

# Transition Systems

Hossein Hojjat  
University of Tehran

Based on Procestheorie(2IT30), Dr.ir. M.A. Reniers

- 1 Informal Description
- 2 Formal Description
- 3 Reviewing Some Concepts in Set Theory
- 4 Equivalence of Transition Systems (Part1)
  - Strong Isomorphism
  - Weak Isomorphism
  - Language Equivalence
  - Trace Equivalence
  - Trace Equivalence
  - Bisimulation
- 5 Composing Transition Systems
  - Sequential Composition
  - Non-deterministic Choice
  - Concurrency and Communication
  - Encapsulation
  - Abstraction
- 6 Equivalence of Transition Systems (Part2)
  - Branching Bisimulation
  - Weak Bisimulation

# Transition System vs. Automata

**Similarity:** Both contains states and labeled transitions

**Dissimilarity:** Automata accepts languages, transition system describes the behavior of interacting systems

# Transition System vs. Automata

**Similarity:** Both contains states and labeled transitions

**Dissimilarity:** Automata accepts languages, transition system describes the behavior of interacting systems

## Transition

A transition from a state  $s$  to state  $s'$  labeled by action  $a$  is written:

$$s \xrightarrow{a} s'$$

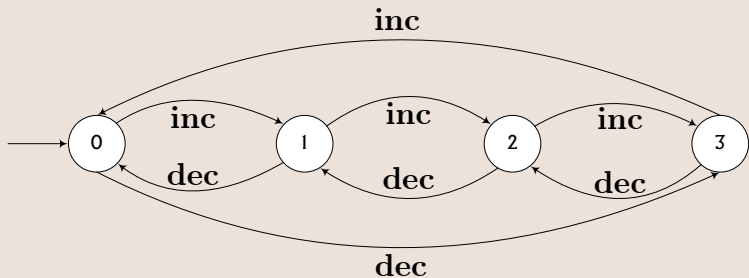
# Transition System Example

Describe a modulo-4 counter. This counter offers two actions `inc` and `dec`.

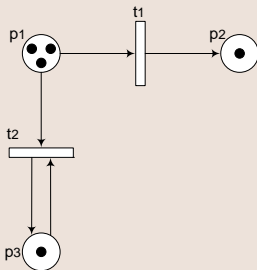
# Transition System Example

Describe a modulo-4 counter. This counter offers two actions `inc` and `dec`.

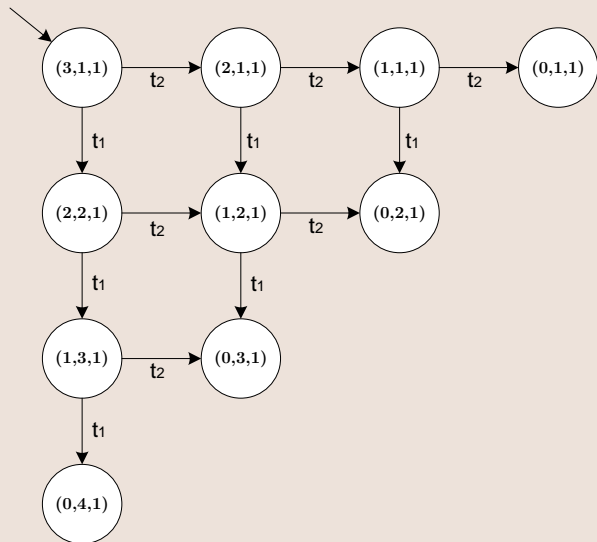
Solution



Give a transition system for the following petri net:



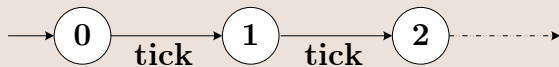
# Exercise Solution



# Infinite Example

## Clock

A clock that constantly generates ticks.



- 1 Informal Description
- 2 Formal Description
- 3 Reviewing Some Concepts in Set Theory
- 4 Equivalence of Transition Systems (Part1)
  - Strong Isomorphism
  - Weak Isomorphism
  - Language Equivalence
  - Trace Equivalence
  - Trace Equivalence
  - Bisimulation
- 5 Composing Transition Systems
  - Sequential Composition
  - Non-deterministic Choice
  - Concurrency and Communication
  - Encapsulation
  - Abstraction
- 6 Equivalence of Transition Systems (Part2)
  - Branching Bisimulation
  - Weak Bisimulation

## Transition System

A *transition system* is a five-tuple  $(S, A, \rightarrow, \downarrow, s_0)$  where

## Transition System

A *transition system* is a five-tuple  $(S, A, \rightarrow, \downarrow, s_0)$  where

- $S$  is a set of states

## Transition System

A *transition system* is a five-tuple  $(S, A, \rightarrow, \downarrow, s_0)$  where

- $S$  is a set of states
- $A$  is a set of actions

## Transition System

A *transition system* is a five-tuple  $(S, A, \rightarrow, \downarrow, s_0)$  where

- $S$  is a set of states
- $A$  is a set of actions
- $\rightarrow \subseteq S \times A \times S$  is a transition relation

## Transition System

A *transition system* is a five-tuple  $(S, A, \rightarrow, \downarrow, s_0)$  where

- $S$  is a set of states
- $A$  is a set of actions
- $\rightarrow \subseteq S \times A \times S$  is a transition relation
- $\downarrow \subseteq S$  is the set of successfully terminating states

## Transition System

A *transition system* is a five-tuple  $(S, A, \rightarrow, \downarrow, s_0)$  where

- $S$  is a set of states
- $A$  is a set of actions
- $\rightarrow \subseteq S \times A \times S$  is a transition relation
- $\downarrow \subseteq S$  is the set of successfully terminating states
- $s_0$  is the initial/start state

## Transition System

A *transition system* is a five-tuple  $(S, A, \rightarrow, \downarrow, s_0)$  where

- $S$  is a set of states
- $A$  is a set of actions
- $\rightarrow \subseteq S \times A \times S$  is a transition relation
- $\downarrow \subseteq S$  is the set of successfully terminating states
- $s_0$  is the initial/start state

- We often write  $s \xrightarrow{a} s'$  instead of  $(s, a, s') \in \rightarrow$
- We often write  $s \downarrow$  instead of  $s \in \downarrow$

## Transition System

A *transition system* is a five-tuple  $(S, A, \rightarrow, \downarrow, s_0)$  where

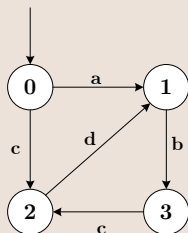
- $S$  is a set of states
- $A$  is a set of actions
- $\rightarrow \subseteq S \times A \times S$  is a transition relation
- $\downarrow \subseteq S$  is the set of successfully terminating states
- $s_0$  is the initial/start state

- We often write  $s \xrightarrow{a} s'$  instead of  $(s, a, s') \in \rightarrow$
- We often write  $s \downarrow$  instead of  $s \in \downarrow$

## Finiteness

A transition system  $T = (S, A, \rightarrow, \downarrow, s_0)$  is *finite* if both  $S$  and  $A$  are finite.

# Example



- states:  $S = \{0, 1, 2, 4\}$
- actions:  $A = \{a, b, c, d\}$
- transition relation:  $\rightarrow = \{(0, c, 2), (0, a, 1), (1, b, 3), (2, d, 1), (3, c, 2)\}$
- successful termination:  $\downarrow = \emptyset$
- start state:  $s_0 = 0$

## Generalized Transition

For a give transition relation  $\rightarrow \subseteq S \times A \times S$ , we define the *generalized transition relation*  $\twoheadrightarrow \subseteq S \times A^* \times S$  such that

## Generalized Transition

For a give transition relation  $\rightarrow \subseteq S \times A \times S$ , we define the *generalized transition relation*  $\twoheadrightarrow \subseteq S \times A^* \times S$  such that

- $s \xrightarrow{\epsilon} s$  for each  $s \in S$

## Generalized Transition

For a give transition relation  $\rightarrow \subseteq S \times A \times S$ , we define the *generalized transition relation*  $\twoheadrightarrow \subseteq S \times A^* \times S$  such that

- $s \xrightarrow{\epsilon} s$  for each  $s \in S$
- if  $s \xrightarrow{a} s'$ , then  $s \xrightarrow{a} s'$

## Generalized Transition

For a give transition relation  $\rightarrow \subseteq S \times A \times S$ , we define the *generalized transition relation*  $\twoheadrightarrow \subseteq S \times A^* \times S$  such that

- $s \xrightarrow{\epsilon} s$  for each  $s \in S$
- if  $s \xrightarrow{a} s'$ , then  $s \xrightarrow{a} s'$
- if  $s \xrightarrow{\sigma} s'$  and  $s' \xrightarrow{\sigma'} s''$ , then  $s \xrightarrow{\sigma\sigma'} s''$  in which  $\sigma$  and  $\sigma'$  are sequences over  $A$

## Generalized Transition

For a give transition relation  $\rightarrow \subseteq S \times A \times S$ , we define the *generalized transition relation*  $\twoheadrightarrow \subseteq S \times A^* \times S$  such that

- $s \xrightarrow{\epsilon} s$  for each  $s \in S$
- if  $s \xrightarrow{a} s'$ , then  $s \xrightarrow{a} s'$
- if  $s \xrightarrow{\sigma} s'$  and  $s' \xrightarrow{\sigma'} s''$ , then  $s \xrightarrow{\sigma\sigma'} s''$  in which  $\sigma$  and  $\sigma'$  are sequences over  $A$

## Reachable State

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system. A state  $s \in S$  is *reachable* if  $s_0 \xrightarrow{\sigma} s$  for some  $\sigma \in A^*$ .

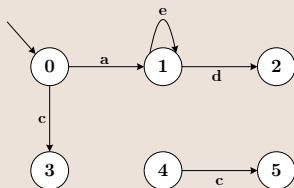
# Some Definitions (cont.)

## Terminal State

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system. A state  $s \in S$  is a *terminal* state of  $T$  if  $s \notin \downarrow$  and there are no  $a \in A$  and  $s' \in S$  such that  $s \xrightarrow{a} s'$ .

## Deadlock State

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system. A state  $s \in S$  is a *deadlock* state of  $T$  if  $s$  is reachable and terminal.



