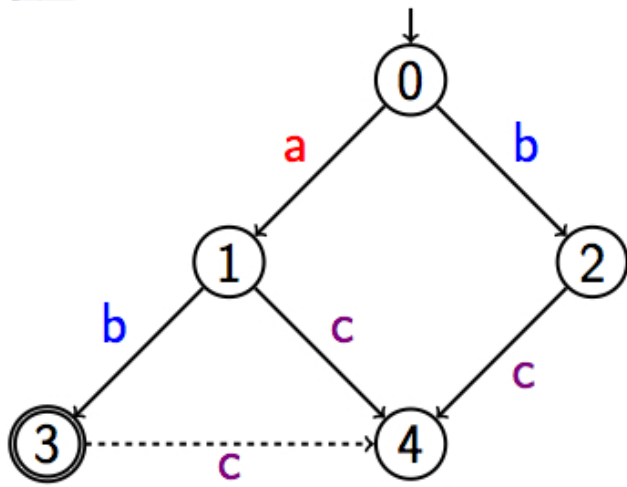
Identifying violations of co-observability with U

- In the product, a state is marked ONLY if each individual state in the aggregated state is marked
- Indicates that the system is in a state where the event of interest must be disabled
- Whereas all the agents that control that event believe that the system could be in a state where that event should be enabled
- Call this an illegal configuration: it represents a violation of co-observability.

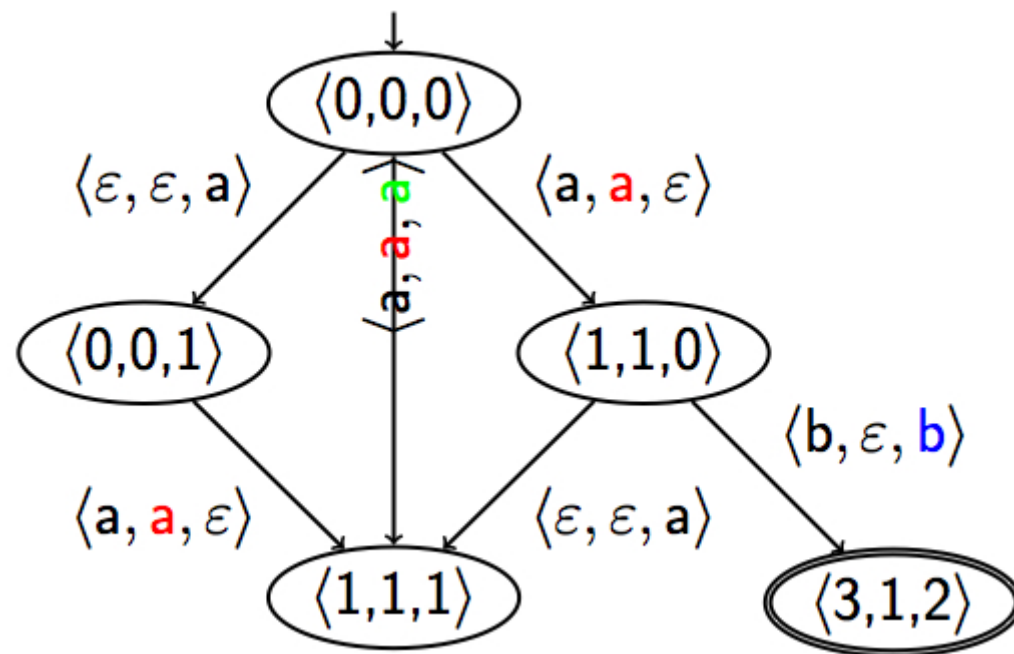
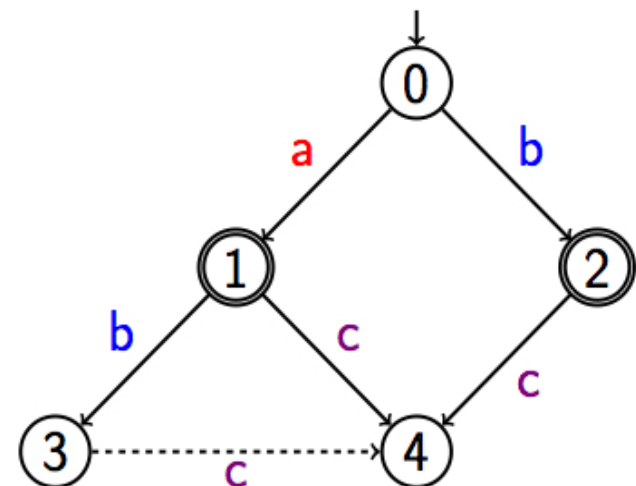
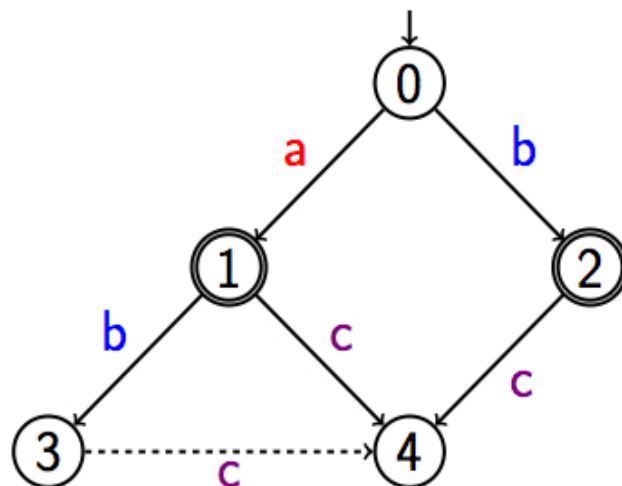
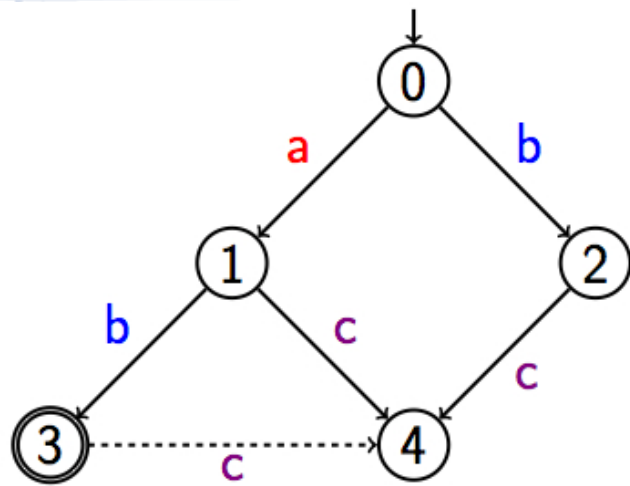






