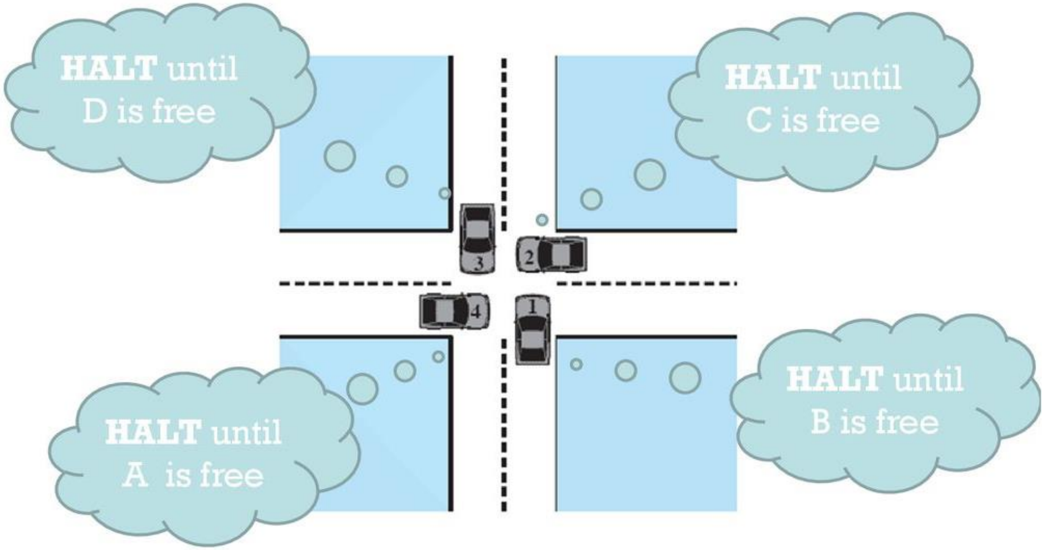
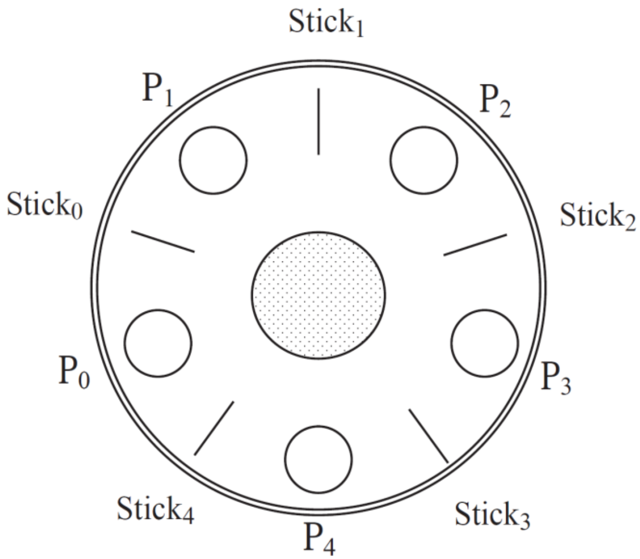
Formal Properties

Deadlock

- Sequential programs may have terminal states
- For parallel systems, however, computations typically do not terminate
- **Deadlock state:** $Post(x) = \emptyset$; a system with no deadlock is called **live**
- Therefore, deadlocks are undesirable and mostly represent a design error.
- A typical deadlock scenario occurs in the synchronization when components mutually wait for each other to progress.
- We **assume mostly the systems is live**; deadlock avoidance is another story.



Example: Dining Philosophers

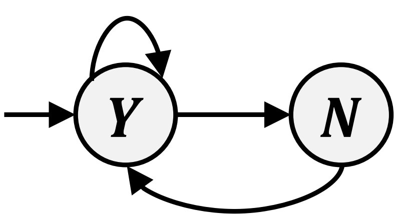


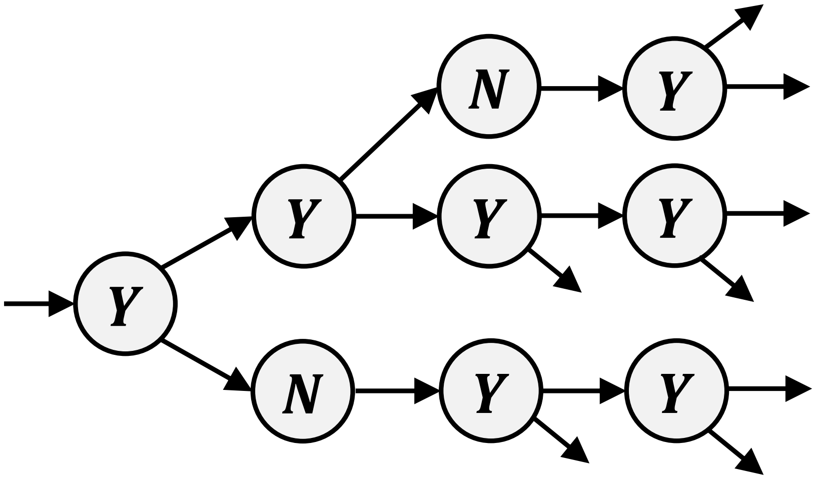
- **To take food, each philosopher needs two sticks**
- **A deadlock occurs when all philosophers possess a single stick**
- **The problem is to design a protocol such that the complete system is deadlock-free, i.e., at least one philosopher can eat and think infinitely often.**
- **A fair solution may be required with each philosopher being able to think and eat infinitely often (freedom of individual starvation)**

A possible solution is to make the sticks available for only one philosopher at a time. It can be verified that this solution is deadlock- and starvation-free.

Linear-Time Property

- Recall $Trace(T) \subseteq \left(2^{AP}\right)^{\omega}$ is the set of infinite sequence generated by T
- A linear-time property P over AP is a subset of $\left(2^{AP}\right)^{\omega}$ specifying the traces that a transition system should exhibit
- We say that system T satisfies P, denoted by $T \models P$, if $Trace(T) \subseteq P$
- “Linear” is the opposite of “branching” not “nonlinear”
- LT property is on a specific infinite execution





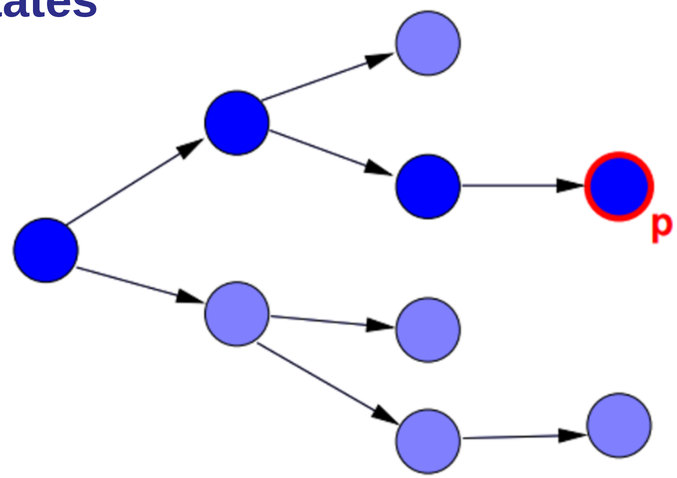
Safety & Invariant

- **Safety**: bad things never happen ⇔ always good things
- **Invariant**: some property should hold for all reachable state

An LT property P_{inv} is an **invariant** if there is a propositional logic formula Φ (called the invariant condition) over AP such that

$$P_{inv} = \{A_0A_1A_2 \cdots \in \left(2^{AP}\right)^{\omega} : \forall i \geq 0, A_i \vDash \Phi\}$$

- Therefore, $T \vDash P_{inv}$ iff $L(x) \vDash \Phi$ for all reachable states
- Can be checked easily be a DFS or a BFS
- Mutual exclusion property: $\Phi = \neg crit_1 \vee \neg crit_2$
- Traffic light: $\Phi = \neg green \vee \neg walk$



Safety

- Invariant is essentially a state-based safety property
- In general, safety may impose requirements on finite path fragments
- Ex: Money can only be withdrawn from the ATM once a correct PIN has been provided; this is not invariant but is still safety

An LT property P_{safe} is a **safety property** if for all $\sigma \in \left(2^{AP}\right)^{\omega} \setminus P_{safe}$ there exists **a finite bad prefix** $\hat{\sigma}$ of σ such that

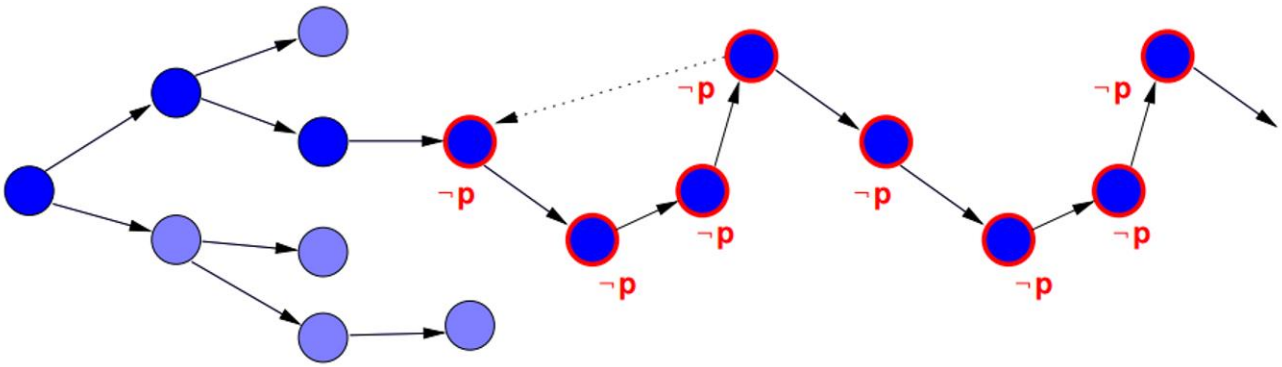
$$P_{safe} \cap \left\{\sigma' \in \left(2^{AP}\right)^{\omega} : \hat{\sigma} \text{ is a finite prefix of } \sigma'\right\} = \emptyset$$