## Definition

An *automaton* is a five-tuple  $(S, A, \rightarrow, s_0, F)$  where

## Definition

An *automaton* is a five-tuple  $(S, A, \rightarrow, s_0, F)$  where

- $S$  is a set of states

## Definition

An *automaton* is a five-tuple  $(S, A, \rightarrow, s_0, F)$  where

- $S$  is a set of states
- $A$  is a set of actions

## Definition

An *automaton* is a five-tuple  $(S, A, \rightarrow, s_0, F)$  where

- $S$  is a set of states
- $A$  is a set of actions
- $\rightarrow \subseteq S \times A \times S$  is a transition relation

## Definition

An *automaton* is a five-tuple  $(S, A, \rightarrow, s_0, F)$  where

- $S$  is a set of states
- $A$  is a set of actions
- $\rightarrow \subseteq S \times A \times S$  is a transition relation
- $s_0$  is the initial state

## Definition

An *automaton* is a five-tuple  $(S, A, \rightarrow, s_0, F)$  where

- $S$  is a set of states
- $A$  is a set of actions
- $\rightarrow \subseteq S \times A \times S$  is a transition relation
- $s_0$  is the initial state
- $F \subseteq S$  is the set of final states

## Definition

An *automaton* is a five-tuple  $(S, A, \rightarrow, s_0, F)$  where

- $S$  is a set of states
- $A$  is a set of actions
- $\rightarrow \subseteq S \times A \times S$  is a transition relation
- $s_0$  is the initial state
- $F \subseteq S$  is the set of final states

Standard definition:  $\delta : S \times A \rightarrow \mathcal{P}(S)$

- $\delta(s, a) = \{s' \in S \mid s \xrightarrow{a} s'\}$

## Definition

An *automaton* is a five-tuple  $(S, A, \rightarrow, s_0, F)$  where

- $S$  is a set of states
- $A$  is a set of actions
- $\rightarrow \subseteq S \times A \times S$  is a transition relation
- $s_0$  is the initial state
- $F \subseteq S$  is the set of final states

Standard definition:  $\delta : S \times A \rightarrow \mathcal{P}(S)$

- $\delta(s, a) = \{s' \in S \mid s \xrightarrow{a} s'\}$

## Automaton Language

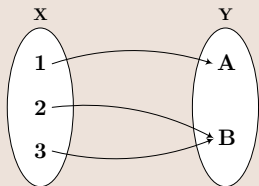
$$\mathcal{L}(M) = \{\sigma \in A^* \mid s_0 \xrightarrow{\sigma} s \text{ for some } s \in F\}$$

- 1 Informal Description
- 2 Formal Description
- 3 Reviewing Some Concepts in Set Theory
- 4 Equivalence of Transition Systems (Part1)
  - Strong Isomorphism
  - Weak Isomorphism
  - Language Equivalence
  - Trace Equivalence
  - Trace Equivalence
  - Bisimulation
- 5 Composing Transition Systems
  - Sequential Composition
  - Non-deterministic Choice
  - Concurrency and Communication
  - Encapsulation
  - Abstraction
- 6 Equivalence of Transition Systems (Part2)
  - Branching Bisimulation
  - Weak Bisimulation

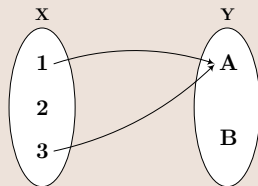
# Surjective Function

## Definition

A function  $f$  is said to be *surjective* if its values span its whole codomain



A surjective function

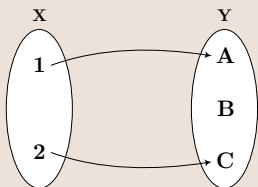


A non-surjective function

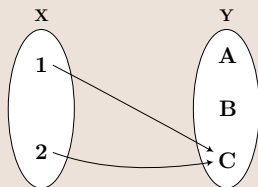
# Injective Function

## Definition

An *injective* function is a function which associates distinct arguments to distinct values



An injective function

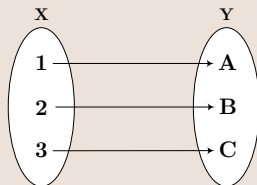


A non-injective function

# Bijjective Function

## Definition

A function  $f$  is a *bijjective* function if it is both injective and surjective. (This is often called a “one-to-one correspondence”.)



A bijective function

## Definition

A relation  $R \subseteq X \times X$  is an equivalence (relation) if and only if

## Definition

A relation  $R \subseteq X \times X$  is an equivalence (relation) if and only if

- **Reflexive:** for all  $x \in X : (x, x) \in R$

## Definition

A relation  $R \subseteq X \times X$  is an equivalence (relation) if and only if

- **Reflexive:** for all  $x \in X : (x, x) \in R$
- **Symmetric:** for all  $x, y \in X : \text{if } (x, y) \in R, \text{ then } (y, x) \in R$

## Definition

A relation  $R \subseteq X \times X$  is an equivalence (relation) if and only if

- **Reflexive:** for all  $x \in X : (x, x) \in R$
- **Symmetric:** for all  $x, y \in X : \text{if } (x, y) \in R, \text{ then } (y, x) \in R$
- **Transitive:** for all  $x, y, z \in X : \text{if } (x, y) \in R \text{ and } (y, z) \in R \text{ then } (x, z) \in R$

- 1 Informal Description
- 2 Formal Description
- 3 Reviewing Some Concepts in Set Theory
- 4 Equivalence of Transition Systems (Part1)**
  - Strong Isomorphism
  - Weak Isomorphism
  - Language Equivalence
  - Trace Equivalence
  - Trace Equivalence
  - Bisimulation
- 5 Composing Transition Systems
  - Sequential Composition
  - Non-deterministic Choice
  - Concurrency and Communication
  - Encapsulation
  - Abstraction
- 6 Equivalence of Transition Systems (Part2)
  - Branching Bisimulation
  - Weak Bisimulation

# Variety of Descriptions

- For a single system, many descriptions are conceivable

# Variety of Descriptions

- For a single system, many descriptions are conceivable
  - Depending on the aspects of the system being considered

# Variety of Descriptions

- For a single system, many descriptions are conceivable
  - Depending on the aspects of the system being considered
  - Depending on the purpose of description

# Variety of Descriptions

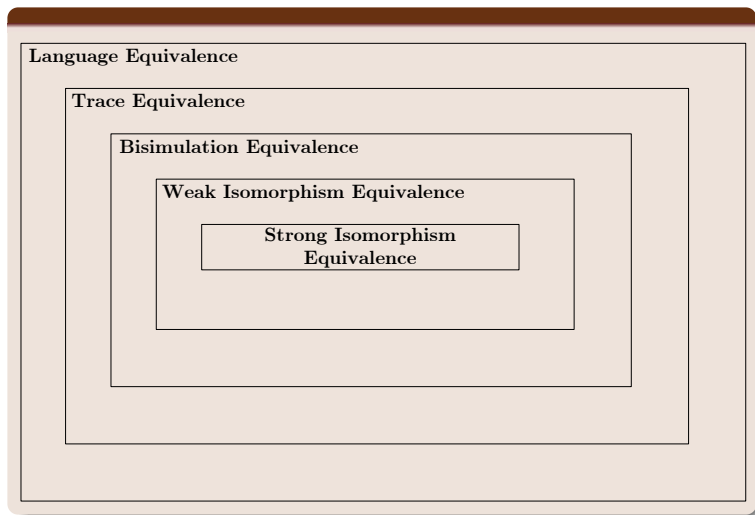
- For a single system, many descriptions are conceivable
  - Depending on the aspects of the system being considered
  - Depending on the purpose of description
- In this lecture we assume the aspects of interest are the actions

# Variety of Descriptions

- For a single system, many descriptions are conceivable
  - Depending on the aspects of the system being considered
  - Depending on the purpose of description
- In this lecture we assume the aspects of interest are the actions
- There are numerous notions of equivalency for transition systems in the literature

# Variety of Descriptions

- For a single system, many descriptions are conceivable
    - Depending on the aspects of the system being considered
    - Depending on the purpose of description
  - In this lecture we assume the aspects of interest are the actions
- 
- There are numerous notions of equivalency for transition systems in the literature
  - We consider these equivalencies:
    - Mathematical equivalence
    - Strong isomorphism
    - Weak isomorphism
    - Language equivalence
    - Trace equivalence
    - Bisimulation equivalence



## Definition

Two transition systems  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  are *strongly isomorphic* iff there exists a bijective function  $h : S \rightarrow S'$  such that for all  $s_0, s_1, s_2 \in S$ ,  $s'_0, s'_1, s'_2 \in S'$  and  $a \in A \cup A'$ :

## Definition

Two transition systems  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  are *strongly isomorphic* iff there exists a bijective function  $h : S \rightarrow S'$  such that for all  $s_0, s_1, s_2 \in S$ ,  $s'_0, s'_1, s'_2 \in S'$  and  $a \in A \cup A'$ :

①  $h(s_0) = s'_0$

## Definition

Two transition systems  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  are *strongly isomorphic* iff there exists a bijective function  $h : S \rightarrow S'$  such that for all  $s_0, s_1, s_2 \in S$ ,  $s'_0, s'_1, s'_2 \in S'$  and  $a \in A \cup A'$ :

- 1  $h(s_0) = s'_0$
- 2 if  $h(s_1) = s'_1$  and  $h(s_2) = s'_2$  then:  $s_1 \xrightarrow{a} s_2$  iff  $s'_1 \xrightarrow{a} s'_2$

# Strong Isomorphism

## Definition

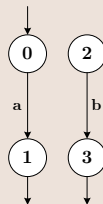
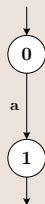
Two transition systems  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  are *strongly isomorphic* iff there exists a bijective function  $h : S \rightarrow S'$  such that for all  $s_0, s_1, s_2 \in S$ ,  $s'_0, s'_1, s'_2 \in S'$  and  $a \in A \cup A'$ :

- 1  $h(s_0) = s'_0$
- 2 if  $h(s_1) = s'_1$  and  $h(s_2) = s'_2$  then:  $s_1 \xrightarrow{a} s_2$  iff  $s'_1 \xrightarrow{a} s'_2$
- 3 if  $h(s_1) = s'_1$  then:  $s_1 \downarrow$  iff  $s'_1 \downarrow'$

## Theorem

*Strong isomorphism is an equivalence relation.*

Are these transition diagrams isomorphic?