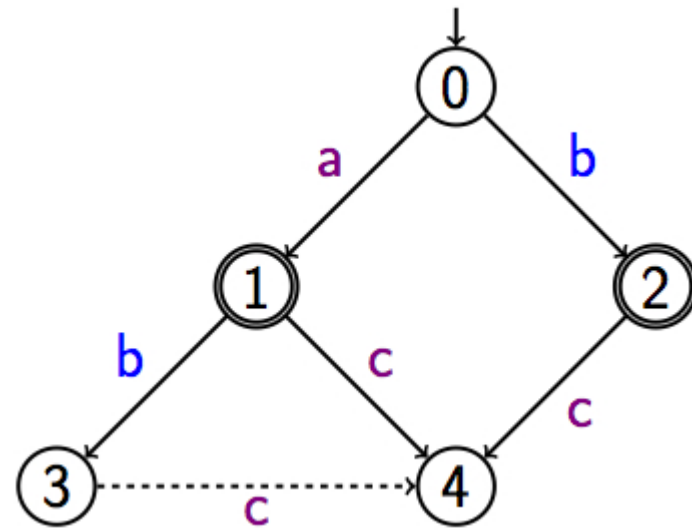
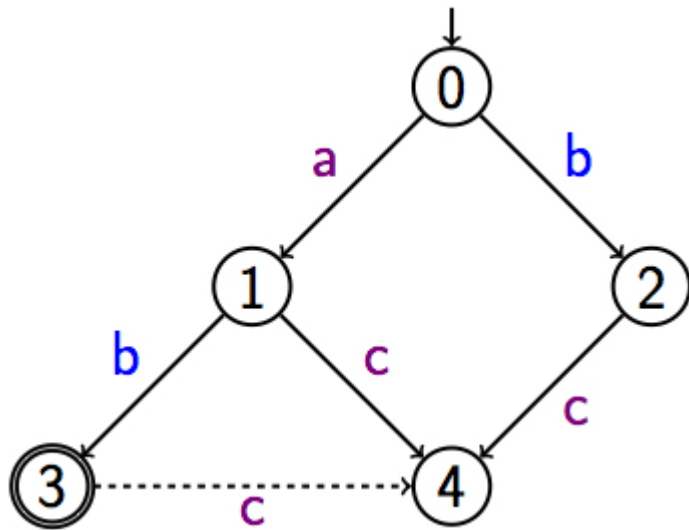


In total: 57 states, 23 transition labels, 193 transitions

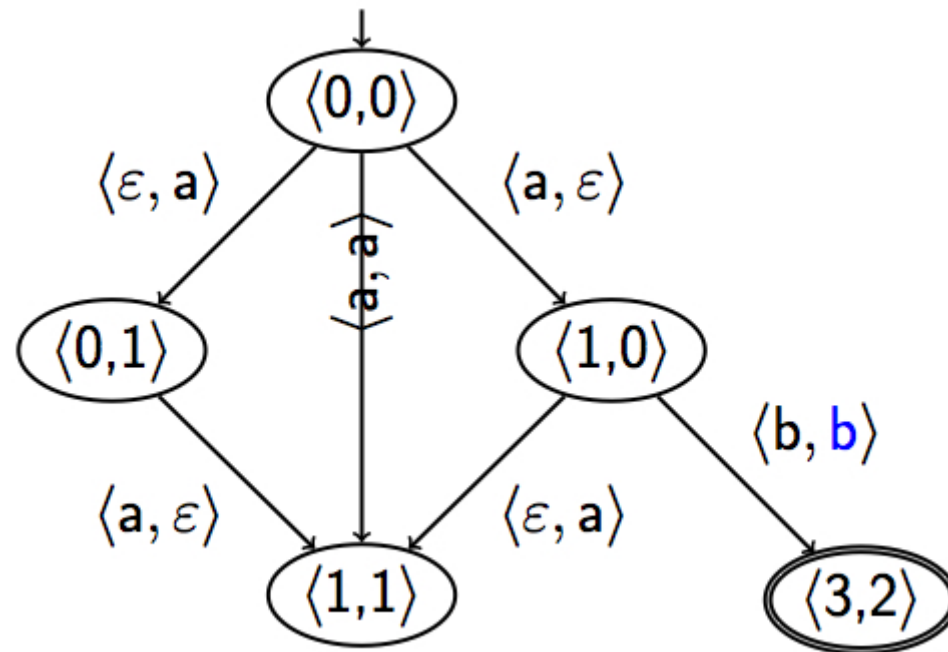
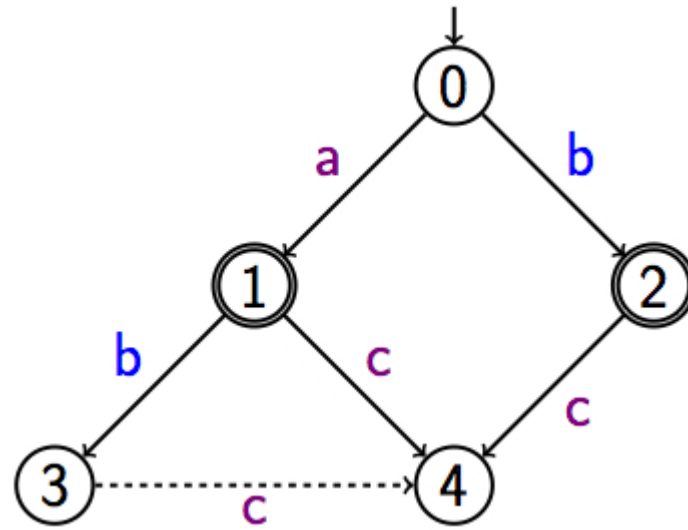
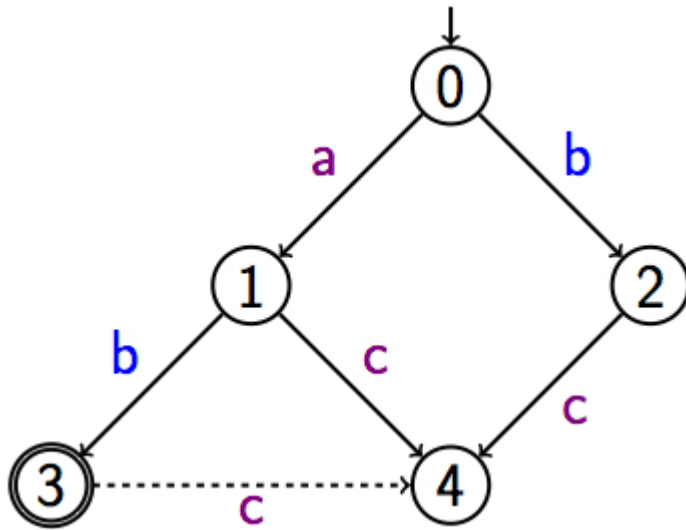


Choose a potential communication to avoid reaching an illegal configuration...