- $AP = \{red, yellow\}$
- red phase must be preceded immediately by a yellow phase
- Bad prefix: $\{yellow\}\emptyset\emptyset\{red\}$, $\{yellow\}\{yellow\}\{red\}\{red\}$

Liveness

- **Safety says "something bad never happens"**
- **We also need liveness saying "something good will happen"**
- **Liveness should not constrain the finite behaviors, but require a certain condition on the infinite behaviors.**
- **For example, certain events occur infinitely often.**

> **An LT property P_{live} is a liveness property if $pref(P_{live}) = \left(2^{AP}\right)^*$**
> - $pref(P_{live})$ **denotes the set of all finite prefix of P_{live}**
> - $\left(2^{AP}\right)^*$ **denotes the set all of finite words over 2^{AP}**



Example: Liveness

- **Eventually**: each process will eventually enter its critical section:

  the set of all infinite words $A_0 A_1 \cdots \in \left(2^{AP}\right)^{\omega}$ such that

  $$(\exists i \geq 0: crit_1 \in A_i) \wedge (\exists i \geq 0: crit_2 \in A_i)$$

- **Repeated eventually**: each process will enter its critical section infinitely often

  the set of all infinite words $A_0 A_1 \cdots \in \left(2^{AP}\right)^{\omega}$ such that

  $$(\forall k \geq 0, \exists i \geq k: crit_1 \in A_i) \wedge (\forall k \geq, \exists i \geq k: crit_2 \in A_i)$$

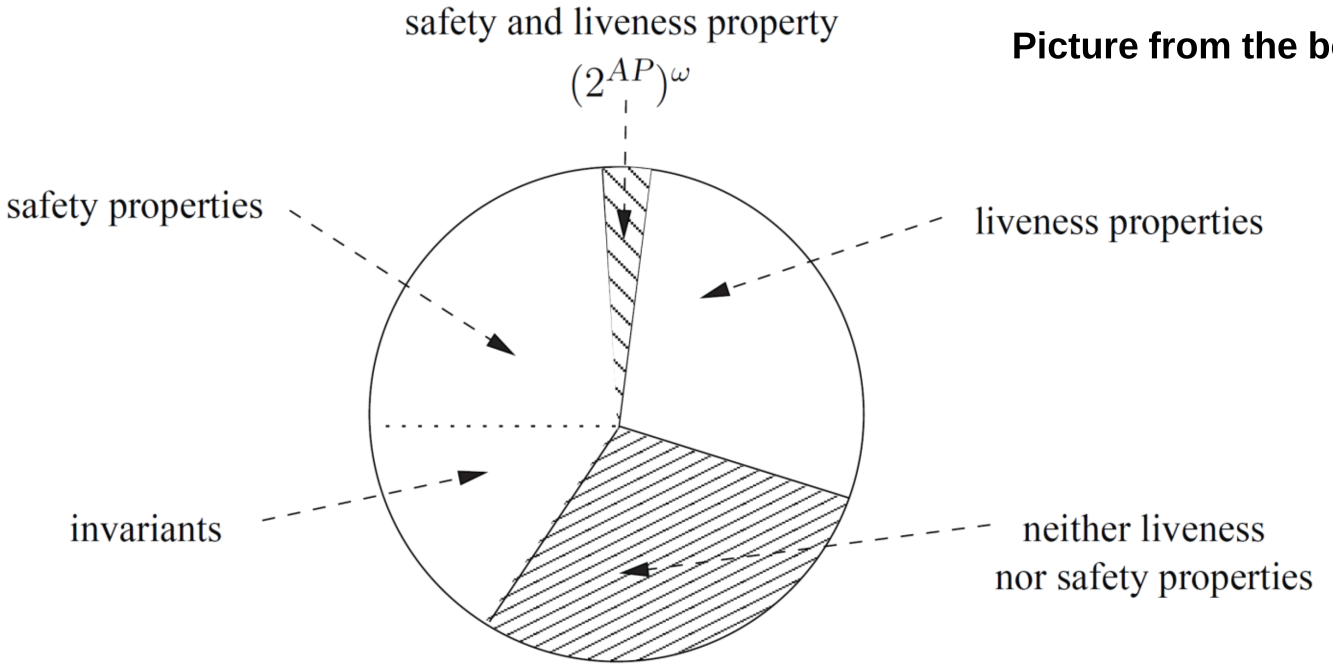
- **Starvation freedom**: each waiting process will eventually enter its critical section

  the set of all infinite words $A_0 A_1 \cdots \in \left(2^{AP}\right)^{\omega}$ such that

  $$\left(\forall i \geq: wait_1 \in A_i \Rightarrow (\exists k > i: crit_1 \in A_k)\right) \wedge$$
  $$\left(\forall i \geq: wait_2 \in A_i \Rightarrow (\exists k > i: crit_2 \in A_k)\right)$$

Decomposition Theorem

Any LT property P can be decomposed as $P = P_{safe} \cap P_{live}$



Picture from the book of Baier and Katoen

- $P = (2^{AP})^\omega$ **is the only property that is both safe and live**
- **In general, a property can be neither safe nor live**
  - **Consider** $AP = \{a\}$ **and** $P =$ **first** $\emptyset$ **and then** $\{a\}$ **infinitely often**
  - **It can be decomposed as** $P = \emptyset(2^{AP})^\omega \cap \{\sigma : \{a\} \text{ infinitely often in } \sigma\}$

Stage Summary

- System having no deadlock will generate infinite sequences
- Linear-time properties evaluate infinite sequences
- Safety is a property that is violated in a finite horizon
- Liveness is a property that does not care about what have done
- In general, an LT property consists of both safety and liveness

Question

(a) If a becomes valid, afterward b stays valid ad infinitum or until c holds.

(b) Between two neighboring occurrences of a, b always holds.

(c) Between two neighboring occurrences of a, b occurs more often than c.

(d) $a \wedge \neg b$ and $b \wedge \neg a$ are valid in alternation or until c becomes valid.

Question: For each property, determine if it is a safety or liveness or both or none.

Review of Last Lecture

- A dynamic system can be modeled as an LTS $T = (X, U, \rightarrow, X_0, AP, L)$
- A system can generate infinite sequences with properties $Trace(T)$
- A (linear-time) property is a set of “good” infinite traces $P \subseteq \left(2^{AP}\right)^{\omega}$
- $T \vDash P$ if $Trace(T) \subseteq P$ (nothing to do with actions)
- Some property can be violated in a finite horizon (safety)
- In general, a property can be decomposed as safety and liveness
- Large systems are obtained by composition $T = T_1 \otimes T_2 \otimes \cdots \otimes T_n$
- Product composition is essentially synchronization
- A general form of synchronization can be written as $T = T_1 \otimes_H T_2$, where $H \subseteq U_1 \times U_2$ are pairs that should be synchronized

Bisimulation & Abstraction

Model Equivalence by Bisimulation

Motivations

- Different people may build different models for the same system
- Some models are complex but some are simple
- How to determine whether two models are describing the same thing?
- How to simplify a complex model to a simple but equivalent one?

Basic Ideas

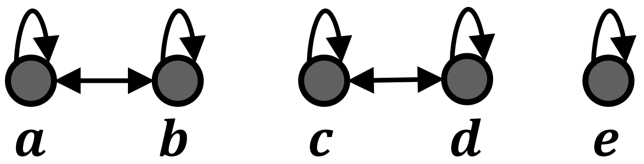
- Model equivalence is captured by "bisimulation"
- One model is more "precise" than the other if

  "no matter what you do, I can do the same thing" (simulation)
- Two models are equivalent if they can simulate each other

Equivalence Relation

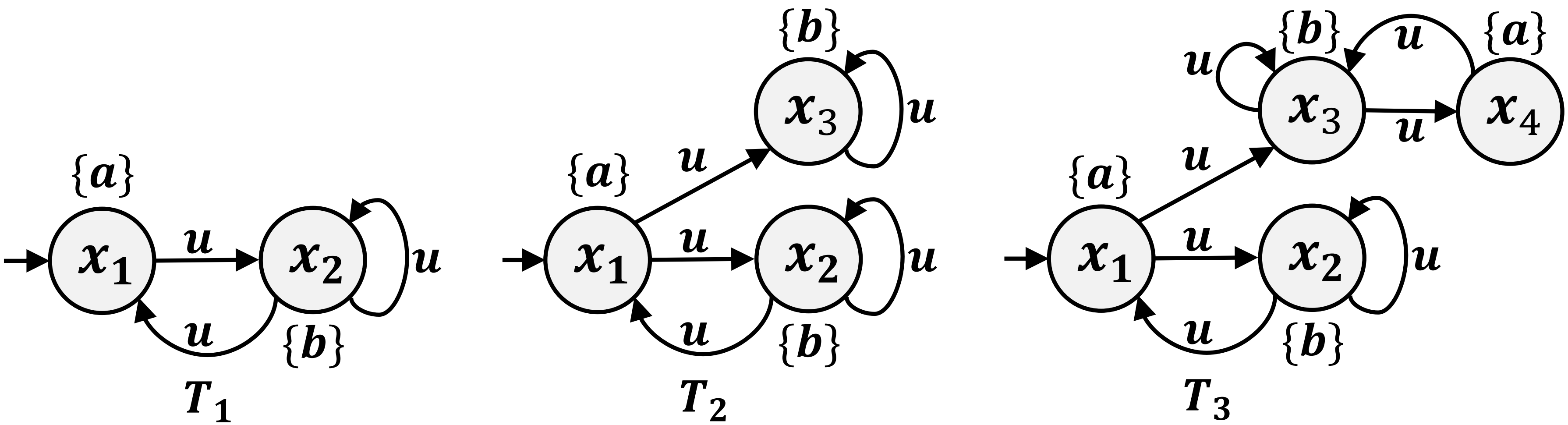
- a **relation** from set A to set B is a set of pairs $\sim \subseteq A \times B$
- we write $a \sim b$ if $(a, b) \in \sim$
- a relation $\sim \subseteq A \times A$ on A is an **equivalence relation** if it satisfies:
  - reflexivity: $\forall a \in A: a \sim a$
  - symmetry: $\forall a, b \in A$, if $a \sim b$, then $b \sim a$
  - transitivity: $\forall a, b, \in A$, if $a \sim b$ and $b \sim c$, then $a \sim c$
- an equivalent relation induces an **equivalent class**

$$A/_{\sim} = \left\{[a] \in 2^A : a \in A\right\}, \text{ where } [a] = \{b \in A : a \sim b\}$$



