ESE601: Hybrid Systems

Introduction to verification

Spring 2006

Suggested reading material

Papers (R14) - (R16) on the website.

The book “Model checking” by Clarke, Grumberg and Peled.

What is verification?

We need to make sure that the engineering systems we build are safe, functioning correctly, etc.

Systems can mean software, hardware, protocols, etc. Thus, not restricted to hybrid systems. In fact, verification originates in computer science, i.e. for discrete event systems.

How is verification done? The system is represented as transition system, the properties to be verified are represented as temporal logic formulas, whose truth values are to be determined/verified.

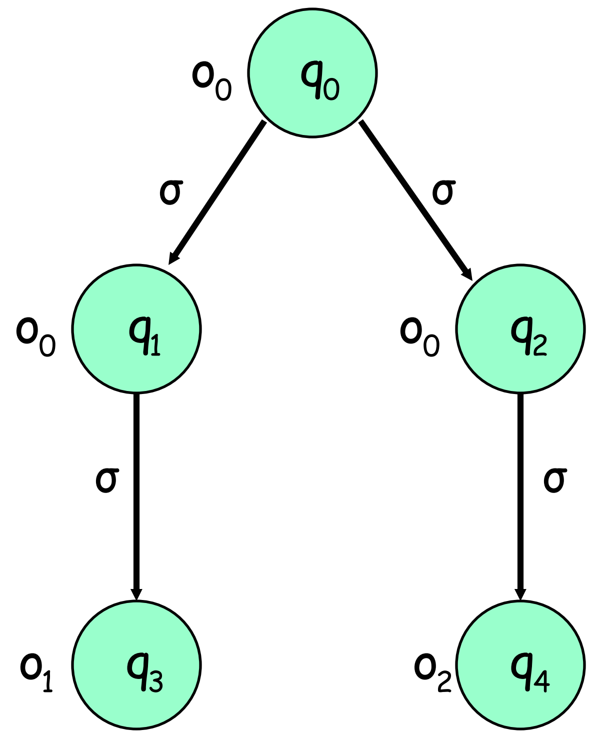
Transition Systems

A transition system

$$T = (Q, \Sigma, \rightarrow, O, \langle\cdot\rangle)$$

consists of

- A set of states Q
- A set of events Σ
- A set of observations O
- The transition relation $q_1 \xrightarrow{\sigma} q_2$
- The observation map $\langle q_1 \rangle = o_0$



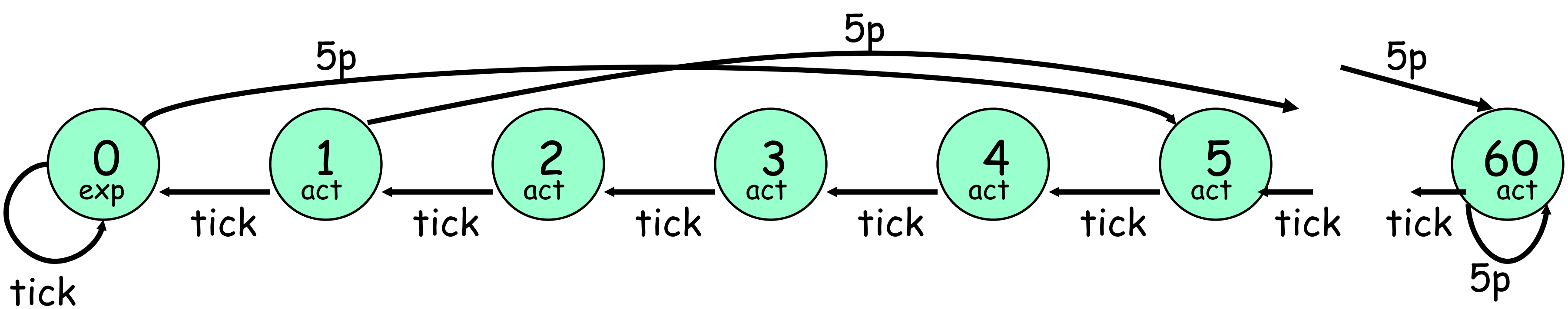
Initial may be incorporated

The sets Q, Σ, and O may be infinite

Language of T is all sequences of observations

A painful example

The parking meter



States Q ={0,1,2,…,60}

Events {tick,5p}

Observations {exp,act}

A possible string of observations (exp,act,act,act,act,act,exp,…)

Temporal logic (informal)

Temporal logic involves logical propositions whose truth values depend on time.