Checking Observability



Checking Observability

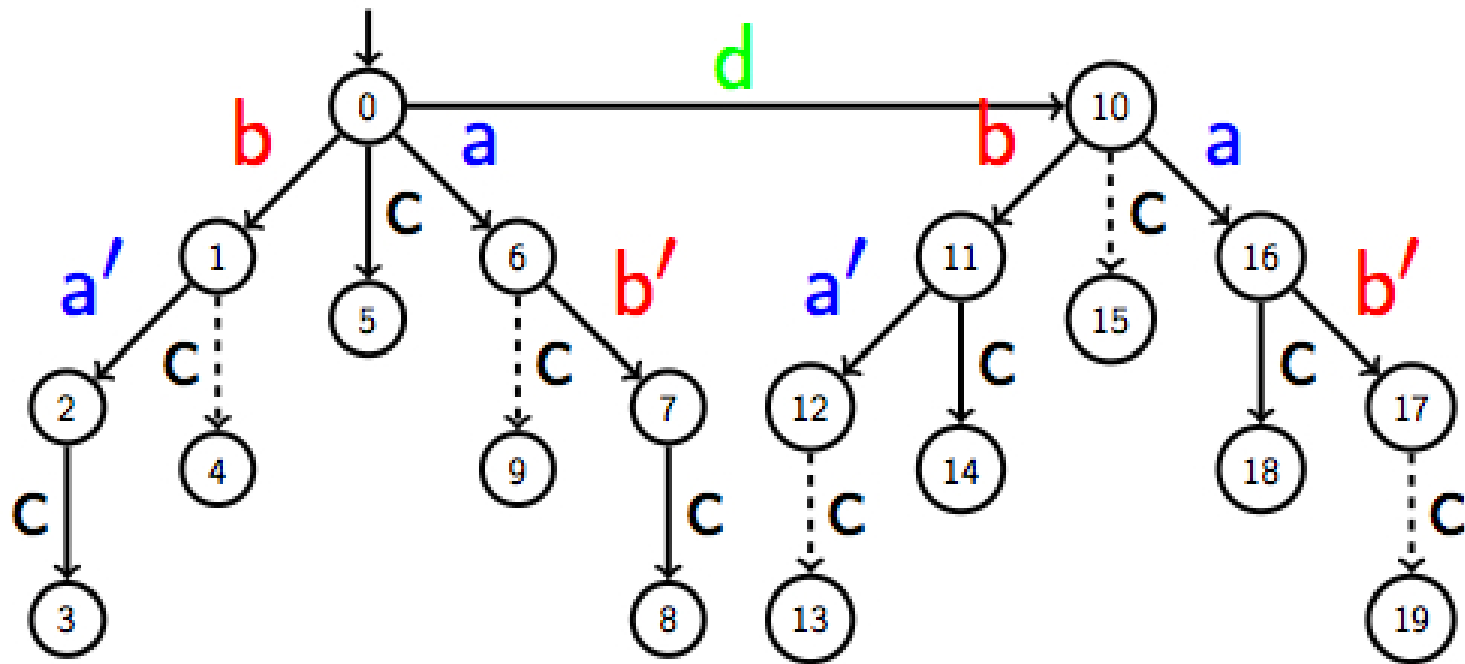


21 states
27 transitions
9 labels

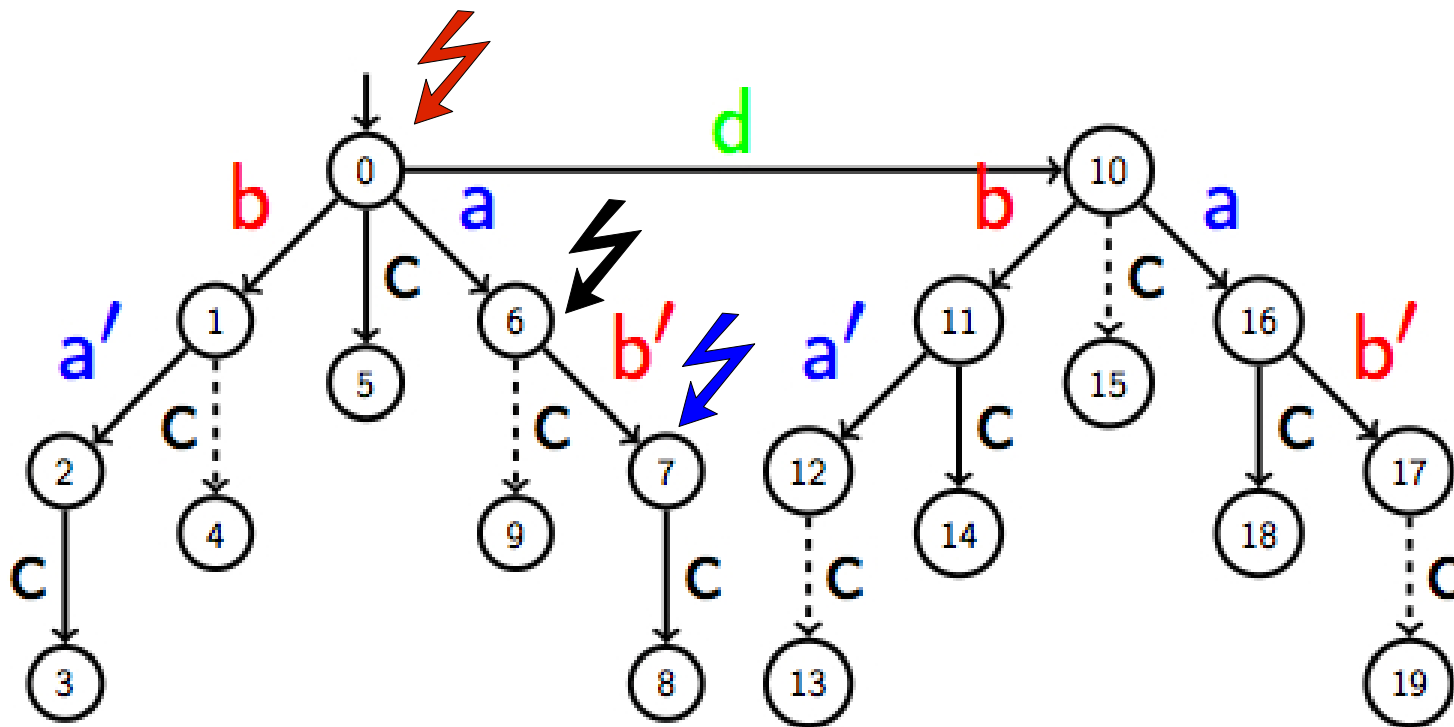
Checking Inference Observability

- Special class of decentralized languages that slipped through the cracks of the original definition of co-observability
- Allows an agent – in light of its own uncertainty – to reason about what another agent knows about a jointly-controlled event
- Yoo & Lafortune (2002), Ricker & Rudie (2007), Kumar & Takai (2007)