- $A = \{a, b, c, d, e\}$
- $\sim = \{(a, a), (a, b), (b, a), (b, b), (c, c), (c, d), (d, c), (d, d), (e, e)\}$
- $A/_{\sim} = \left\{\{a, b\}, \{c, d\}, \{e\}\right\}$
- $[a] = [b] = \{a, b\}, [c] = [d] = \{c, d\}, [e] = \{e\}$

Model Equivalence



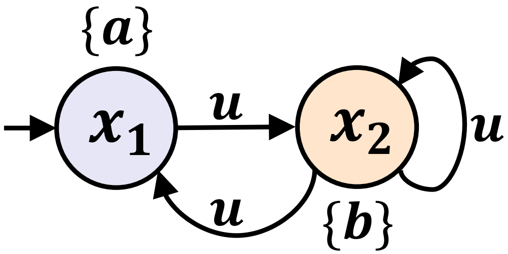
- $Trace(T_1) = Trace(T_2)$ but state x_3 is T_2 seems to be different
- $Trace(T_1) = Trace(T_3)$ but are they really equivalent?

Observations

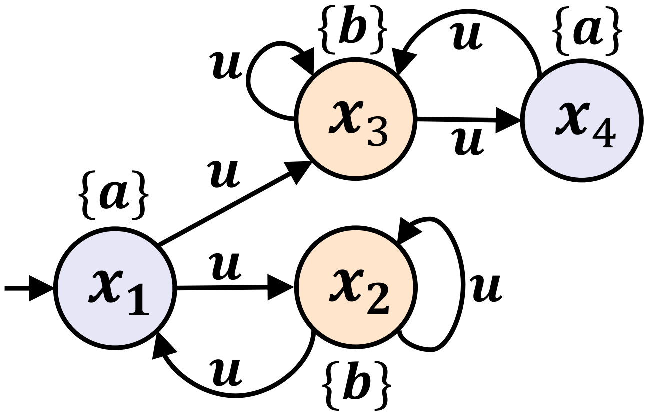
- trace equivalence is not good enough to describe model equivalence although it is good enough for LT properties
- we needs to look at the equivalence of states

State Equivalence

- What does two states are "equivalent" mean?
  - they should have the same property (atomic propositions)
  - they should have the same future behaviors
- Two systems are equivalent if their initial states are equivalent
- For a system itself, we can aggregate equivalent states (abstraction)



T_1



T_3

Simulation Relation

Let T_1 and T_2 be two LTSs, where $T_i = (X_i, U_i, \rightarrow_i, X_{0,i}, AP, L_i)$. Then a relation $\sim \subseteq X_1 \times X_2$ is a simulation relation from T_1 to T_2 if

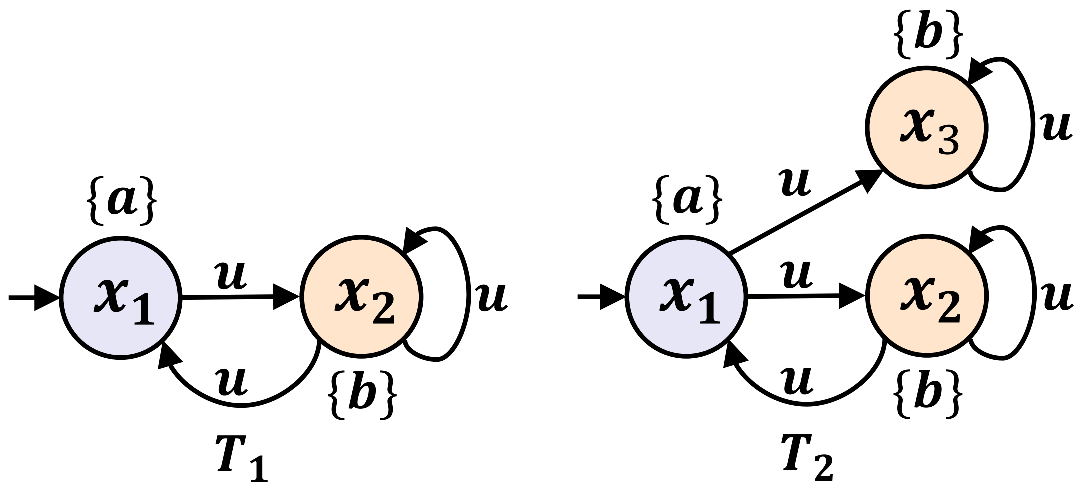
- $\forall x_{0,1} \in X_{0,1}, \exists x_{0,2} \in X_{0,2}: x_{0,1} \sim x_{0,2}$
- **for all $x_1 \sim x_2$, it holds that**
  - $L_1(x_1) = L_2(x_2)$
  - **If $x_1' \in Post(x_1)$ then there exists $x_2' \in Post(x_2)$ with $x_1' \sim x_2'$**

We say T_1 is simulated by T_2 or T_2 simulates T_1, denoted by $T_1 \preccurlyeq T_2$ if there exists a simulation relation from T_1 to T_2

- **$x_1^0 \sim x_2^0$ implies**

$$\forall x_1^0 \xrightarrow[1]{u_1} x_1^1 \xrightarrow[1]{u_2} \cdots \xrightarrow[1]{u_n} x_1^n, \exists x_2^0 \xrightarrow[2]{u_1'} x_2^1 \xrightarrow[2]{u_2'} \cdots \xrightarrow[2]{u_n'} x_2^n: L_1\left(x_1^0 \cdots x_1^n\right) = L_2(x_2^0 \cdots x_2^n)$$

Example: Simulation Relation



- **We have** $T_1 \preccurlyeq T_2$
- **Consider relation** $\sim = \{(x_1, x_1), (x_2, x_2)\} \subseteq X_1 \times X_2$

- $\forall x_{0,1} \in X_{0,1}, \exists x_{0,2} \in X_{0,2}: x_{0,1} \sim x_{0,2}$
- **for all** $x_1 \sim x_2$**, it holds that**
  - $L_1(x_1) = L_2(x_2)$
  - **If** $x_1' \in Post(x_1)$ **then there exists** $x_2' \in Post(x_2)$ **with** $x_1' \sim x_2'$

Bisimulation Relation

Let T_1 and T_2 be two LTSs, where $T_i = (X_i, U_i, \delta_i, X_{0,i}, AP, L_i)$. Then A relation $\sim \subseteq X_1 \times X_2$ is a **bisimulation relation** between T_1 to T_2 if

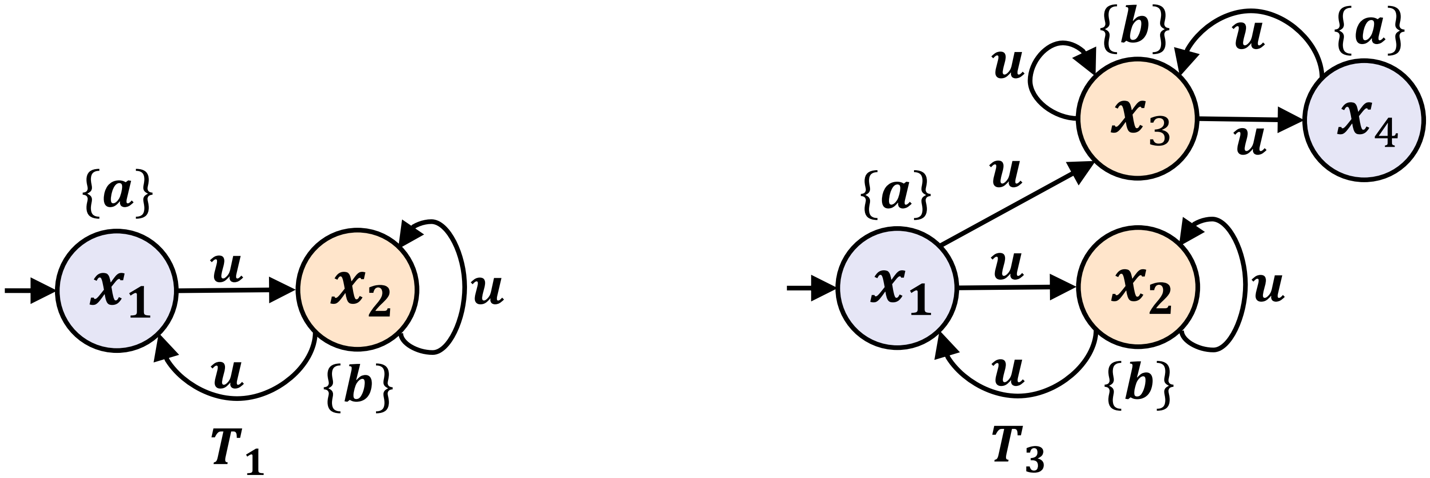
- $\forall x_{0,1} \in X_{0,1}, \exists x_{0,2} \in X_{0,2}: x_{0,1} \sim x_{0,2}$
- $\forall x_{0,2} \in X_{0,2}, \exists x_{0,1} \in X_{0,1}: x_{0,1} \sim x_{0,2}$
- for all $x_1 \sim x_2$, it holds that
  - $L_1(x_1) = L_2(x_2)$
  - if $x_1' \in Post(x_1)$ then there exists $x_2' \in Post(x_2)$ with $x_1' \sim x_2'$
  - if $x_2' \in Post(x_2)$ then there exists $x_1' \in Post(x_1)$ with $x_1' \sim x_2'$