## Reachable States Set

The set of *reachable states* of  $T$ ,  $\text{reach}(T)$  is defined as:

$$\text{reach}(T) = \{s \in S \mid s_0 \xrightarrow{\sigma} s \text{ for some } \sigma \in A^*\}$$

## Reachable States Set

The set of *reachable states* of  $T$ ,  $\text{reach}(T)$  is defined as:

$$\text{reach}(T) = \{s \in S \mid s_0 \xrightarrow{\sigma} s \text{ for some } \sigma \in A^*\}$$

## Definition

Two transition systems  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  are *weakly isomorphic*  $T \cong T'$ , iff there exists a bijective function  $h : \text{reach}(T) \rightarrow \text{reach}(T')$  such that for all  $s_0, s_1, s_2 \in S$ ,  $s'_0, s'_1, s'_2 \in S'$  and  $a \in A \cup A'$ :

- 1  $h(s_0) = s'_0$

## Reachable States Set

The set of *reachable states* of  $T$ ,  $\text{reach}(T)$  is defined as:

$$\text{reach}(T) = \{s \in S \mid s_0 \xrightarrow{\sigma} s \text{ for some } \sigma \in A^*\}$$

## Definition

Two transition systems  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  are *weakly isomorphic*  $T \cong T'$ , iff there exists a bijective function  $h : \text{reach}(T) \rightarrow \text{reach}(T')$  such that for all  $s_0, s_1, s_2 \in S$ ,  $s'_0, s'_1, s'_2 \in S'$  and  $a \in A \cup A'$ :

- 1  $h(s_0) = s'_0$
- 2 if  $h(s_1) = s'_1$  and  $h(s_2) = s'_2$  then:  $s_1 \xrightarrow{a} s_2$  iff  $s'_1 \xrightarrow{a} s'_2$

## Reachable States Set

The set of *reachable states* of  $T$ ,  $\text{reach}(T)$  is defined as:

$$\text{reach}(T) = \{s \in S \mid s_0 \xrightarrow{\sigma} s \text{ for some } \sigma \in A^*\}$$

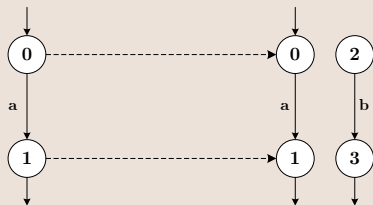
## Definition

Two transition systems  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  are *weakly isomorphic*  $T \cong T'$ , iff there exists a bijective function  $h: \text{reach}(T) \rightarrow \text{reach}(T')$  such that for all  $s_0, s_1, s_2 \in S$ ,  $s'_0, s'_1, s'_2 \in S'$  and  $a \in A \cup A'$ :

- 1  $h(s_0) = s'_0$
- 2 if  $h(s_1) = s'_1$  and  $h(s_2) = s'_2$  then:  $s_1 \xrightarrow{a} s_2$  iff  $s'_1 \xrightarrow{a} s'_2$
- 3 if  $h(s_1) = s'_1$  then:  $s_1 \downarrow$  iff  $s'_1 \downarrow'$

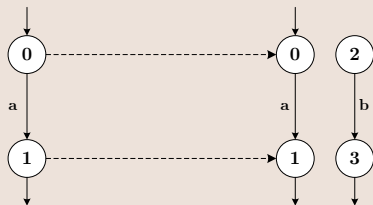
## Weak Isomorphism (cont.)

These transition systems are weakly isomorphic, as on the reachably states the function  $h$  is bijective.



## Weak Isomorphism (cont.)

These transition systems are weakly isomorphic, as on the reachable states the function  $h$  is bijective.

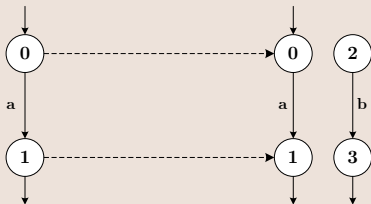


### Theorem

*If two transition systems are strongly isomorphic, then they are weakly isomorphic.*

## Weak Isomorphism (cont.)

These transition systems are weakly isomorphic, as on the reachable states the function  $h$  is bijective.



### Theorem

*If two transition systems are strongly isomorphic, then they are weakly isomorphic.*

### Theorem

*Weak isomorphism is an equivalence relation.*

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system. The *reduction* of  $T$ ,  $\text{red}(T)$ , is the transition system  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  where

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system. The *reduction* of  $T$ ,  $\text{red}(T)$ , is the transition system  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  where

- $S' = \text{reach}(T)$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system. The *reduction* of  $T$ ,  $\text{red}(T)$ , is the transition system  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  where

- $S' = \text{reach}(T)$
- $A' = \{a \in A \mid s \xrightarrow{a} s' \text{ for some } s, s' \in S'\}$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system. The *reduction* of  $T$ ,  $\text{red}(T)$ , is the transition system  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  where

- $S' = \text{reach}(T)$
- $A' = \{a \in A \mid s \xrightarrow{a} s' \text{ for some } s, s' \in S'\}$
- $\rightarrow' = \rightarrow \cup (S' \times A' \times S')$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system. The *reduction* of  $T$ ,  $\text{red}(T)$ , is the transition system  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  where

- $S' = \text{reach}(T)$
- $A' = \{a \in A \mid s \xrightarrow{a} s' \text{ for some } s, s' \in S'\}$
- $\rightarrow' = \rightarrow \cup (S' \times A' \times S')$
- $\downarrow' = \downarrow \cup S'$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system. The *reduction* of  $T$ ,  $\text{red}(T)$ , is the transition system  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  where

- $S' = \text{reach}(T)$
- $A' = \{a \in A \mid s \xrightarrow{a} s' \text{ for some } s, s' \in S'\}$
- $\rightarrow' = \rightarrow \cup (S' \times A' \times S')$
- $\downarrow' = \downarrow \cup S'$
- $s'_0 = s_0$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system. The *reduction* of  $T$ ,  $\text{red}(T)$ , is the transition system  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  where

- $S' = \text{reach}(T)$
- $A' = \{a \in A \mid s \xrightarrow{a} s' \text{ for some } s, s' \in S'\}$
- $\rightarrow' = \rightarrow \cup (S' \times A' \times S')$
- $\downarrow' = \downarrow \cup S'$
- $s'_0 = s_0$

## Theorem

$$T \cong \text{red}(T)$$

## Definition

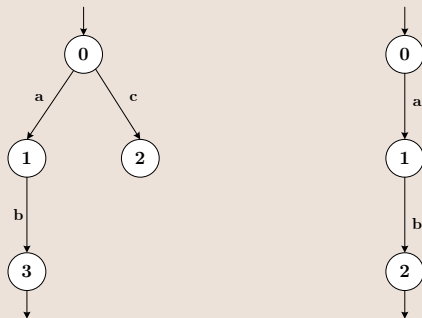
Two transition systems  $T$  and  $T'$  are *language equivalent*, written  $T \equiv_l T'$ , iff  $\mathcal{L}(T) = \mathcal{L}(T')$ .

# Language Equivalence

## Definition

Two transition systems  $T$  and  $T'$  are *language equivalent*, written  $T \equiv_l T'$ , iff  $\mathcal{L}(T) = \mathcal{L}(T')$ .

## Example



# Some Theorems on Language Equivalence

## Theorem

*Language equivalence is an equivalence relation.*

# Some Theorems on Language Equivalence

## Theorem

*Language equivalence is an equivalence relation.*

## Theorem

*Any two isomorphic transition systems are also language equivalent: if  $T \cong T'$ , then  $T \equiv_l T'$ .*

# Trace Equivalence

Important property of language equivalence: Only the sequences of actions are considered that end in successfully terminating states  
In some cases, all the sequences of start state is important

# Trace Equivalence

Important property of language equivalence: Only the sequences of actions are considered that end in successfully terminating states

In some cases, all the sequences of start state is important

## Traces

The set of all traces of  $T$ , notation  $\text{traces}(T)$ , is defined as

$$\text{traces}(T) = \{\sigma \in A^* \mid s_0 \xrightarrow{\sigma} s \text{ for some } s \in S\}$$

# Trace Equivalence

Important property of language equivalence: Only the sequences of actions are considered that end in successfully terminating states

In some cases, all the sequences of start state is important

## Traces

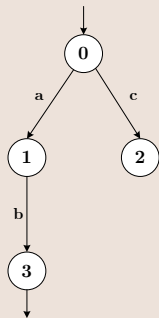
The set of all traces of  $T$ , notation  $\text{traces}(T)$ , is defined as

$$\text{traces}(T) = \{\sigma \in A^* \mid s_0 \xrightarrow{\sigma} s \text{ for some } s \in S\}$$

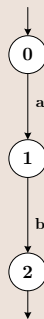
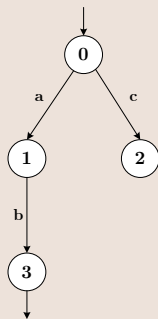
## Definition

Two transition systems  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  are *trace equivalent*,  $T \equiv_{\text{tr}} T'$ , iff  $\mathcal{L}(T) = \mathcal{L}(T')$  and  $\text{traces}(T) = \text{traces}(T')$ .

# Example

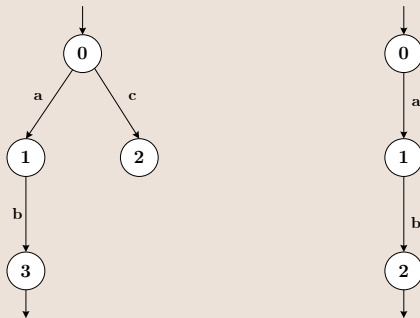


# Example



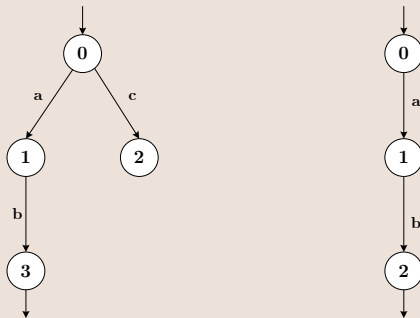
- Traces of the first transition system:  $\{\epsilon, a, ab, c\}$

# Example



- Traces of the first transition system:  $\{\epsilon, a, ab, c\}$
- Traces of the second transition system:  $\{\epsilon, a, ab\}$

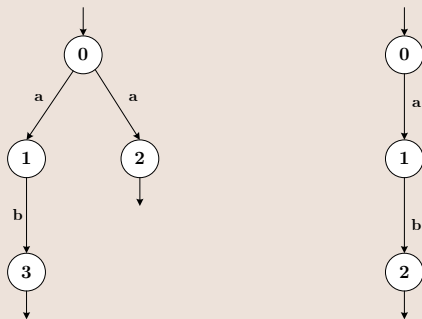
# Example



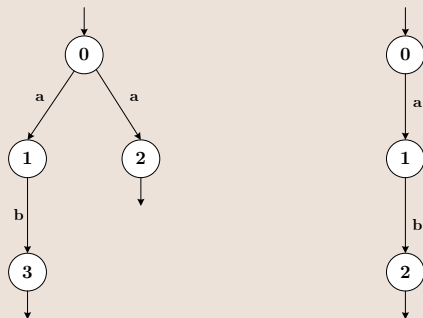
- Traces of the first transition system:  $\{\epsilon, a, ab, c\}$
- Traces of the second transition system:  $\{\epsilon, a, ab\}$

They are not trace equivalent!