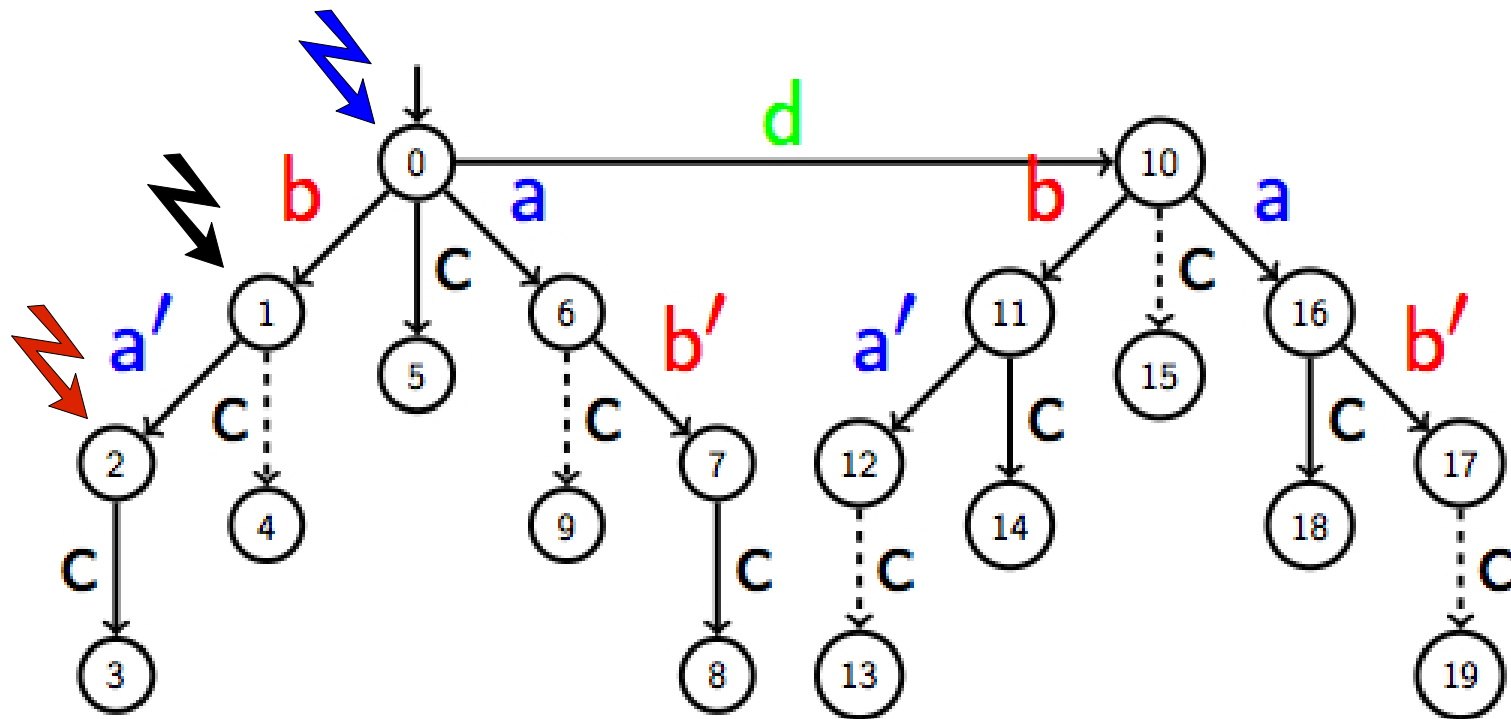
Checking Inference Observability



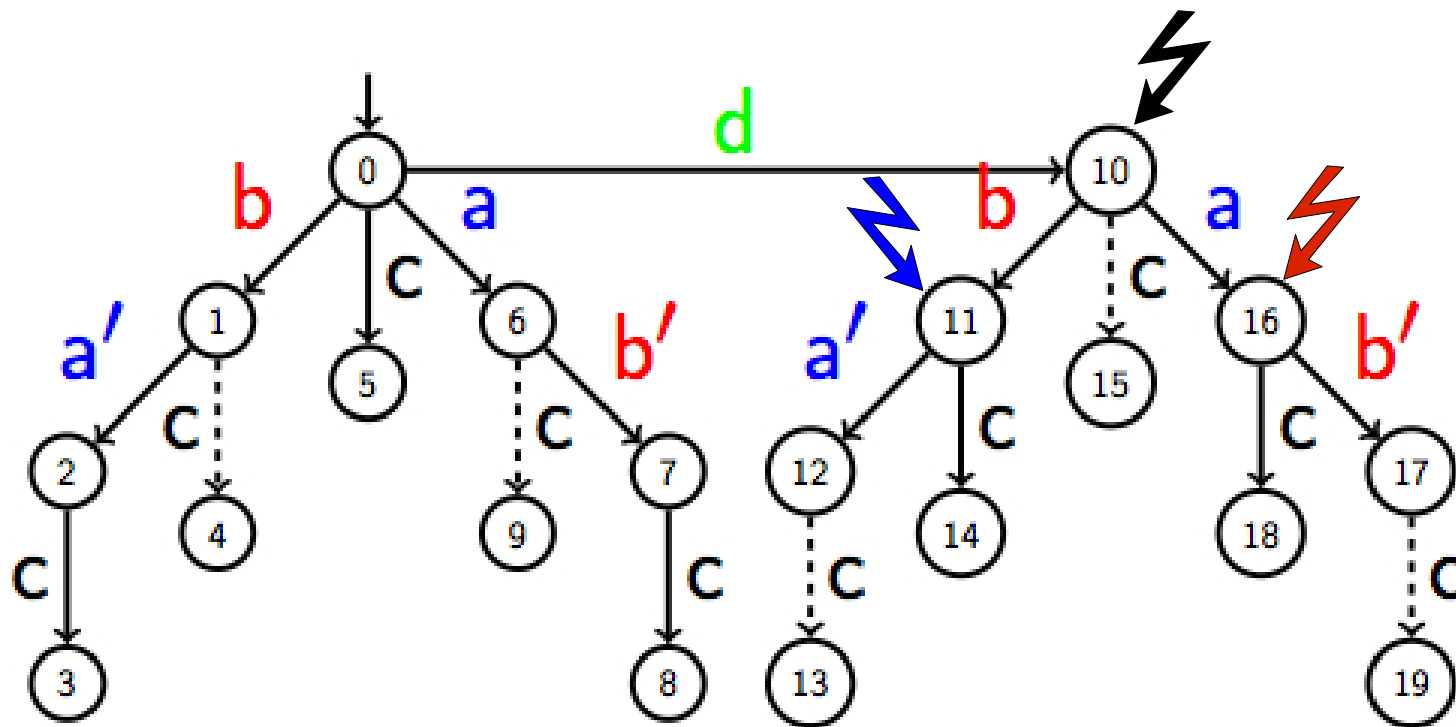