We say **T_1 and T_2 are bisimilar**, denoted by $T_1 \cong T_2$, if there exists a bisimulation relation between T_1 and T_2

Remark: bisimulation is equivalent to

- $\sim \subseteq X_1 \times X_2$ is a simulation relation from T_1 to T_2; and
- $\sim^{-1} \subseteq X_2 \times X_1$ is a simulation relation from T_2 to T_1.

Example: Bisimulation Relation

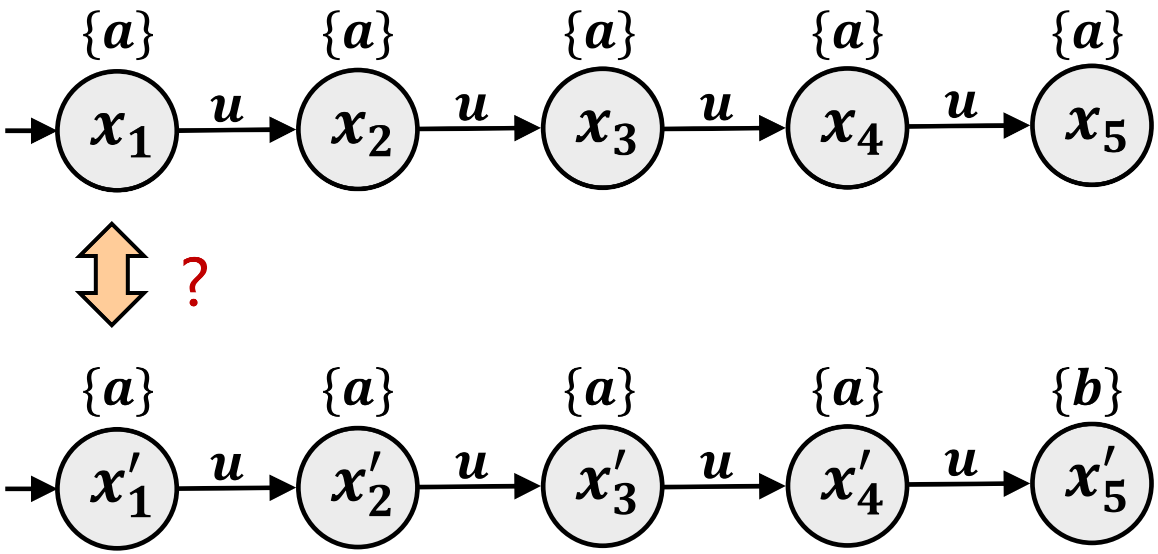


- We have $T_1 \cong T_3$
- Consider relation $\sim = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3)\} \subseteq X_1 \times X_3$

- $\forall x_{0,1} \in X_{0,1}, \exists x_{0,2} \in X_{0,2}: x_{0,1} \sim x_{0,2}$
- $\forall x_{0,2} \in X_{0,2}, \exists x_{0,1} \in X_{0,1}: x_{0,1} \sim x_{0,2}$
- for all $x_1 \sim x_2$, it holds that
  - $L_1(x_1) = L_2(x_2)$
  - if $x_1' \in Post(x_1)$ then there exists $x_2' \in Post(x_2)$ with $x_1' \sim x_2'$
  - if $x_2' \in Post(x_2)$ then there exists $x_1' \in Post(x_1)$ with $x_1' \sim x_2'$

Algorithm for Computing Bisimulation

- **Question: how to determine whether or not $T_1 \cong T_2$?**
- **Problem: bisimulation is a global property**



Fixed-Point Algorithm for Bisimulation

- Question: how to determine whether or not $T_1 \cong T_2$?
- Idea: first relate all pairs and then iterative shrink the relation

Define operator

$$F: 2^{X_1 \times X_2} \rightarrow 2^{X_1 \times X_2}$$

by: for any $R \subseteq X_1 \times X_2$, we have $(x_1, x_2) \in F(R)$ if

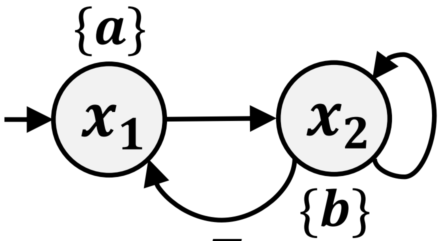
- $(x_1, x_2) \in R$
- $\forall x_1' \in Post(x_1), \exists x_2' \in Post(x_2): (x_1', x_2') \in R$
- $\forall x_2' \in Post(x_2), \exists x_1' \in Post(x_1): (x_1', x_2') \in R$

Then the fixed-point

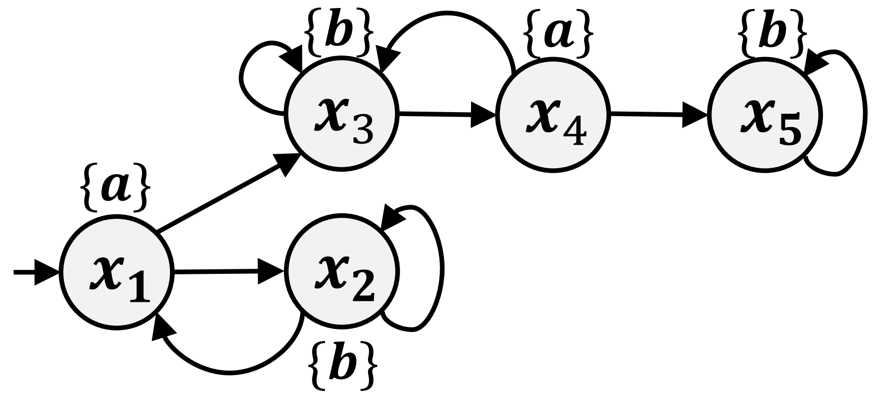
$$R^* := \lim_{k \to \infty} F^k(R_0), \text{ where } R_0 = \{(x_1, x_2): L_1(x_1) = L_2(x_2)\}$$

is the maximal bisimulation relation between T_1 and T_2.