# Example

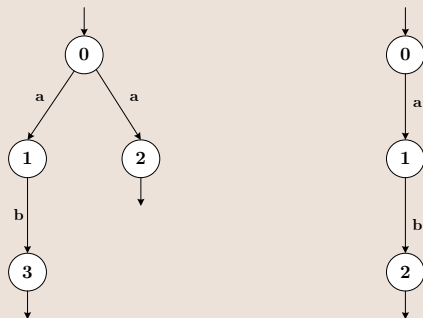


# Example



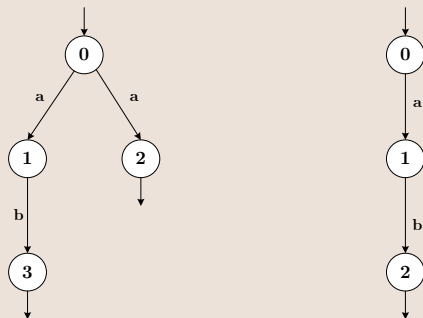
- Traces of the first transition system:  $\{\epsilon, a, ab\}$

# Example



- Traces of the first transition system:  $\{\epsilon, a, ab\}$
- Traces of the second transition system:  $\{\epsilon, a, ab\}$

# Example



- Traces of the first transition system:  $\{\epsilon, a, ab\}$
- Traces of the second transition system:  $\{\epsilon, a, ab\}$

But they are *not* trace equivalent, because of the languages.

# Some Theorems on Language Equivalence

## Theorem

*Trace equivalence is an equivalence relation.*

# Some Theorems on Language Equivalence

## Theorem

*Trace equivalence is an equivalence relation.*

## Theorem

*If any two transition systems are isomorphic, they are trace equivalent: if  $T \cong T'$ , then  $T \equiv_{tr} T'$ .*

# Some Theorems on Language Equivalence

## Theorem

*Trace equivalence is an equivalence relation.*

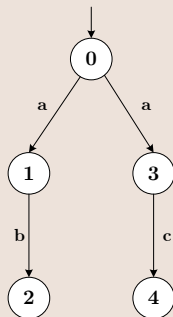
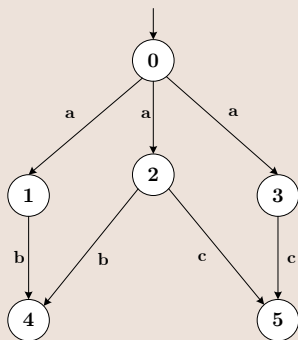
## Theorem

*If any two transition systems are isomorphic, they are trace equivalent: if  $T \cong T'$ , then  $T \equiv_{tr} T'$ .*

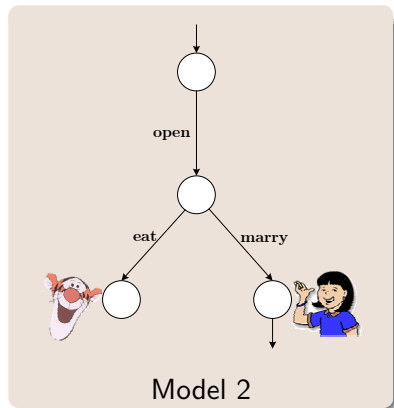
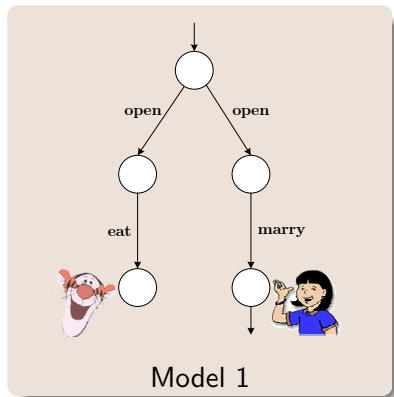
## Theorem

*If two transition systems are trace equivalent, they are language equivalent: if  $T \equiv_{tr} T'$ , then  $T \equiv_l T'$ .*

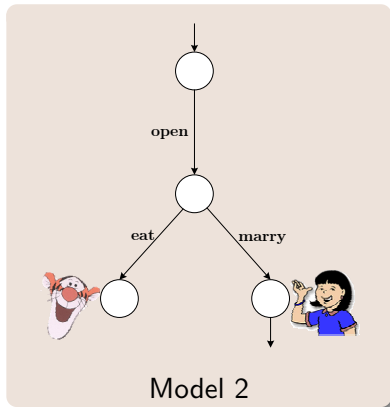
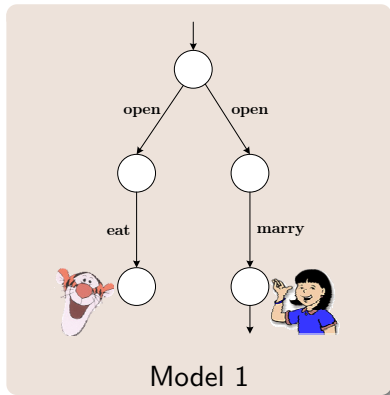
Determine whether the following transition systems are language equivalent, trace equivalent and isomorphic.



# The lady or the tiger?

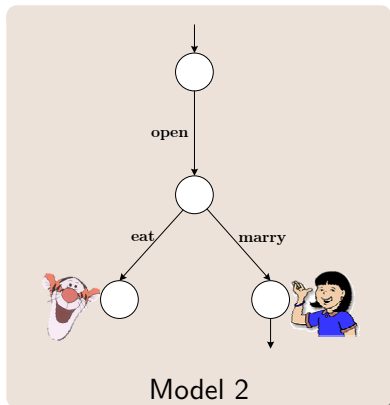
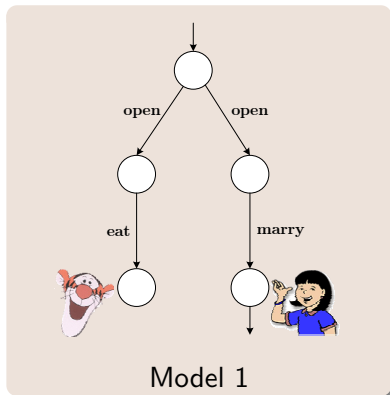


# The lady or the tiger?



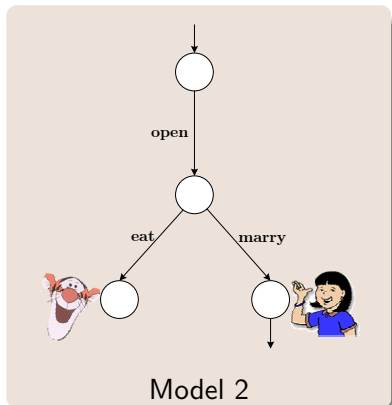
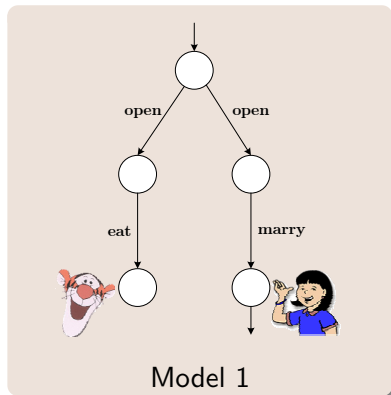
- These system are trace equivalent

# The lady or the tiger?



- These systems are trace equivalent
- In model 2 the choice is made after opening the door

# The lady or the tiger?



- These systems are trace equivalent
- In model 2 the choice is made after opening the door
- In model 1 the choice is non-deterministic

# Bisimulation Equivalence

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems. A bisimulation  $B$  between  $T$  and  $T'$  is a binary relation  $B \subseteq S \times S'$  such that

# Bisimulation Equivalence

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems. A bisimulation  $B$  between  $T$  and  $T'$  is a binary relation  $B \subseteq S \times S'$  such that

- $B(s_0, s'_0)$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems. A bisimulation  $B$  between  $T$  and  $T'$  is a binary relation  $B \subseteq S \times S'$  such that

- $B(s_0, s'_0)$
- if  $B(s_1, s'_1)$  and  $s_1 \xrightarrow{a} s_2$ , then there is a  $s'_2 \in S'$  such that  $s'_1 \xrightarrow{a} s'_2$  and  $B(s_2, s'_2)$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems. A bisimulation  $B$  between  $T$  and  $T'$  is a binary relation  $B \subseteq S \times S'$  such that

- $B(s_0, s'_0)$
- if  $B(s_1, s'_1)$  and  $s_1 \xrightarrow{a} s_2$ , then there is a  $s'_2 \in S'$  such that  $s'_1 \xrightarrow{a} s'_2$  and  $B(s_2, s'_2)$
- if  $B(s_1, s'_1)$  and  $s'_1 \xrightarrow{a} s'_2$ , then there is a  $s_2 \in S$  such that  $s_1 \xrightarrow{a} s_2$  and  $B(s_2, s'_2)$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems. A bisimulation  $B$  between  $T$  and  $T'$  is a binary relation  $B \subseteq S \times S'$  such that

- $B(s_0, s'_0)$
- if  $B(s_1, s'_1)$  and  $s_1 \xrightarrow{a} s_2$ , then there is a  $s'_2 \in S'$  such that  $s'_1 \xrightarrow{a} s'_2$  and  $B(s_2, s'_2)$
- if  $B(s_1, s'_1)$  and  $s'_1 \xrightarrow{a} s'_2$ , then there is a  $s_2 \in S$  such that  $s_1 \xrightarrow{a} s_2$  and  $B(s_2, s'_2)$
- if  $B(s, s')$  and  $s \downarrow$ , then  $s' \downarrow'$ .