Checking Inference Observability



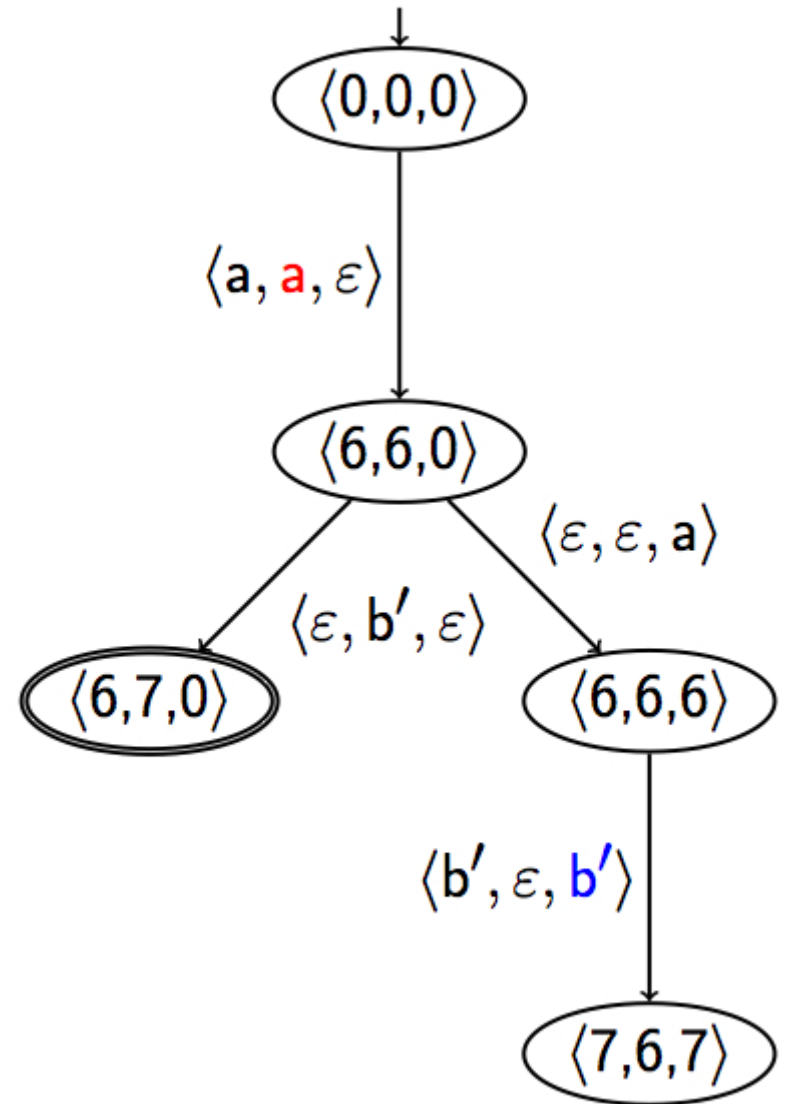
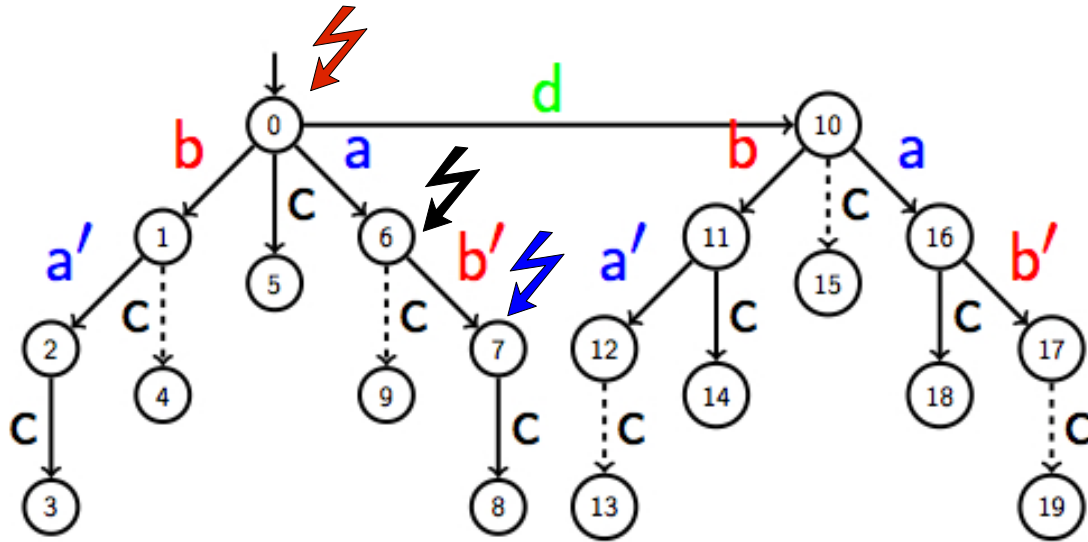
Checking Inference Observability