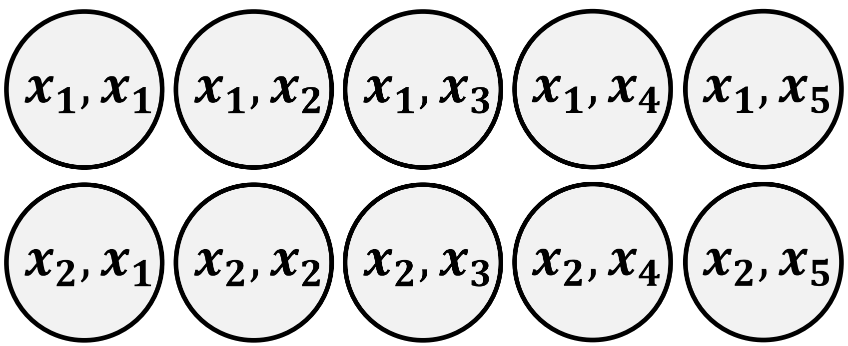
Example: Fixed-Point Iteration



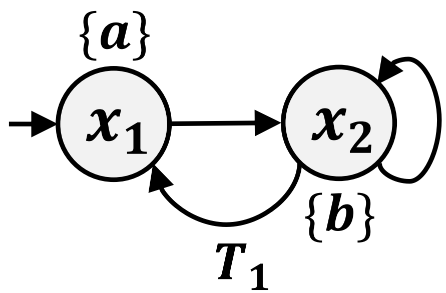
T_1

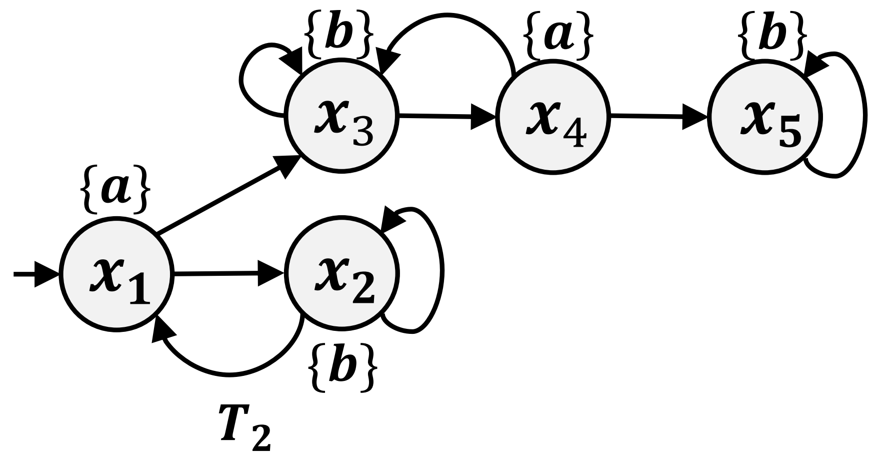


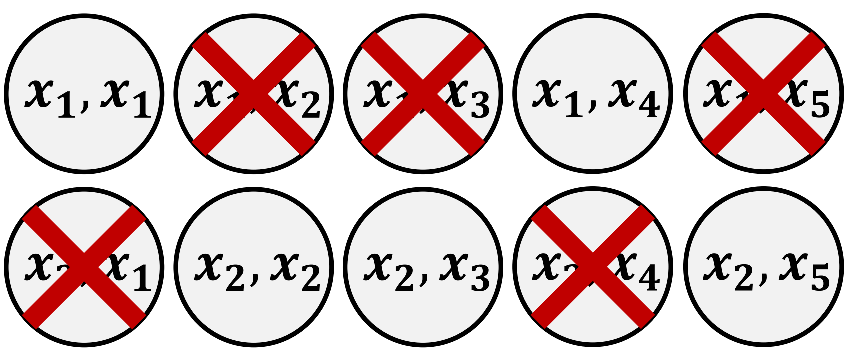
T_2



Example: Fixed-Point Iteration

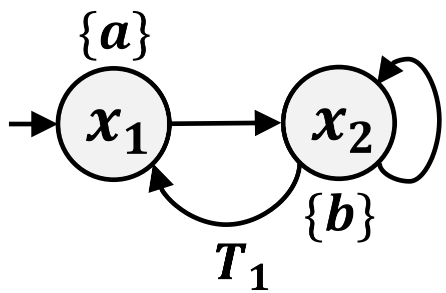


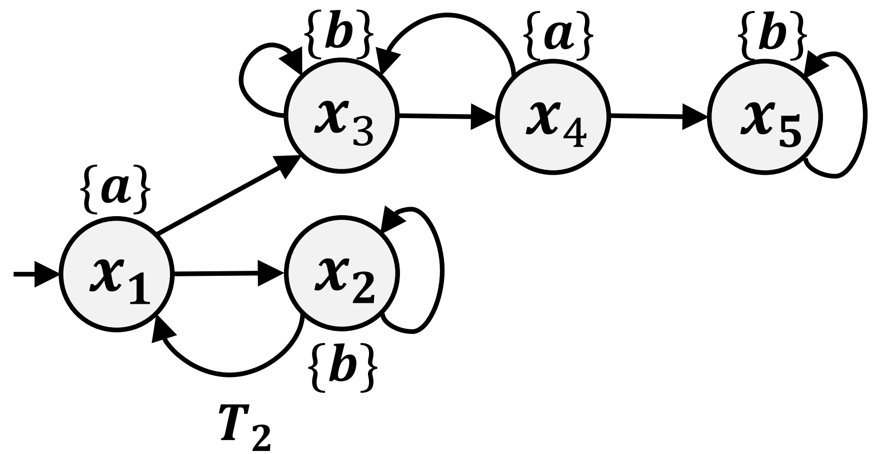


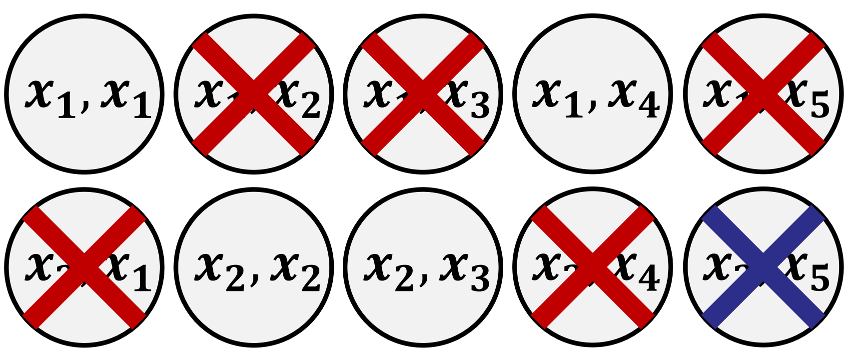


- $R_0 = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3), (x_2, x_5)\}$

Example: Fixed-Point Iteration

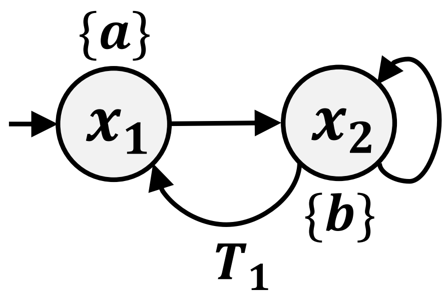


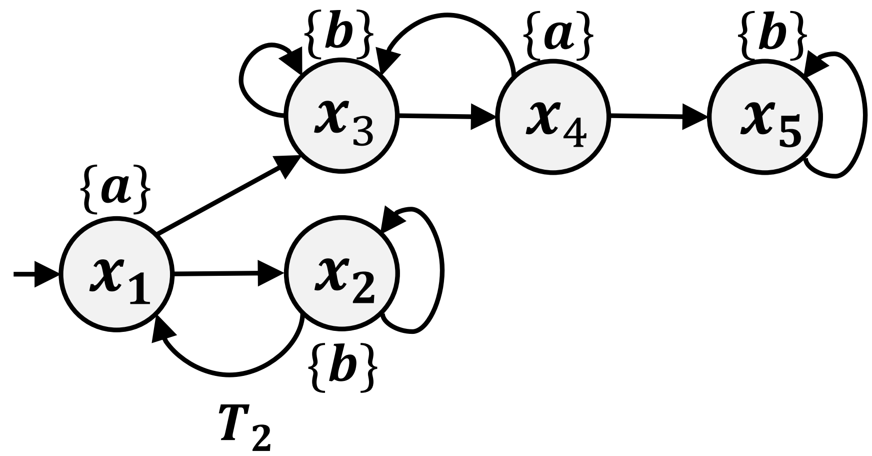


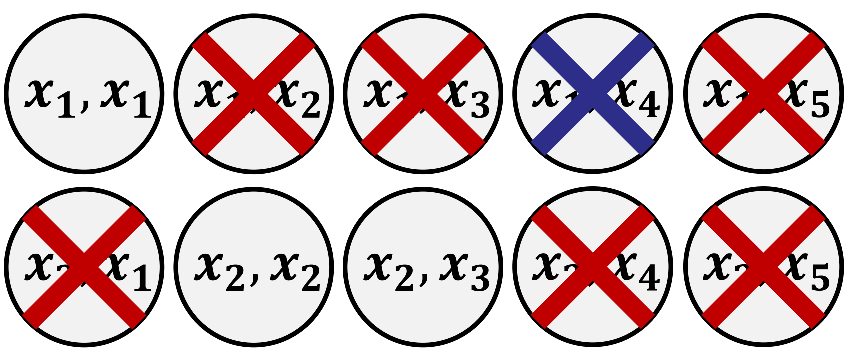


- $R_0 = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3), (x_2, x_5)\}$
- $R_1 = F(R_0) = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3)\}$

Example: Fixed-Point Iteration

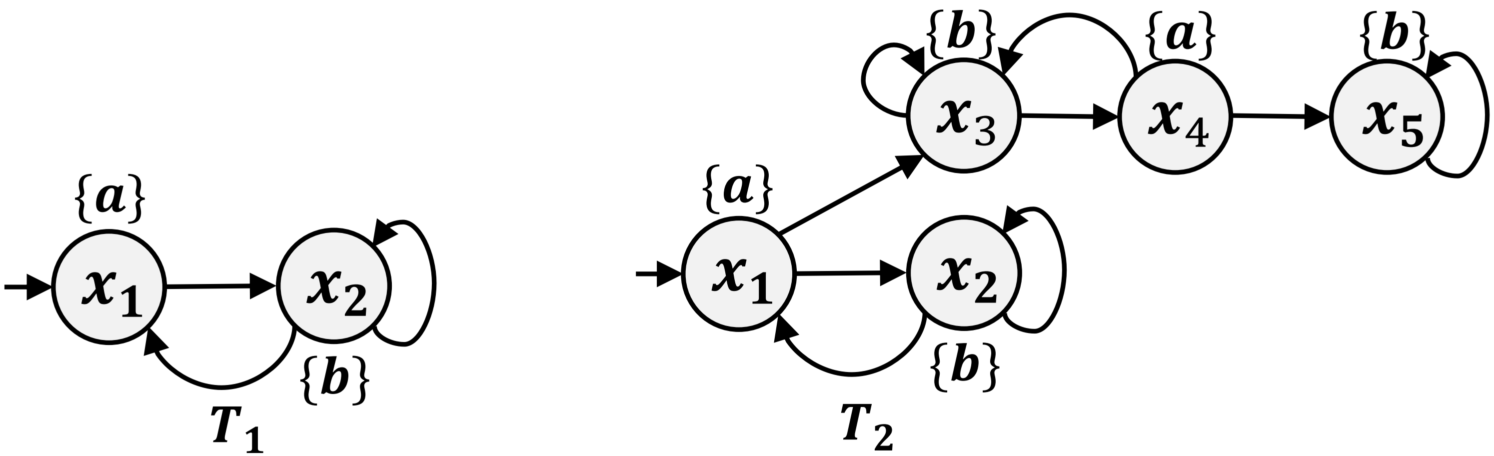


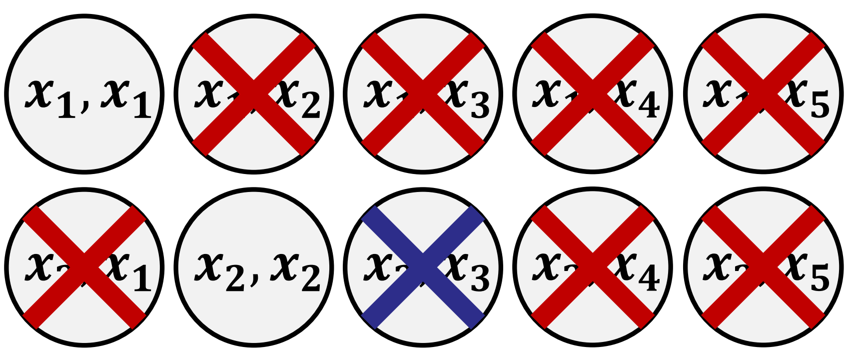




- $R_0 = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3), (x_2, x_5)\}$
- $R_1 = F(R_0) = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3)\}$
- $R_2 = F(R_1) = \{(x_1, x_1), (x_2, x_2), (x_2, x_3)\}$