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems. A bisimulation  $B$  between  $T$  and  $T'$  is a binary relation  $B \subseteq S \times S'$  such that

- $B(s_0, s'_0)$
- if  $B(s_1, s'_1)$  and  $s_1 \xrightarrow{a} s_2$ , then there is a  $s'_2 \in S'$  such that  $s'_1 \xrightarrow{a} s'_2$  and  $B(s_2, s'_2)$
- if  $B(s_1, s'_1)$  and  $s'_1 \xrightarrow{a} s'_2$ , then there is a  $s_2 \in S$  such that  $s_1 \xrightarrow{a} s_2$  and  $B(s_2, s'_2)$
- if  $B(s, s')$  and  $s \downarrow$ , then  $s' \downarrow'$ .
- if  $B(s, s')$  and  $s' \downarrow'$ , then  $s \downarrow$ .

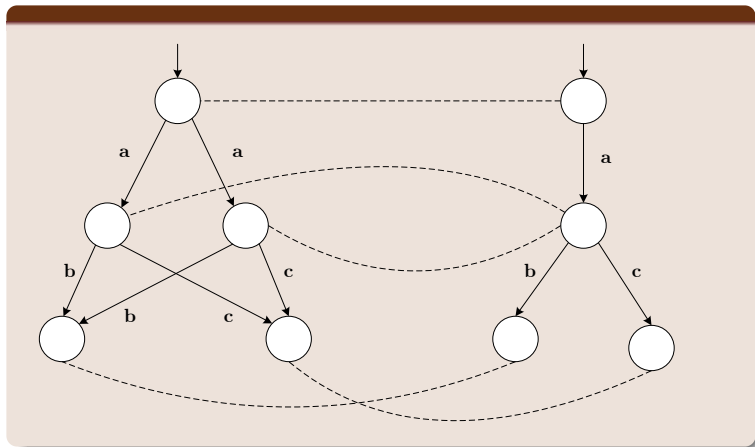
## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems. A bisimulation  $B$  between  $T$  and  $T'$  is a binary relation  $B \subseteq S \times S'$  such that

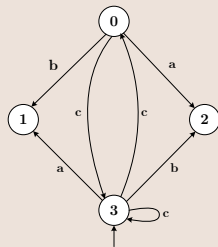
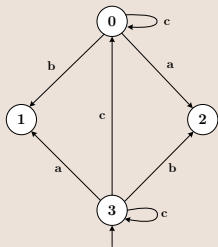
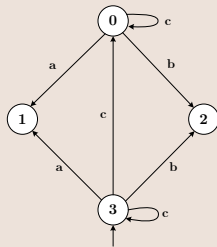
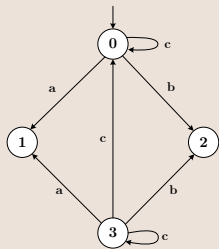
- $B(s_0, s'_0)$
- if  $B(s_1, s'_1)$  and  $s_1 \xrightarrow{a} s_2$ , then there is a  $s'_2 \in S'$  such that  $s'_1 \xrightarrow{a} s'_2$  and  $B(s_2, s'_2)$
- if  $B(s_1, s'_1)$  and  $s'_1 \xrightarrow{a} s'_2$ , then there is a  $s_2 \in S$  such that  $s_1 \xrightarrow{a} s_2$  and  $B(s_2, s'_2)$
- if  $B(s, s')$  and  $s \downarrow$ , then  $s' \downarrow'$ .
- if  $B(s, s')$  and  $s' \downarrow'$ , then  $s \downarrow$ .

$T$  and  $T'$  are bisimulation equivalent, notation  $T \leftrightarrow T'$ , iff there exists a bisimulation between  $T$  and  $T'$ .

# Example



# Exercise





- **Mathematical equivalence:** the (formal) definitions of the transition systems are the same from a mathematical perspective. Notation: =

- **Mathematical equivalence:** the (formal) definitions of the transition systems are the same from a mathematical perspective. Notation: =
- **Strong Isomorphism:** the transition systems are identical except for the names of the states.

- **Mathematical equivalence:** the (formal) definitions of the transition systems are the same from a mathematical perspective. Notation:  $=$
- **Strong Isomorphism:** the transition systems are identical except for the names of the states.
- **Weak Isomorphism:** the transition systems are strongly isomorphic provided that the transition systems are restricted to the reachable states. Notation:  $\cong$

- **Mathematical equivalence:** the (formal) definitions of the transition systems are the same from a mathematical perspective. Notation:  $=$
- **Strong Isomorphism:** the transition systems are identical except for the names of the states.
- **Weak Isomorphism:** the transition systems are strongly isomorphic provided that the transition systems are restricted to the reachable states. Notation:  $\cong$
- **Language Equivalence:** the transition systems have the same language. Notation:  $\equiv_l$

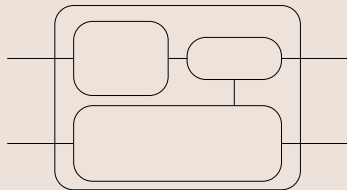
- **Mathematical equivalence:** the (formal) definitions of the transition systems are the same from a mathematical perspective. Notation:  $=$
- **Strong Isomorphism:** the transition systems are identical except for the names of the states.
- **Weak Isomorphism:** the transition systems are strongly isomorphic provided that the transition systems are restricted to the reachable states. Notation:  $\cong$
- **Language Equivalence:** the transition systems have the same language. Notation:  $\equiv_l$
- **Trace Equivalence:** the transition systems have the same traces and the same language. Notation:  $\equiv_{tr}$

- **Mathematical equivalence:** the (formal) definitions of the transition systems are the same from a mathematical perspective. Notation:  $=$
- **Strong Isomorphism:** the transition systems are identical except for the names of the states.
- **Weak Isomorphism:** the transition systems are strongly isomorphic provided that the transition systems are restricted to the reachable states. Notation:  $\cong$
- **Language Equivalence:** the transition systems have the same language. Notation:  $\equiv_l$
- **Trace Equivalence:** the transition systems have the same traces and the same language. Notation:  $\equiv_{tr}$
- **Bisimulation Equivalence:** the transition systems have the same traces, the same languages and the same moments of choice. Notation:  $\leftrightarrow$

- 1 Informal Description
- 2 Formal Description
- 3 Reviewing Some Concepts in Set Theory
- 4 Equivalence of Transition Systems (Part1)
  - Strong Isomorphism
  - Weak Isomorphism
  - Language Equivalence
  - Trace Equivalence
  - Trace Equivalence
  - Bisimulation
- 5 Composing Transition Systems
  - Sequential Composition
  - Non-deterministic Choice
  - Concurrency and Communication
  - Encapsulation
  - Abstraction
- 6 Equivalence of Transition Systems (Part2)
  - Branching Bisimulation
  - Weak Bisimulation

# Modelling

- Complex systems consist of a number of components that executes in parallel
- The components interact with each other and with the environment of the system



## Parallelism Forms

## Parallelism Forms

- **True Concurrency:** actions from different components can occur in arbitrary order also simultaneously

## Parallelism Forms

- **True Concurrency:** actions from different components can occur in arbitrary order also simultaneously
- **Interleaving:** actions from different components occur in arbitrary order, but not simultaneously

# Parallelism and Interaction

## Parallelism Forms

- **True Concurrency:** actions from different components can occur in arbitrary order also simultaneously
- **Interleaving:** actions from different components occur in arbitrary order, but not simultaneously

## Interaction Forms

# Parallelism and Interaction

## Parallelism Forms

- **True Concurrency:** actions from different components can occur in arbitrary order also simultaneously
- **Interleaving:** actions from different components occur in arbitrary order, but not simultaneously

## Interaction Forms

- Shared variables

# Parallelism and Interaction

## Parallelism Forms

- **True Concurrency:** actions from different components can occur in arbitrary order also simultaneously
- **Interleaving:** actions from different components occur in arbitrary order, but not simultaneously

## Interaction Forms

- Shared variables
- Synchronous communication

# Parallelism and Interaction

## Parallelism Forms

- **True Concurrency:** actions from different components can occur in arbitrary order also simultaneously
- **Interleaving:** actions from different components occur in arbitrary order, but not simultaneously

## Interaction Forms

- Shared variables
- Synchronous communication
- Asynchronous communication

# Parallelism and Interaction

## Parallelism Forms

- **True Concurrency:** actions from different components can occur in arbitrary order also simultaneously
- **Interleaving:** actions from different components occur in arbitrary order, but not simultaneously

## Interaction Forms

- Shared variables
- Synchronous communication
- Asynchronous communication

Here we will consider interleaving and synchronous communication

## Idea

The behavior described by the first transition system is, upon successful termination, followed by the behavior described by the second transition system

# Sequential Composition

## Idea

The behavior described by the first transition system is, upon successful termination, followed by the behavior described by the second transition system

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems such that  $S \cap S' = \emptyset$ . The *Sequential Composition* of  $T$  and  $T'$ , written  $T.T'$ , is the transition system  $T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

- $S'' = S \cup S'$

# Sequential Composition

## Idea

The behavior described by the first transition system is, upon successful termination, followed by the behavior described by the second transition system

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems such that  $S \cap S' = \emptyset$ . The *Sequential Composition* of  $T$  and  $T'$ , written  $T.T'$ , is the transition system  $T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

- $S'' = S \cup S'$
- $A'' = A \cup A'$

# Sequential Composition

## Idea

The behavior described by the first transition system is, upon successful termination, followed by the behavior described by the second transition system

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems such that  $S \cap S' = \emptyset$ . The *Sequential Composition* of  $T$  and  $T'$ , written  $T.T'$ , is the transition system  $T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

- $S'' = S \cup S'$
- $A'' = A \cup A'$
- $\rightarrow'' = \rightarrow \cup \rightarrow' \cup \rightarrow^n$  where  $s_1 \xrightarrow{a} s'_1$  iff  $s_1 \downarrow$  and  $s'_0 \xrightarrow{a} s'_1$

# Sequential Composition

## Idea

The behavior described by the first transition system is, upon successful termination, followed by the behavior described by the second transition system

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems such that  $S \cap S' = \emptyset$ . The *Sequential Composition* of  $T$  and  $T'$ , written  $T.T'$ , is the transition system  $T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

- $S'' = S \cup S'$
- $A'' = A \cup A'$
- $\rightarrow'' = \rightarrow \cup \rightarrow' \cup \rightarrow^n$  where  $s_1 \xrightarrow{a} s'_1$  iff  $s_1 \downarrow$  and  $s'_0 \xrightarrow{a} s'_1$
- $\downarrow'' = \downarrow' \cup \{s \mid s \downarrow \wedge s'_0 \downarrow'\}$