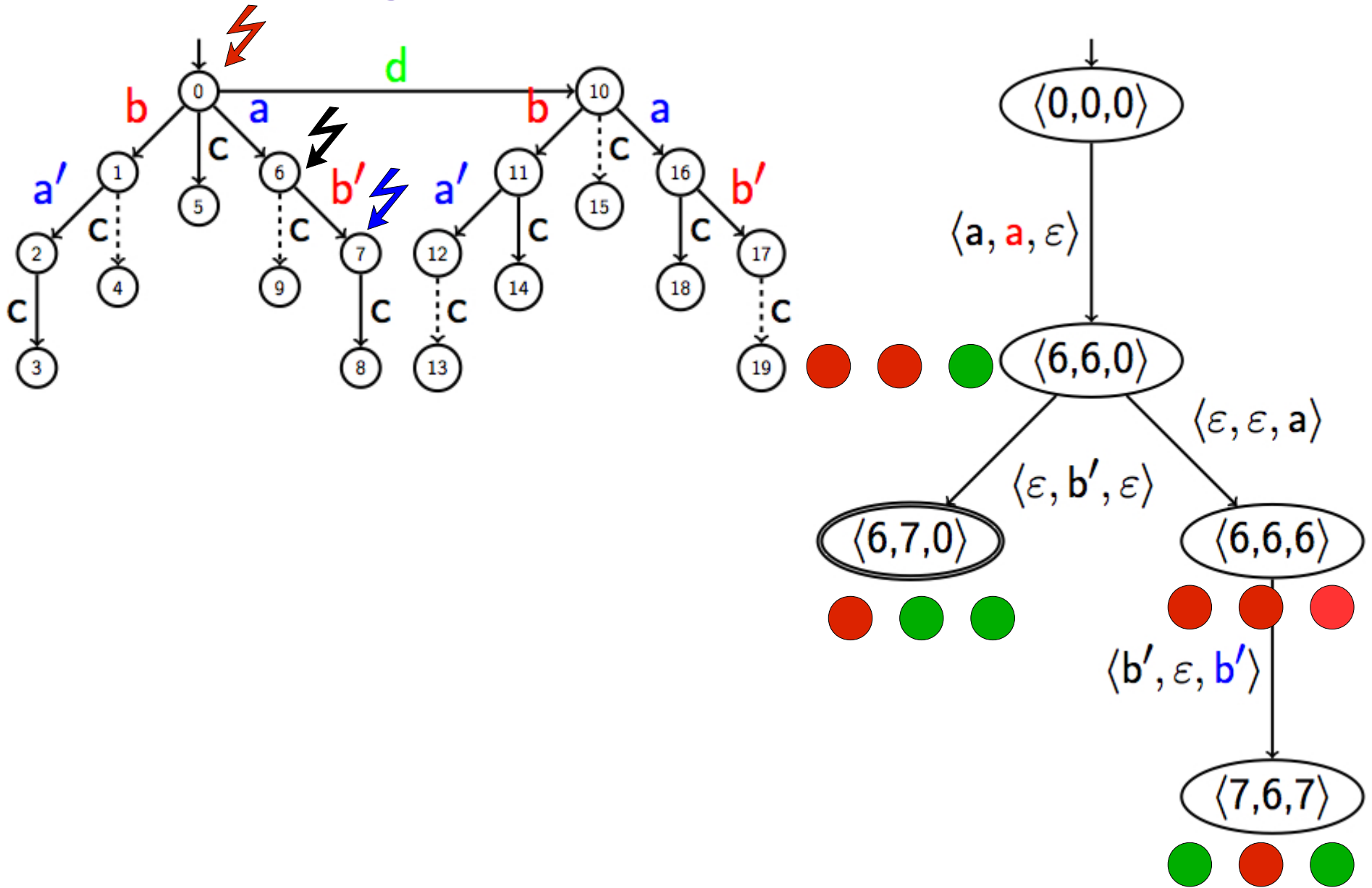
Checking Inference Observability



Checking Inference Observability

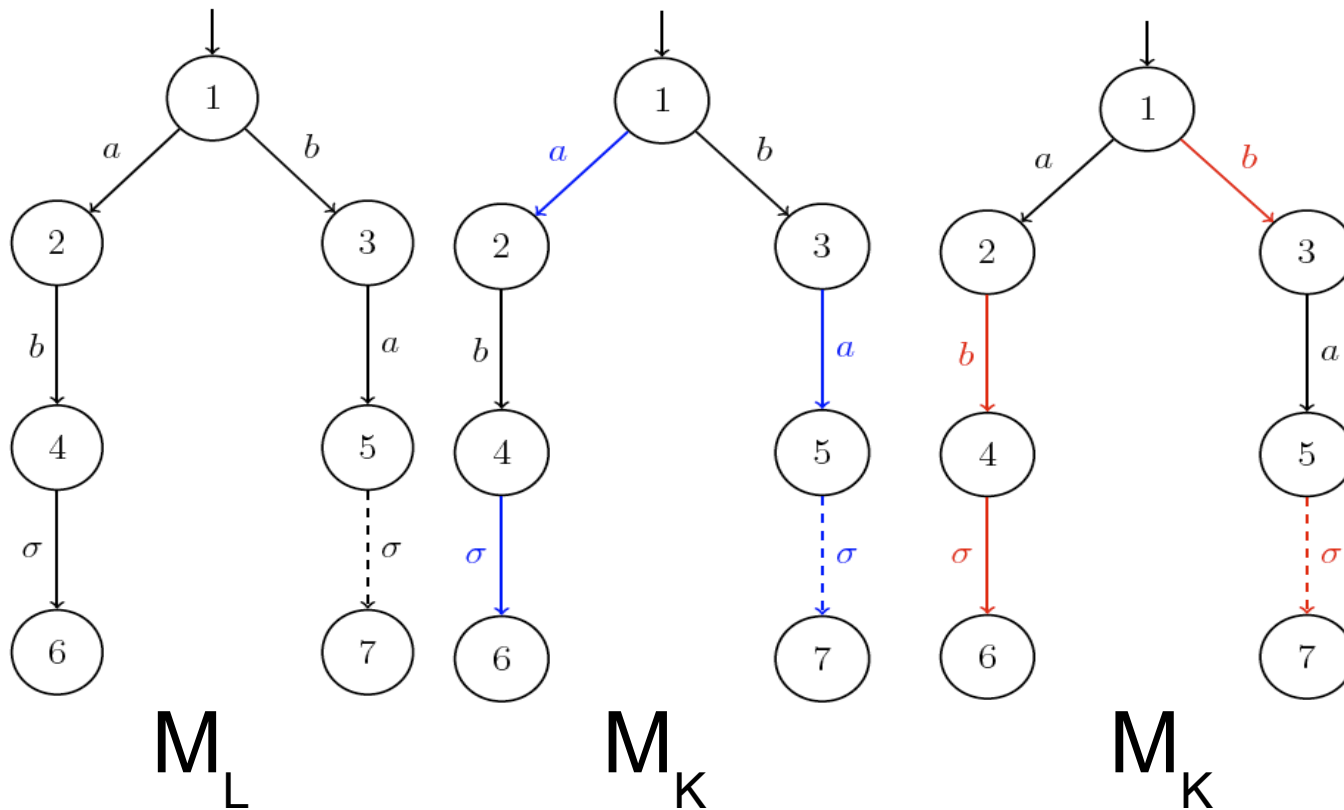


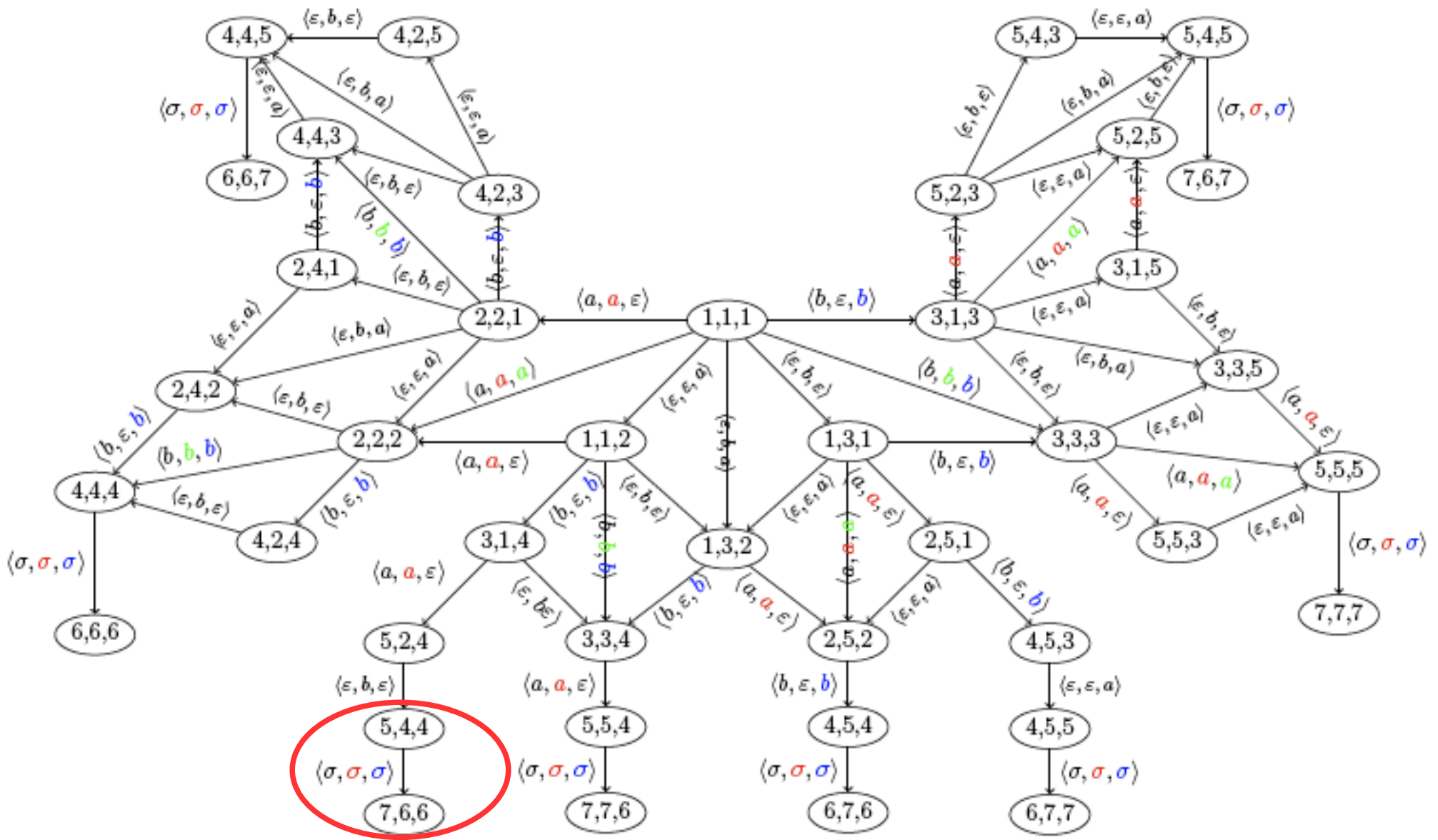
Checking Inference Observability

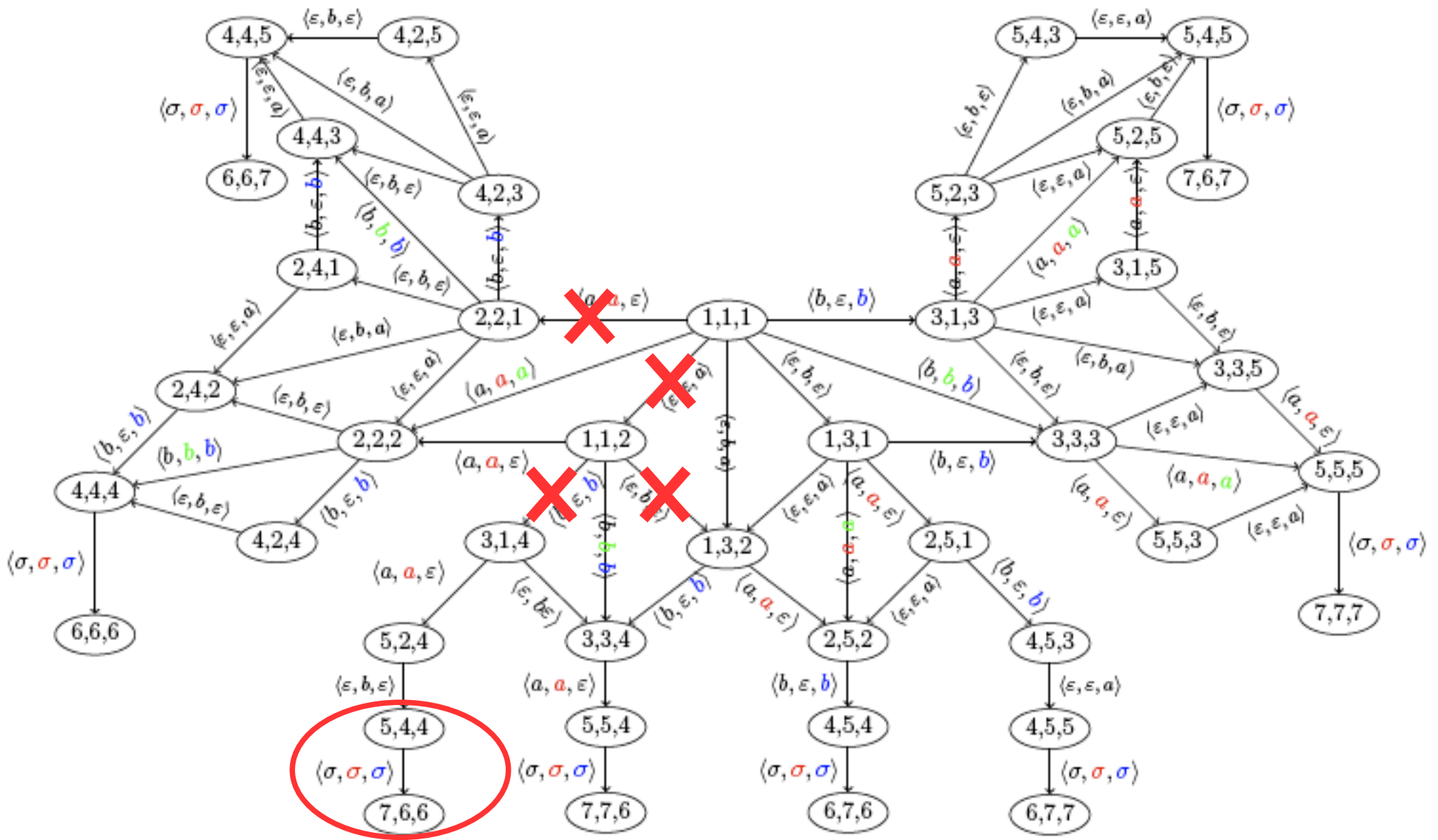


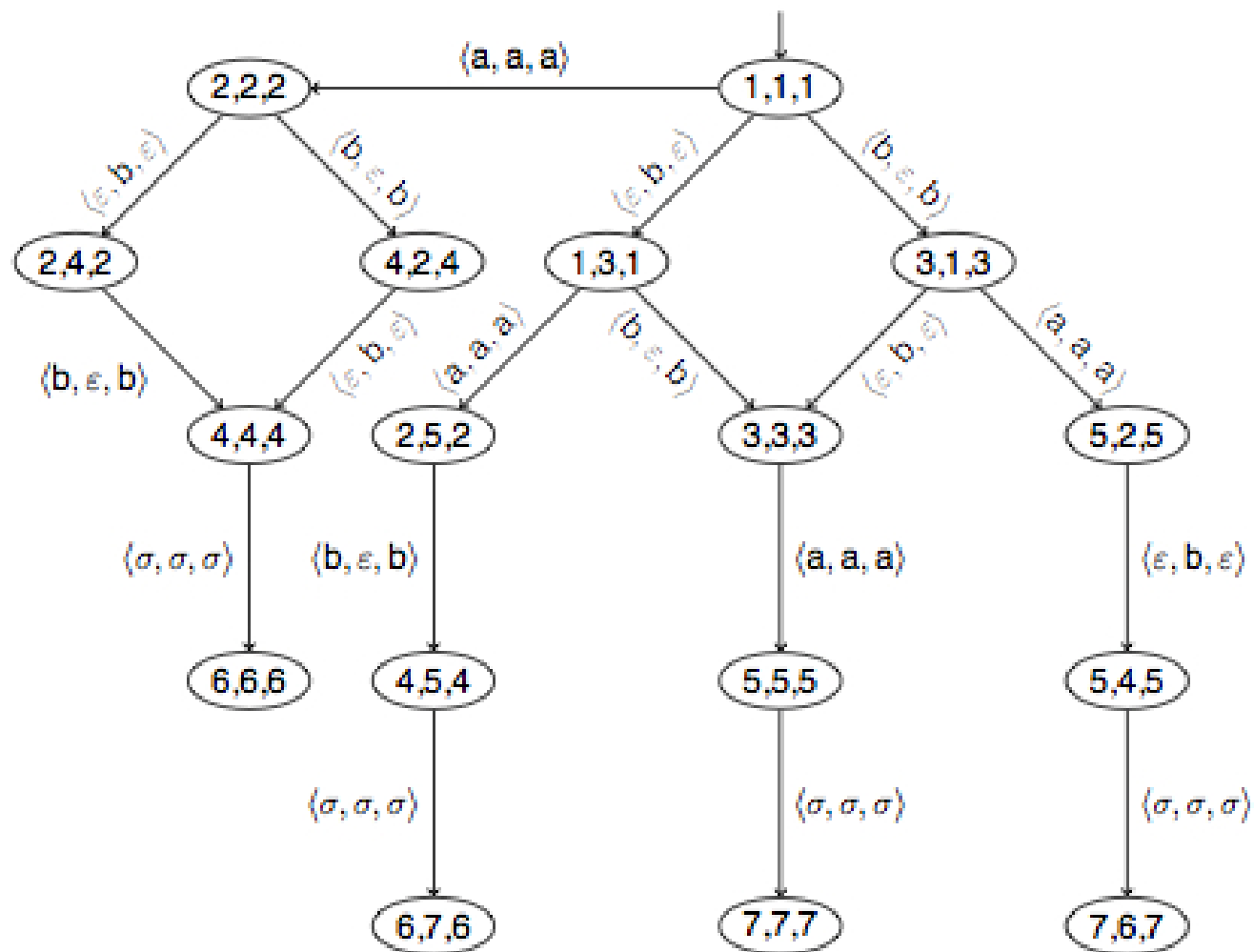
Finite automaton for designing communication protocols

State space: product of M_L and n copies of M_K