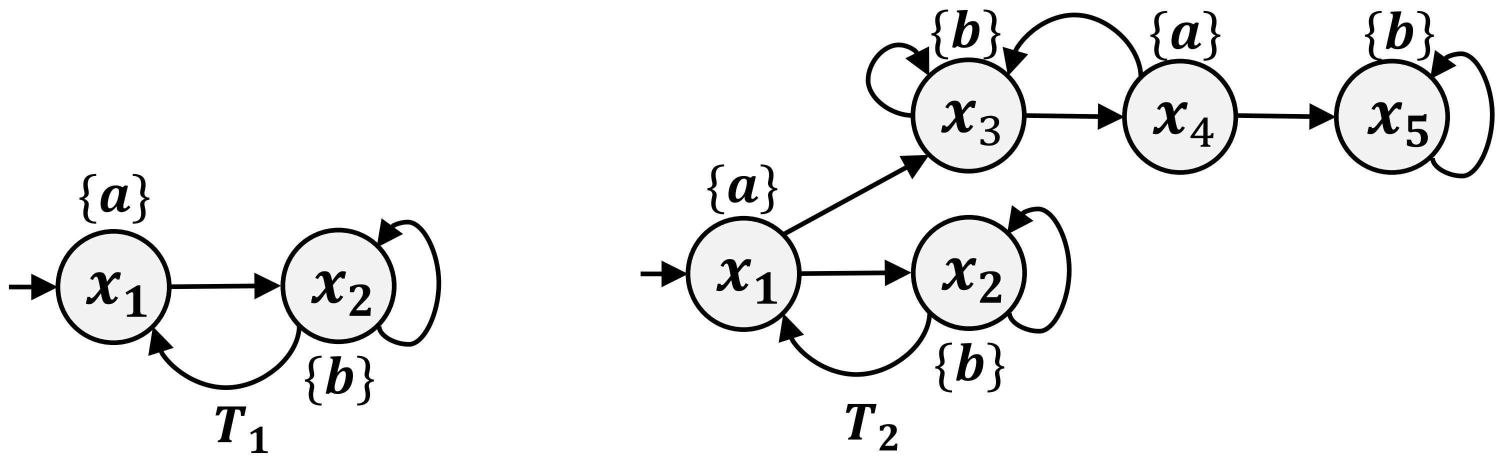
Example: Fixed-Point Iteration

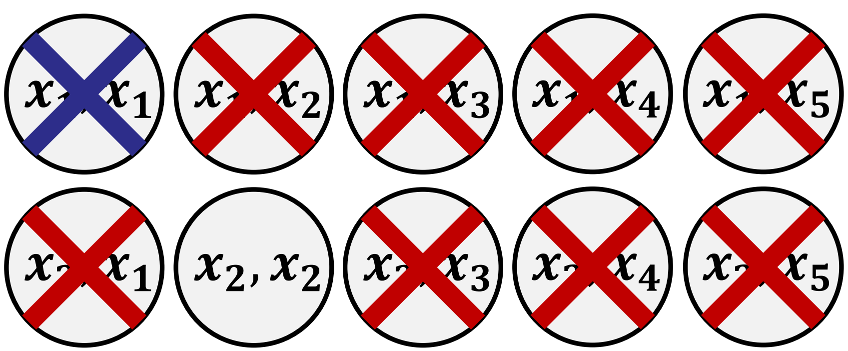




- $R_0 = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3), (x_2, x_5)\}$
- $R_1 = F(R_0) = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3)\}$
- $R_2 = F(R_1) = \{(x_1, x_1), (x_2, x_2), (x_2, x_3)\}$
- $R_3 = F(R_2) = \{(x_1, x_1), (x_2, x_2)\}$

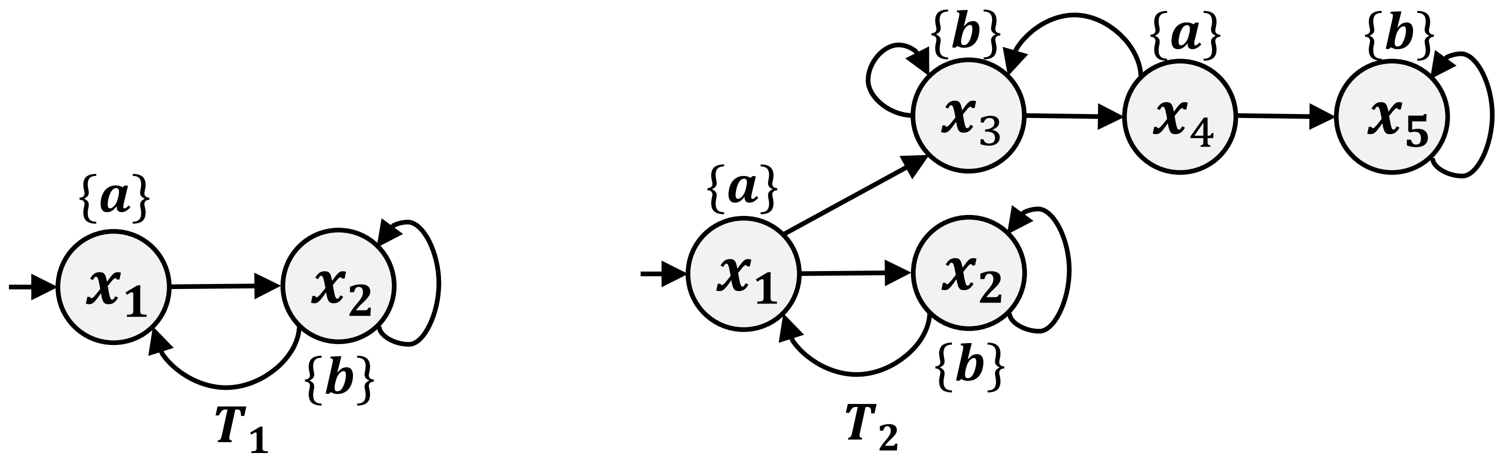
Example: Fixed-Point Iteration

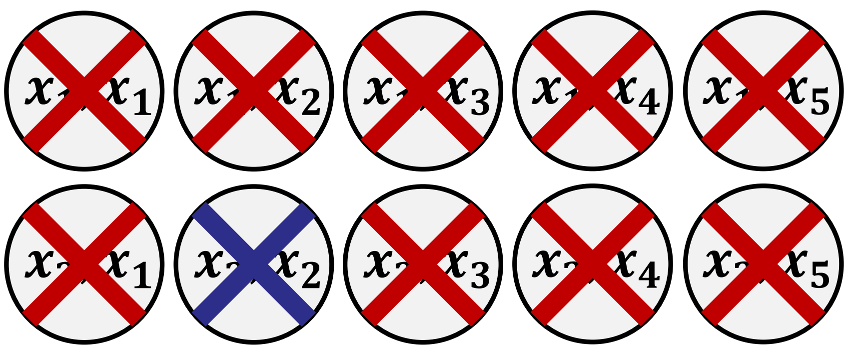




- $R_0 = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3), (x_2, x_5)\}$
- $R_1 = F(R_0) = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3)\}$
- $R_2 = F(R_1) = \{(x_1, x_1), (x_2, x_2), (x_2, x_3)\}$
- $R_3 = F(R_2) = \{(x_1, x_1), (x_2, x_2)\}$
- $R_4 = F(R_3) = \{(x_2, x_2)\}$