# Sequential Composition

## Idea

The behavior described by the first transition system is, upon successful termination, followed by the behavior described by the second transition system

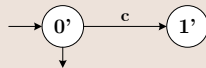
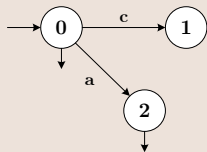
## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems such that  $S \cap S' = \emptyset$ . The *Sequential Composition* of  $T$  and  $T'$ , written  $T.T'$ , is the transition system  $T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

- $S'' = S \cup S'$
- $A'' = A \cup A'$
- $\rightarrow'' = \rightarrow \cup \rightarrow' \cup \rightarrow^n$  where  $s_1 \xrightarrow{a} s'_1$  iff  $s_1 \downarrow$  and  $s'_0 \xrightarrow{a} s'_1$
- $\downarrow'' = \downarrow' \cup \{s \mid s \downarrow \wedge s'_0 \downarrow'\}$
- $s''_0 = s_0$

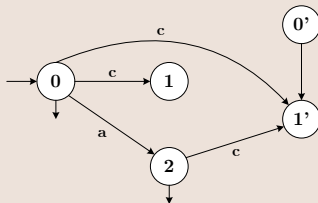
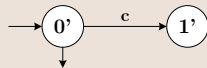
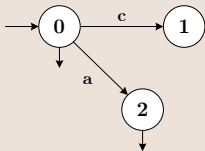
# Exercise

Find the sequential composition of the following transition diagrams:



# Exercise

Find the sequential composition of the following transition diagrams:



## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems such that  $S \cap S' = \emptyset$ . The *Non-deterministic choice* of  $T$  and  $T'$ , written  $T + T'$ , is the transition system where

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems such that  $S \cap S' = \emptyset$ . The *Non-deterministic choice* of  $T$  and  $T'$ , written  $T + T'$ , is the transition system where

- $S'' = \{ns\} \cup S \cup S', ns \notin S \cup S'$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems such that  $S \cap S' = \emptyset$ . The *Non-deterministic choice* of  $T$  and  $T'$ , written  $T + T'$ , is the transition system where

- $S'' = \{ns\} \cup S \cup S', ns \notin S \cup S'$
- $A'' = A \cup A'$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems such that  $S \cap S' = \emptyset$ . The *Non-deterministic choice* of  $T$  and  $T'$ , written  $T + T'$ , is the transition system where

- $S'' = \{ns\} \cup S \cup S', ns \notin S \cup S'$
- $A'' = A \cup A'$
- $\rightarrow'' = \rightarrow \cup \rightarrow' \cup \rightarrow^n$  where  $ns \xrightarrow{a} s$  iff  $s_0 \xrightarrow{a} s$  or  $s'_0 \xrightarrow{a'} s$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems such that  $S \cap S' = \emptyset$ . The *Non-deterministic choice* of  $T$  and  $T'$ , written  $T + T'$ , is the transition system where

- $S'' = \{ns\} \cup S \cup S', ns \notin S \cup S'$
- $A'' = A \cup A'$
- $\rightarrow'' = \rightarrow \cup \rightarrow' \cup \rightarrow^n$  where  $ns \xrightarrow{a} s$  iff  $s_0 \xrightarrow{a} s$  or  $s'_0 \xrightarrow{a'} s$
- $\downarrow'' = \downarrow \cup \downarrow' \cup \{ns \mid s_0 \downarrow \vee s'_0 \downarrow'\}$

## Definition

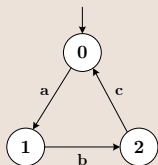
Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems such that  $S \cap S' = \emptyset$ . The *Non-deterministic choice* of  $T$  and  $T'$ , written  $T + T'$ , is the transition system where

- $S'' = \{ns\} \cup S \cup S', ns \notin S \cup S'$
- $A'' = A \cup A'$
- $\rightarrow'' = \rightarrow \cup \rightarrow' \cup \rightarrow^n$  where  $ns \xrightarrow{a} s$  iff  $s_0 \xrightarrow{a} s$  or  $s'_0 \xrightarrow{a'} s$
- $\downarrow'' = \downarrow \cup \downarrow' \cup \{ns \mid s_0 \downarrow \vee s'_0 \downarrow'\}$
- $s''_0 = ns$

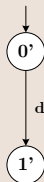
# Exercise

Find the alternative composition (non-deterministic choice) of the following transition diagrams:

1)



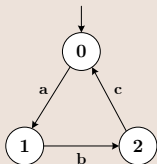
2)



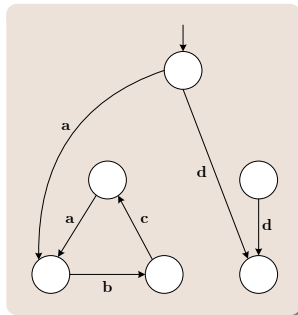
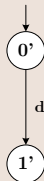
# Exercise

Find the alternative composition (non-deterministic choice) of the following transition diagrams:

1)



2)



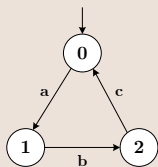
## Identification

Identification of initial states does not work in general

## Identification

Identification of initial states does not work in general

1)



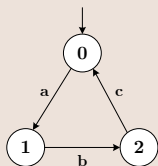
2)



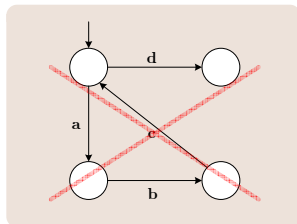
## Identification

Identification of initial states does not work in general

1)



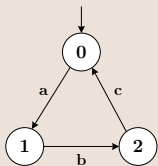
2)



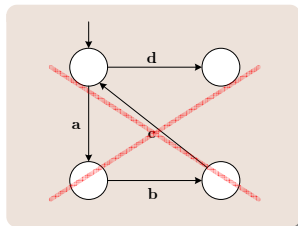
## Identification

Identification of initial states does not work in general

1)



2)



## Conclusion

Identification of initial states only works if both initial states do not have any incoming transitions.

## Idea

$T \parallel_{\gamma} T'$ :  $T$  and  $T'$  behave concurrently; they may talk to each other according to a communication scheme  $\gamma$ .

## Idea

$T \parallel_{\gamma} T'$ :  $T$  and  $T'$  behave concurrently; they may talk to each other according to a communication scheme  $\gamma$ .

## Communication Scheme

Let  $A$  be a set of actions. A (partial) *communication function* on  $A$  is a partial function  $\gamma : A \times A \rightarrow A$  that satisfies: for all  $a, b, c \in A$

- commutative:  $\gamma(a, b) = \gamma(b, a)$

## Idea

$T \parallel_{\gamma} T'$ :  $T$  and  $T'$  behave concurrently; they may talk to each other according to a communication scheme  $\gamma$ .

## Communication Scheme

Let  $A$  be a set of actions. A (partial) *communication function* on  $A$  is a partial function  $\gamma : A \times A \rightarrow A$  that satisfies: for all  $a, b, c \in A$

- commutative:  $\gamma(a, b) = \gamma(b, a)$
- associative:  $\gamma(a, (b, c)) = \gamma((a, b), c)$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems and let  $\gamma$  be a communication function. The *parallel composition* of  $T$  and  $T'$ , notation  $T \parallel_{\gamma} T'$ , is the transition system  $T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems and let  $\gamma$  be a communication function. The *parallel composition* of  $T$  and  $T'$ , notation  $T \parallel_{\gamma} T'$ , is the transition system  $T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

- $S'' = S \times S'$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems and let  $\gamma$  be a communication function. The *parallel composition* of  $T$  and  $T'$ , notation  $T \parallel_{\gamma} T'$ , is the transition system

$T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

- $S'' = S \times S'$
- $A'' = A \cup A' \cup \{\gamma(a, b) \mid a \in A \wedge b \in A'\}$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems and let  $\gamma$  be a communication function. The *parallel composition* of  $T$  and  $T'$ , notation  $T \parallel_{\gamma} T'$ , is the transition system

$T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

- $S'' = S \times S'$
- $A'' = A \cup A' \cup \{\gamma(a, b) \mid a \in A \wedge b \in A'\}$
- $(s_1, s'_1) \xrightarrow{c}'' (s_2, s'_2)$  iff

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems and let  $\gamma$  be a communication function. The *parallel composition* of  $T$  and  $T'$ , notation  $T \parallel_{\gamma} T'$ , is the transition system

$T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

- $S'' = S \times S'$
- $A'' = A \cup A' \cup \{\gamma(a, b) \mid a \in A \wedge b \in A'\}$
- $(s_1, s'_1) \xrightarrow{c}'' (s_2, s'_2)$  iff
  - $s_1 \xrightarrow{c} s_2$  and  $s'_1 = s'_2$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems and let  $\gamma$  be a communication function. The *parallel composition* of  $T$  and  $T'$ , notation  $T \parallel_{\gamma} T'$ , is the transition system  $T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

- $S'' = S \times S'$
- $A'' = A \cup A' \cup \{\gamma(a, b) \mid a \in A \wedge b \in A'\}$
- $(s_1, s'_1) \xrightarrow{c}'' (s_2, s'_2)$  iff
  - $s_1 \xrightarrow{c} s_2$  and  $s'_1 = s'_2$
  - $s'_1 \xrightarrow{c} s'_2$  and  $s_1 = s_2$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems and let  $\gamma$  be a communication function. The *parallel composition* of  $T$  and  $T'$ , notation  $T \parallel_{\gamma} T'$ , is the transition system  $T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

- $S'' = S \times S'$
- $A'' = A \cup A' \cup \{\gamma(a, b) \mid a \in A \wedge b \in A'\}$
- $(s_1, s'_1) \xrightarrow{c}'' (s_2, s'_2)$  iff
  - $s_1 \xrightarrow{c} s_2$  and  $s'_1 = s'_2$
  - $s'_1 \xrightarrow{c}' s'_2$  and  $s_1 = s_2$
  - $s_1 \xrightarrow{a} s_2$  and  $s'_1 \xrightarrow{b}' s'_2$  and  $\gamma(a, b) = c$

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems and let  $\gamma$  be a communication function. The *parallel composition* of  $T$  and  $T'$ , notation  $T \parallel_{\gamma} T'$ , is the transition system