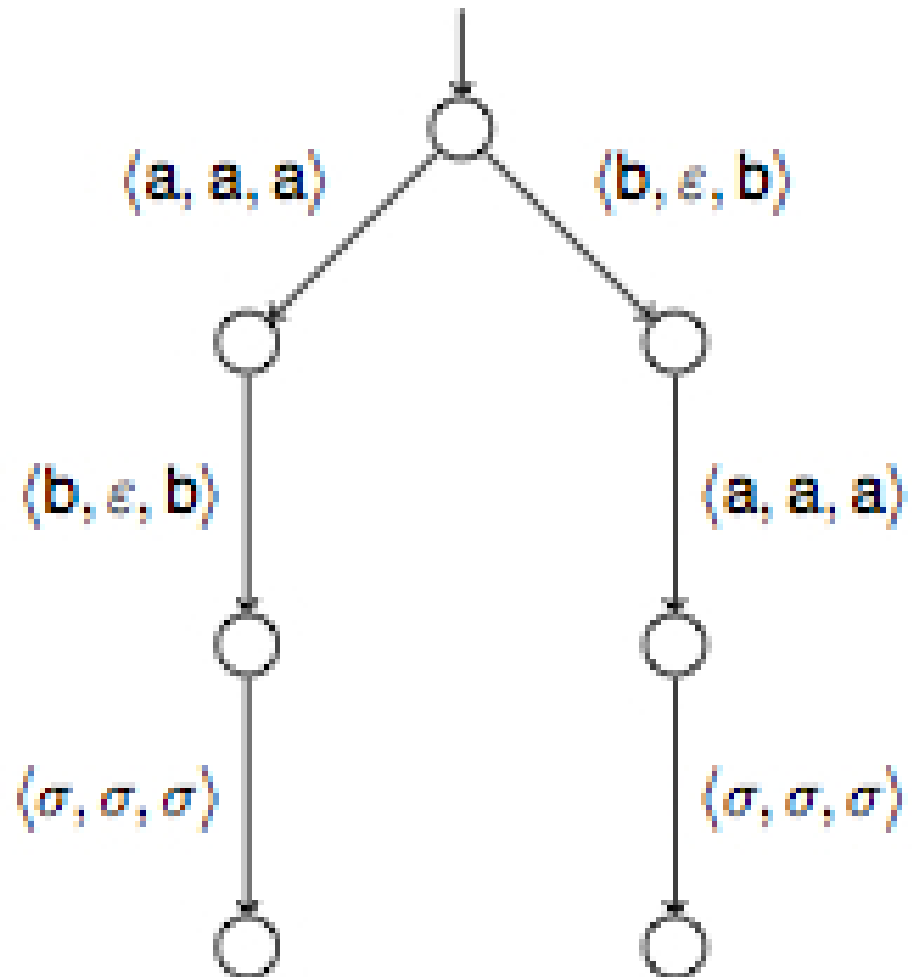
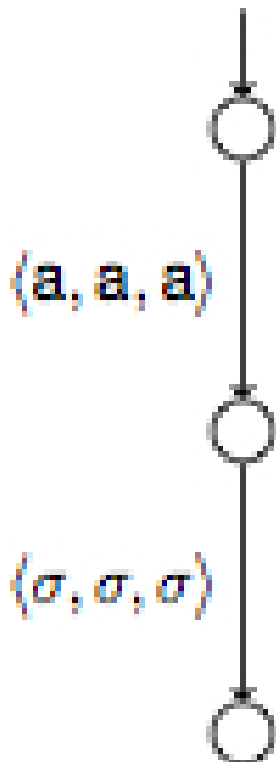






Synthesizing Controllers

Use notion of crushed automata from Morin, 1998



Other uses for U

- Anything we can do for control, we can do for diagnosis
 - Test for diagnosability
 - Test for co-diagnosability
 - Test for all flavours of conditional decisions for decentralized diagnosis
 - Introduce communication for decentralized diagnosis

Future use for U

- Bounded delay communication in decentralized control
 - Assume that the plant has a global clock (represent passage of unit of time as the “standard” tick event)
 - A tick event is uncontrollable
 - Can then use tick events to determine maximum delay allowable (or known delay) for communication between controllers - before illegal configurations are reached