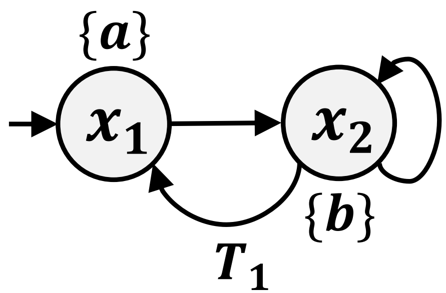
Example: Fixed-Point Iteration

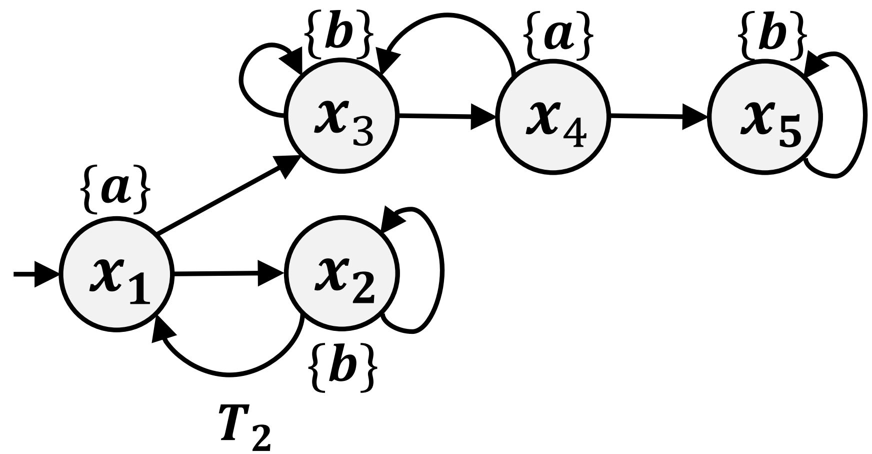


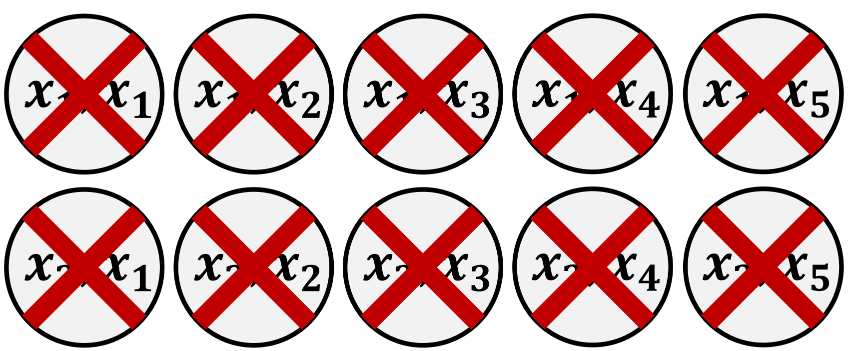


- $R_0 = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3), (x_2, x_5)\}$
- $R_1 = F(R_0) = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3)\}$
- $R_2 = F(R_1) = \{(x_1, x_1), (x_2, x_2), (x_2, x_3)\}$
- $R_3 = F(R_2) = \{(x_1, x_1), (x_2, x_2)\}$
- $R_4 = F(R_3) = \{(x_2, x_2)\}$
- $R_5 = F(R_4) = \emptyset$

Example: Fixed-Point Iteration







- $R_0 = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3), (x_2, x_5)\}$
- $R_1 = F(R_0) = \{(x_1, x_1), (x_1, x_4), (x_2, x_2), (x_2, x_3)\}$
- $R_2 = F(R_1) = \{(x_1, x_1), (x_2, x_2), (x_2, x_3)\}$
- $R_3 = F(R_2) = \{(x_1, x_1), (x_2, x_2)\}$
- $R_4 = F(R_3) = \{(x_2, x_2)\}$
- $R_5 = F(R_4) = \emptyset$

$T_1 \not\cong T_2$!

Comment on Bisimulation

- $T_1 \cong T_2$ iff each each $X_{0,i}$ is related to some $X_{0,j}$ in R^*

- Simulation implies trace inclusion, i.e.,

$$T_1 \preccurlyeq T_2 \Rightarrow Trace(T_1) \subseteq Trace(T_2)$$

- Bisimulation implies trace equivalence, i.e.,

$$T_1 \cong T_2 \Rightarrow Trace(T_1) = Trace(T_2)$$

- The vice versa is not true in general

- What if we also want to match control inputs?
  Change the definitions of the operator to

$$\forall x_1' \in Post(x_1, u), \exists x_2' \in Post(x_2, u) \cdots$$

Bisimulation on Itself

- For a single system T, we can compute the maximal bisimulation relation $\sim \subseteq X \times X$ **between T and itself** (by the fixed-point alg.)
- Note that such a relation $\sim$ is always non-empty. Why? since a state should be equivalent to itself, i.e., the identity relation is included in $\sim$
- Relation $\sim$ **is in fact an equivalent relation** telling which states are equivalent in terms of both the current property and the future
- Therefore, we can aggregate equivalent states and treat them as a new state (the equivalent classes)
- In this way, we are able to **abstract** the system model without losing any information

Quotient-Based Abstraction

Let $T = (X, U, \rightarrow, X_0, AP, L)$ be an LTS and $\sim \subseteq X \times X$ be an equivalence relation on X. Then $\sim$ induces a quotient transition system

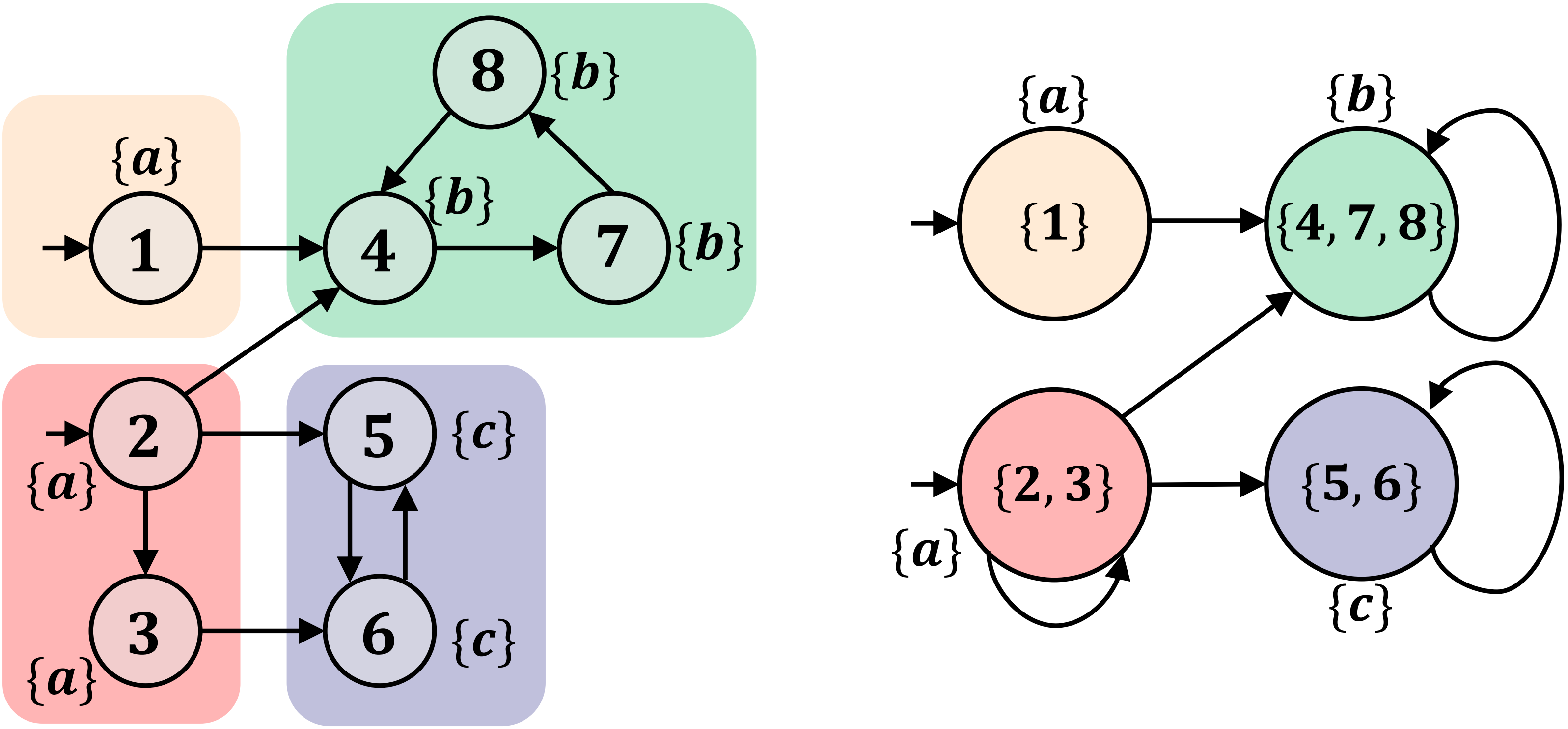
$$T/_{\sim} = (X/_{\sim}, U, \rightarrow_{\sim}, X/_{\sim,0}, AP, L_{\sim})$$

- $X/_{\sim}$ is the quotient space (the set of all equivalence classes) with $X/_{\sim,0} = \{[x] \in 2^X : [x] \cap X_0 \neq \emptyset\}$
- for $X_1, X_2 \in X/_{\sim}$ and $u \in U$, we have
$$X_1 \underset{\sim}{\overset{u}{\rightarrow}} X_2 \Leftrightarrow \exists x_1 \in X_1, \exists x_2 \in X_2 : x_1 \overset{u}{\rightarrow} x_2$$
- $L/_{\sim}(x) = L(x)$ for all $x \in X$

Theorem

- for any $\sim \subseteq X \times X$, we have $T \preccurlyeq T/_{\sim}$
- if $\sim \subseteq X \times X$ is a bisimulation relation for T, then $T \cong T/_{\sim}$