$T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

- $S'' = S \times S'$
- $A'' = A \cup A' \cup \{\gamma(a, b) \mid a \in A \wedge b \in A'\}$
- $(s_1, s'_1) \xrightarrow{c}'' (s_2, s'_2)$  iff
  - $s_1 \xrightarrow{c} s_2$  and  $s'_1 = s'_2$
  - $s'_1 \xrightarrow{c} s'_2$  and  $s_1 = s_2$
  - $s_1 \xrightarrow{a} s_2$  and  $s'_1 \xrightarrow{b} s'_2$  and  $\gamma(a, b) = c$
- $s''_0 = (s_0, s'_0)$

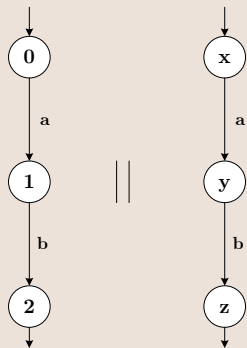
## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems and let  $\gamma$  be a communication function. The *parallel composition* of  $T$  and  $T'$ , notation  $T \parallel_{\gamma} T'$ , is the transition system

$T'' = (S'', A'', \rightarrow'', \downarrow'', s''_0)$  where

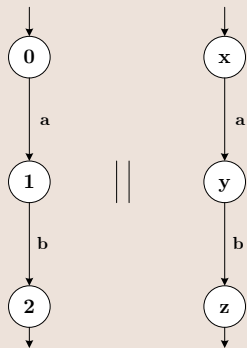
- $S'' = S \times S'$
- $A'' = A \cup A' \cup \{\gamma(a, b) \mid a \in A \wedge b \in A'\}$
- $(s_1, s'_1) \xrightarrow{c}'' (s_2, s'_2)$  iff
  - $s_1 \xrightarrow{c} s_2$  and  $s'_1 = s'_2$
  - $s'_1 \xrightarrow{c} s'_2$  and  $s_1 = s_2$
  - $s_1 \xrightarrow{a} s_2$  and  $s'_1 \xrightarrow{b} s'_2$  and  $\gamma(a, b) = c$
- $s''_0 = (s_0, s'_0)$
- $\downarrow'' = \downarrow \times \downarrow'$

# Exercise

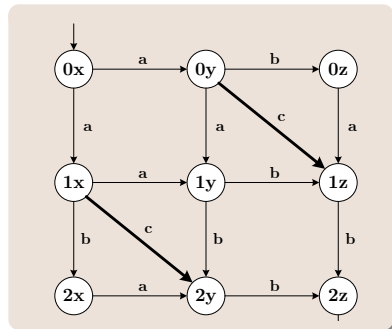


$$\gamma(a, b) = c$$

# Exercise



$$\gamma(a, b) = c$$



## Idea

Prevent a transition system from performing some transitions

# Encapsulation

## Idea

Prevent a transition system from performing some transitions

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system. Let  $H$  be a set of actions. The *encapsulation* of  $T$  w.r.t.  $H$ , notation  $\partial_H(T)$ , is the transition system  $(S, A, \rightarrow', \downarrow, s_0)$  where

- $s \xrightarrow{a}' s'$  iff  $s \xrightarrow{a} s'$  and  $a \notin H$ .

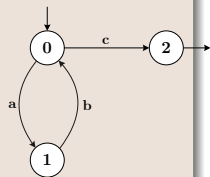
# Exercise

Let  $\gamma$  be a communication function such that  $\gamma(a, b) = c$  and  $\gamma$  is undefined otherwise. Compute the encapsulation of the parallel composition of these transition systems with respect to  $b$ .

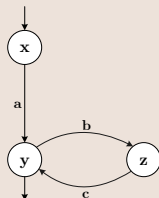
# Exercise

Let  $\gamma$  be a communication function such that  $\gamma(a, b) = c$  and  $\gamma$  is undefined otherwise. Compute the encapsulation of the parallel composition of these transition systems with respect to  $b$ .

1)



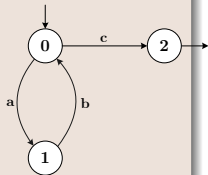
2)



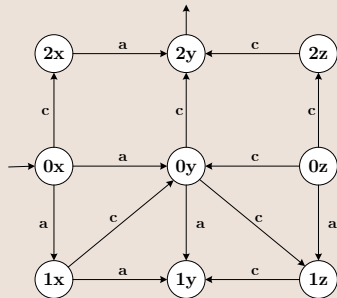
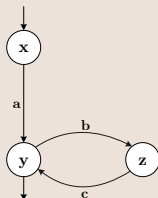
# Exercise

Let  $\gamma$  be a communication function such that  $\gamma(a, b) = c$  and  $\gamma$  is undefined otherwise. Compute the encapsulation of the parallel composition of these transition systems with respect to  $b$ .

1)



2)



## Idea

Hiding the internal actions from the outside world.

## Idea

Hiding the internal actions from the outside world.

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system and let  $I \subseteq A$ . The *abstraction* of  $T$  w.r.t.  $I$ , notation  $\tau_I(T)$ , is the transition system  $(S, A, \rightarrow', \downarrow, s_0)$  where

- if  $s_1 \xrightarrow{a} s_2$  and  $a \in I$ , then  $s_1 \xrightarrow{\tau} s_2$

## Idea

Hiding the internal actions from the outside world.

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  be a transition system and let  $I \subseteq A$ . The *abstraction* of  $T$  w.r.t.  $I$ , notation  $\tau_I(T)$ , is the transition system  $(S, A, \rightarrow', \downarrow, s_0)$  where

- if  $s_1 \xrightarrow{a} s_2$  and  $a \in I$ , then  $s_1 \xrightarrow{\tau} s_2$
- if  $s_1 \xrightarrow{a} s_2$  and  $a \notin I$ , then  $s_1 \xrightarrow{a} s_2$

- 1 Informal Description
- 2 Formal Description
- 3 Reviewing Some Concepts in Set Theory
- 4 Equivalence of Transition Systems (Part1)
  - Strong Isomorphism
  - Weak Isomorphism
  - Language Equivalence
  - Trace Equivalence
  - Trace Equivalence
  - Bisimulation
- 5 Composing Transition Systems
  - Sequential Composition
  - Non-deterministic Choice
  - Concurrency and Communication
  - Encapsulation
  - Abstraction
- 6 Equivalence of Transition Systems (Part2)
  - Branching Bisimulation
  - Weak Bisimulation

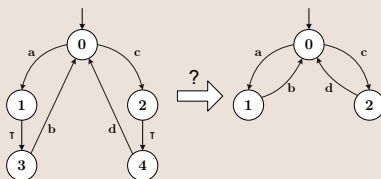
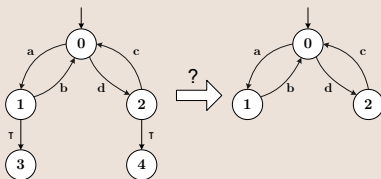
- $\tau$  actions are supposed to be *internal*, and thus not directly *observable*

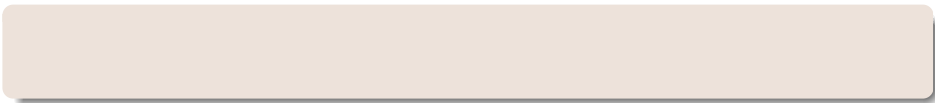
# Intuition

- $\tau$  actions are supposed to be *internal*, and thus not directly *observable*
- Execution of  $\tau$  actions can change the possibilities for executing observable actions (and termination)

# Intuition

- $\tau$  actions are supposed to be *internal*, and thus not directly *observable*
- Execution of  $\tau$  actions can change the possibilities for executing observable actions (and termination)





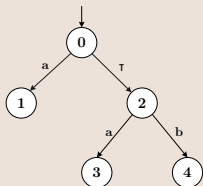
- Q: which  $\tau$ -transitions are “truly” silent?

# Silent Steps

- Q: which  $\tau$ -transitions are “truly” silent?
- A: those  $\tau$ -transitions that do not lose possible behaviors

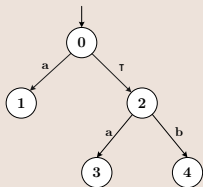
# Silent Steps

- Q: which  $\tau$ -transitions are “truly” silent?
- A: those  $\tau$ -transitions that do not lose possible behaviors



# Silent Steps

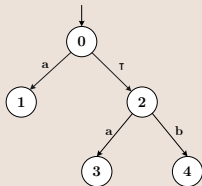
- Q: which  $\tau$ -transitions are “truly” silent?
- A: those  $\tau$ -transitions that do not lose possible behaviors



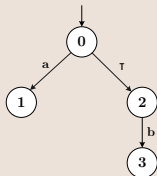
- $\tau$  is silent

# Silent Steps

- Q: which  $\tau$ -transitions are “truly” silent?
- A: those  $\tau$ -transitions that do not lose possible behaviors

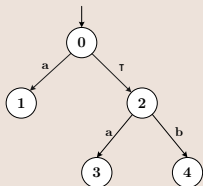


- $\tau$  is silent
- After executing  $\tau$  it is still possible to execute  $a$

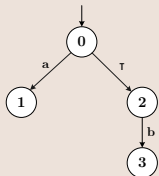


# Silent Steps

- Q: which  $\tau$ -transitions are “truly” silent?
- A: those  $\tau$ -transitions that do not lose possible behaviors



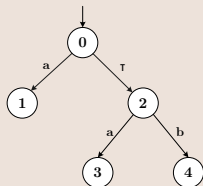
- $\tau$  is silent
- After executing  $\tau$  it is still possible to execute  $a$



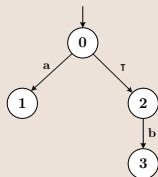
- $\tau$  is no longer silent

# Silent Steps

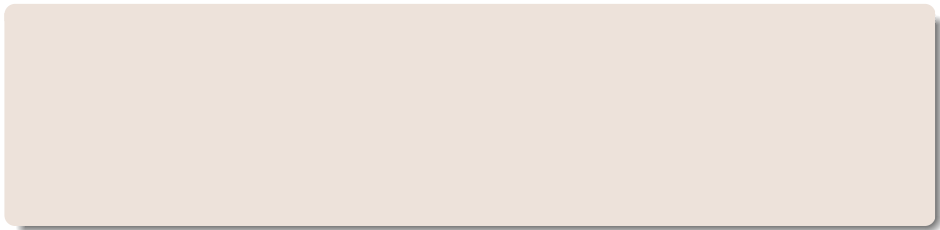
- Q: which  $\tau$ -transitions are “truly” silent?
- A: those  $\tau$ -transitions that do not lose possible behaviors



- $\tau$  is silent
- After executing  $\tau$  it is still possible to execute  $a$



- $\tau$  is no longer silent
- Execution of  $\tau$  means losing the possibility of  $a$



- The redundant  $\tau$  actions are usually called inert  $\tau$  actions

- The redundant  $\tau$  actions are usually called inert  $\tau$  actions
- There are many equivalences that try to remove inert  $\tau$  actions

- The redundant  $\tau$  actions are usually called inert  $\tau$  actions
- There are many equivalences that try to remove inert  $\tau$  actions
  - weak bisimulation(observation equivalence): coarsest

- The redundant  $\tau$  actions are usually called inert  $\tau$  actions
- There are many equivalences that try to remove inert  $\tau$  actions
  - weak bisimulation (observation equivalence): coarsest
  - branching bisimulation: finest

- The redundant  $\tau$  actions are usually called inert  $\tau$  actions
- There are many equivalences that try to remove inert  $\tau$  actions
  - weak bisimulation (observation equivalence): coarsest
  - branching bisimulation: finest
  - delay bisimulation

- The redundant  $\tau$  actions are usually called inert  $\tau$  actions
- There are many equivalences that try to remove inert  $\tau$  actions
  - weak bisimulation (observation equivalence): coarsest
  - branching bisimulation: finest
  - delay bisimulation
  - $\eta$ -bisimulation.

# Branching Bisimulation

## Definition

Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems. The *branching bisimulation*  $B$  between  $T$  and  $T'$ , notation  $T \stackrel{\leftrightarrow_b}{\sim} T'$  is a symmetric binary relation over  $B \subseteq S \times S'$  such that

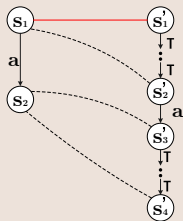
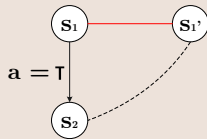
- $B(s_0, s'_0)$

# Branching Bisimulation

## Definition

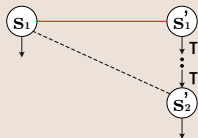
Let  $T = (S, A, \rightarrow, \downarrow, s_0)$  and  $T' = (S', A', \rightarrow', \downarrow', s'_0)$  be transition systems. The *branching bisimulation*  $B$  between  $T$  and  $T'$ , notation  $T \stackrel{b}{\leftrightarrow} T'$  is a symmetric binary relation over  $B \subseteq S \times S'$  such that

- $B(s_0, s'_0)$
- if  $B(s_1, s'_1)$  and  $s_1 \xrightarrow{a} s_2$  then
  - either  $a = \tau$  and  $B(s_2, s'_1)$
  - or there exists a path  $s'_1 \xrightarrow{\tau} \dots \xrightarrow{\tau} s'_2 \xrightarrow{a} s'_3 \xrightarrow{\tau} \dots \xrightarrow{\tau} s'_4$  such that  $B(s_1, s'_2)$ ,  $B(s_2, s'_3)$  and  $B(s_2, s'_4)$ .

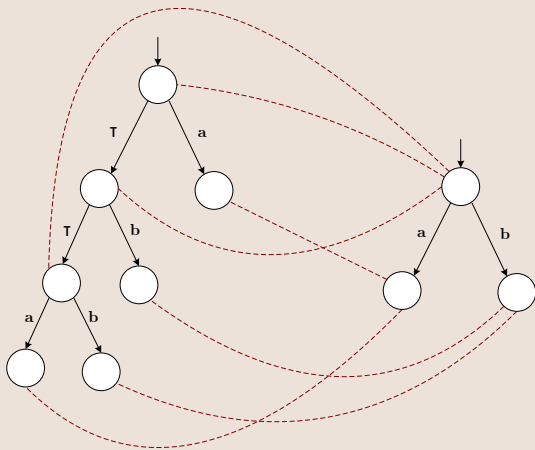


## Definition (Cont.)

- if  $B(s_1, s'_1)$  and  $s_1 \downarrow$ , then there is a path  $s'_1 \xrightarrow{\tau} \dots \xrightarrow{\tau} s'_2$  such that  $B(s_1, s'_2)$  and  $s'_2 \downarrow$ .

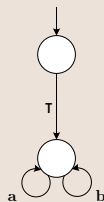
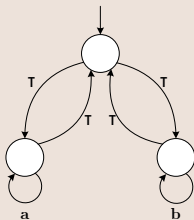


# Example



# Exercise

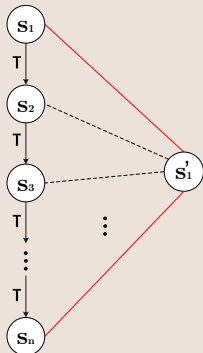
Are these transition diagrams branching bisimilar?



# Stuttering Lemma

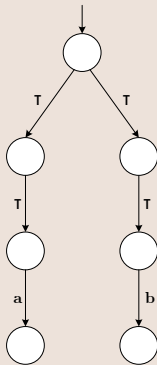
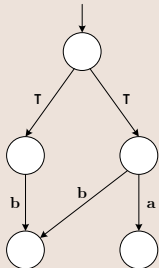
## Lemma

Let  $B$  be the branching bisimulation relation between  $T$  and  $T'$  and let for some  $n \geq 0$   $s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} \dots \xrightarrow{\tau} s_n$  be a path in  $T$ . If  $B(s_1, s'_1)$  and  $B(s_n, s'_1)$  then  $\forall 1 \leq i \leq n R(s_i, s'_1)$



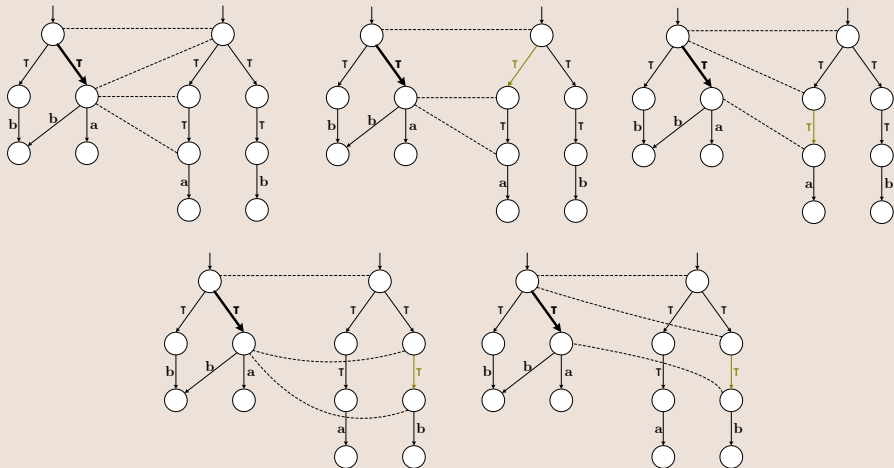
# Exercise

Determine whether these two transition systems are branching bisimilar or not.



# Solution

Consider the different ways that the bold  $\tau$ -action can be mimicked.



# Weak Bisimulation

- Weak bisimulation (observation equivalence) is less restrictive than branching bisimulation

# Weak Bisimulation

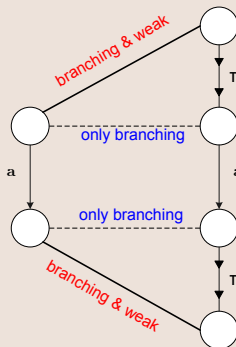
- Weak bisimulation (observation equivalence) is less restrictive than branching bisimulation
- It does not require the intermediate  $\tau$ -actions to be simulated

# Weak Bisimulation

- Weak bisimulation (observation equivalence) is less restrictive than branching bisimulation
- It does not require the intermediate  $\tau$ -actions to be simulated
- It does not preserve the branching structure of the transition system

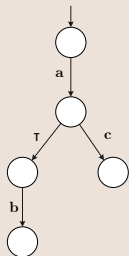
# Weak Bisimulation

- Weak bisimulation (observation equivalence) is less restrictive than branching bisimulation
- It does not require the intermediate  $\tau$ -actions to be simulated
- It does not preserve the branching structure of the transition system

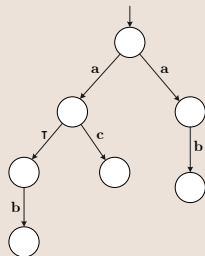


# Example

I)

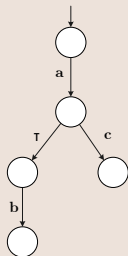


II)

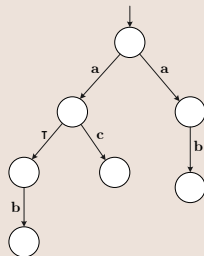


# Example

I)



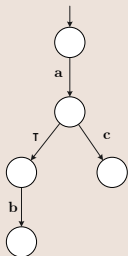
II)



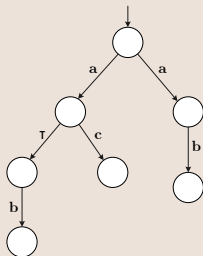
- In (I), always the choice between  $b$  and  $c$  is made after  $a$ -step

# Example

I)



II)



- In (I), always the choice between  $b$  and  $c$  is made after  $a$ -step
- In (II), there is a branch in which  $b$  is already chosen when the  $a$ -step occurs

Based on allowing for (stuttering)  $\tau$ -transition before and/or after an a-step:

Based on allowing for (stuttering)  $\tau$ -transition before and/or after an a-step:

- arbitrary before and arbitrary after: weak bisimulation (observation equivalence, the coarsest)

Based on allowing for (stuttering)  $\tau$ -transition before and/or after an a-step:

- arbitrary before and arbitrary after: weak bisimulation (observation equivalence, the coarsest)
- stuttering before and stuttering after: branching bisimulation (the finest)

Based on allowing for (stuttering)  $\tau$ -transition before and/or after an a-step:

- arbitrary before and arbitrary after: weak bisimulation (observation equivalence, the coarsest)
- stuttering before and stuttering after: branching bisimulation (the finest)
- arbitrary before and stuttering after: delay bisimulation

Based on allowing for (stuttering)  $\tau$ -transition before and/or after an a-step:

- arbitrary before and arbitrary after: weak bisimulation (observation equivalence, the coarsest)
- stuttering before and stuttering after: branching bisimulation (the finest)
- arbitrary before and stuttering after: delay bisimulation
- stuttering before and arbitrary after:  $\eta$ -bisimulation

$\eta$ - and delay bisimulation equivalence are two incomparable finer notions than weak bisimulation