Example: Quotient System

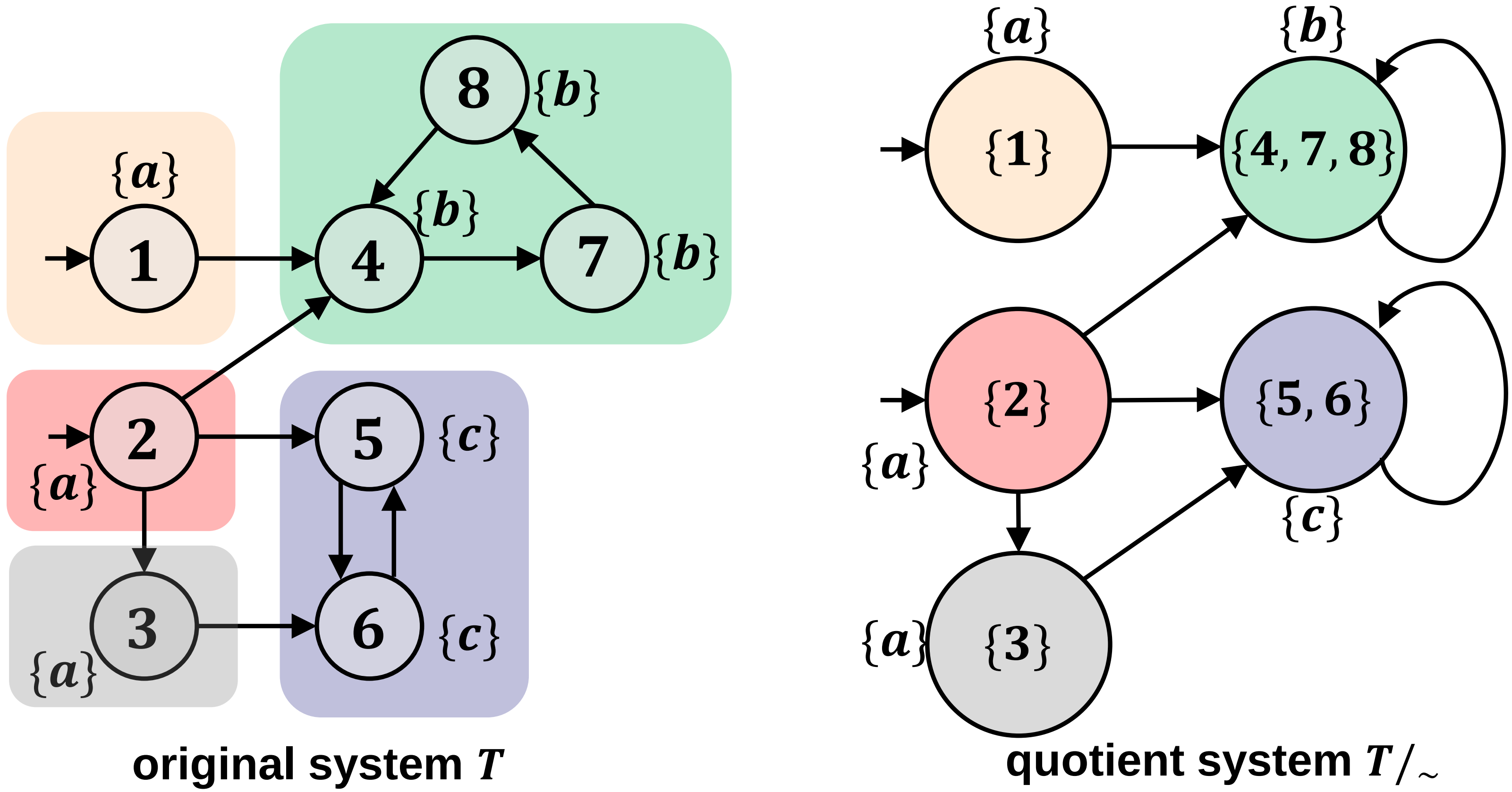


original system T

quotient system $T/_\sim$

- Consider equivalence relation shown by the colors
- Trivially, we have $T \preccurlyeq T/_\sim$
- However, $T \not\cong T/_\sim$ since $\sim$ is not a bisimulation (consider states 2&3)

Example: Quotient System

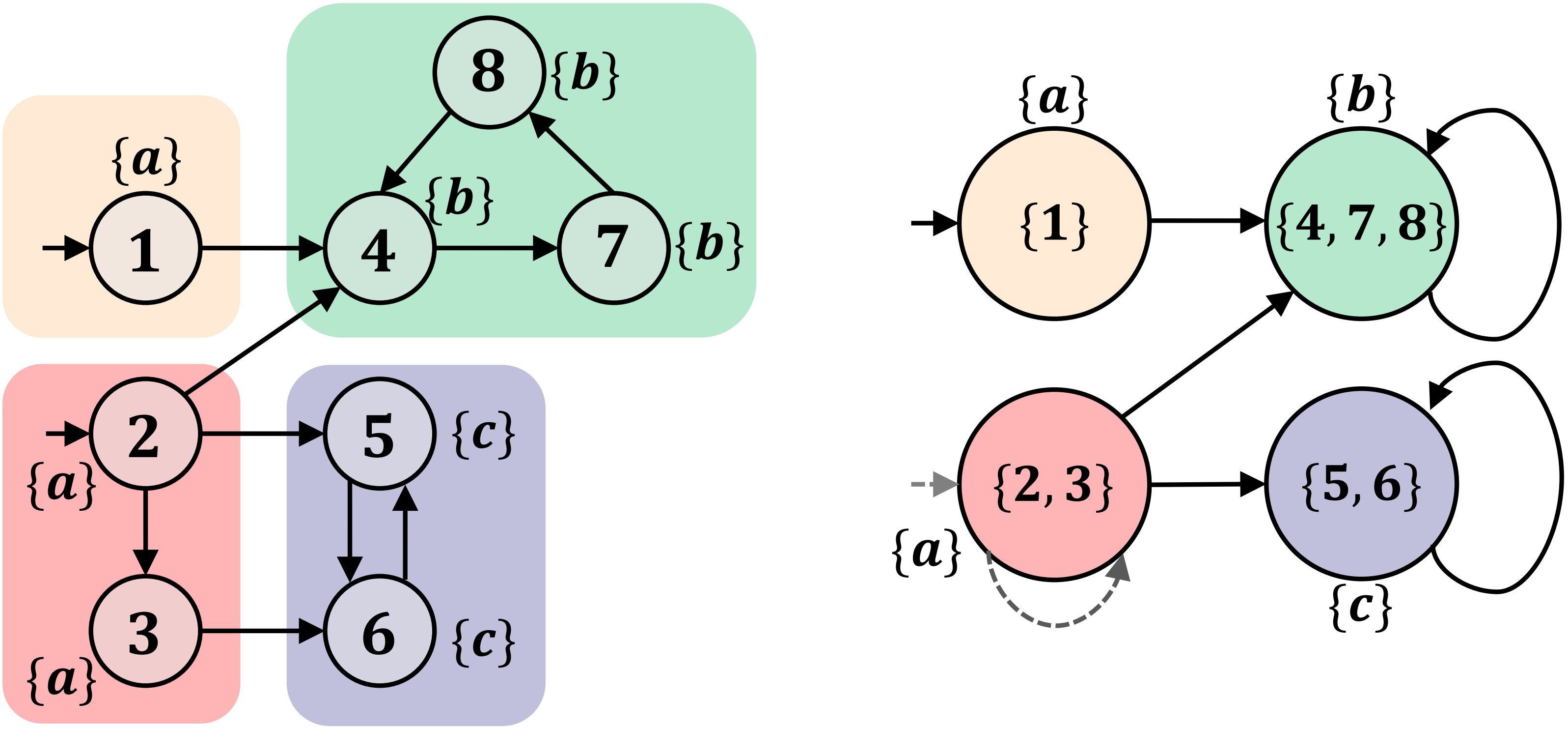


- Since $\sim \subseteq X \times X$ is a bisimulation relation on T
- This time we have $T \cong T/_{\sim}$

Under-Approx. v.s. Over-Approx.

- In $T/_\sim$, we have $X_1 \underset{\sim}{\overset{u}{\rightarrow}} X_2 \Leftrightarrow \exists x_1 \in X_1, \exists x_2 \in X_2: x_1 \overset{u}{\rightarrow} x_2$
- This is why $T \preccurlyeq T/_\sim$ and we call this over-approximation
- What if we change it to $X_1 \underset{\sim}{\overset{u}{\rightarrow}} X_2 \Leftrightarrow \forall x_1 \in X_1, \exists x_2 \in X_2: x_1 \overset{u}{\rightarrow} x_2$
- Then we have $T/_\sim \preccurlyeq T$ and we call this under-approximation
- They coincide when $\sim \subseteq X \times X$ is a bisimulation relation
- For an infinite-state system, there may not always exist a finite quotient; hence we need over/under-approximation
- Over-approximation is useful for checking safety as $Trace(T) \subseteq Trace(T/_\sim)$
- Under-approx. is useful for checking reachability as $Trace(T/_\sim) \subseteq Trace(T)$

Example: Quotient System



original system T quotient system $T/_\sim$

- Over-approximation: with dashed lines, $T \preccurlyeq T/_\sim$
- Under-approximation: without dashed lines, $T/_\sim \preccurlyeq T$

Stage Summary

- **Simulation means "no matter what you do, I can match it and preserve the ability of matching in the future"**
- **Two states are equivalent if they have both the same property and the same future behaviors**
- **Two systems are equivalent if they can simulate each other**
- **By aggregating equivalent states, one can build the quotient system that bisimulates the original system**
- **Bisimulation implies trace equivalent; hence preserves LT properties**

Review of Last Lecture

- Two different models may be essentially equivalent
- $Trace(T_1) = Trace(T_2)$ is not fine enough for model equivalence
- Simulation: $T_1 \preccurlyeq T_2$ means T_2 can “match” T_1
- Bisimulation: $T_1 \cong T_2$ means they can “match” each other
- The maximal bisimulation relation can be computed by fixed-point alg.
- An equivalence relation over X induces a quotient system $T/_\sim$
- If $\sim \subseteq X \times X$ is a bisimulation relation for T, then $T \cong T/_\sim$
- Remark: $T_1 \preccurlyeq_R T_2$ and $T_2 \preccurlyeq_{R'} T_1$ does not necessarily imply $T_1 \cong T_2$; they have to be the same relation, i.e., $R^{-1} = R'$!