"Tomorrow is Thursday"

The time is related to the execution steps of the transition system.

The asserted property is related to the observation of the transition system.

"At the next state, the meter expires"

The basic verification problem

Given transition system T, and temporal logic formula φ

Basic verification problem

$$T \models \varphi$$

The transition system satisfies the formula if:
-All executions satisfy the formula (linear time)
-The initial states satisfy the formula (branching time)

Another verification problem

Given transition system T, and specification system S

Another verification problem

$$L(T) \subseteq L(S)$$

Language inclusion problems. Recall supervisory control problem.

Linear temporal logic

Linear temporal logic syntax

The LTL formulas are defined inductively as follows

Atomic propositions

All observation symbols p are formulas

Boolean operators

If φ_1 and φ_2 are formulas then

$$\varphi_1 \vee \varphi_2 \qquad \neg\varphi_1$$

Temporal operators

If φ_1 and φ_2 are formulas then

$$\varphi_1 \, U \, \varphi_2 \qquad \bigcirc \varphi_1$$

Linear temporal logic

LTL formulas are evaluated over (infinite) sequences of execution, which are called words.

Ex: w=(exp,act,act,act,act,act,exp,…)

$$(w,0)\models \mathrm{exp},\ (w,1)\models \mathrm{act},\ (w,1)\models \neg\mathrm{exp},\ \cdots$$

A word w satisfies a formula iff (w,0) satisfies it.

$$w\models \phi :\Leftrightarrow (w,0)\models \phi$$

$$w\models \bigcirc\, \phi :\Leftrightarrow (w,1)\models \phi$$

$$w\models \theta \;\; U \;\; \phi :\Leftrightarrow (w,i)\models \theta, (w,N)\models \phi, 0 \leq i < N.$$

Linear temporal logic

Express temporal specifications along sequences

Informally	Syntax	Semantics
Eventually p	$\Diamond p$	$* * * * * * * * p$
Always p	$\Box p$	$pppppppppppppppp$
If p then next q	$p \Rightarrow \bigcirc q$	$* * * * * * * pq$
p until q	$p\ U\ q$	$pppppppppppq * * *$

Penn University of Pennsylvania

Linear temporal logic

Syntactic boolean abbreviations

Conjunction	$\varphi_1 \wedge \varphi_2 = \neg(\neg\varphi_1 \vee \neg\varphi_2)$
Implication	$\varphi_1 \Rightarrow \varphi_2 = \neg\varphi_1 \vee \varphi_2$
Equivalence	$\varphi_1 \Leftrightarrow \varphi_2 = (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

Syntactic temporal abbreviations

Eventually	$\Diamond \varphi = \top\, U\, \varphi$
Always	$\Box \varphi = \neg\Diamond \neg\varphi$
In 3 steps	$\bigcirc_3\ \varphi = \bigcirc \bigcirc \bigcirc \varphi$

Penn
University of Pennsylvania

Linear temporal logic semantics

The LTL formulas are interpreted over infinite (omega) words

$$w = p_0\, p_1\, p_2\, p_3\, p_4 \ldots$$

$$(w,i) \models p \quad \text{iff} \quad p_i = p$$

$$(w,i) \models \varphi_1 \vee \varphi_2 \quad \text{iff} \quad (w,i) \models \varphi_1 \;\text{ or }\; (w,i) \models \varphi_2$$

$$(w,i) \models \neg\varphi_1 \;\text{ iff }\; (w,i) \not\models \varphi_1$$

$$(w,i) \models \bigcirc\, \varphi_1 \quad \text{iff} \quad (w,i+1) \models \varphi_1$$

$$(w,i) \models \varphi_1\, U\, \varphi_2$$

$$\exists j \geq i\ (w,j) \models \varphi_2 \quad \text{and} \quad \forall\, i \leq k \leq j \quad (w,k) \models \varphi_2$$

$$w \models \varphi \quad \text{iff} \quad (w,0) \models \varphi$$

$$T \models \varphi \quad \text{iff} \quad \forall w \in L(T) \;\; w \models \varphi$$

LTL examples

Two processors want to access a critical section. Each processor can has three observable states

p1={inCS, outCS, reqCS}

p2={inCS, outCS, reqCS}

Mutual exclusion

Both processors are not in the critical section at the same time.

$$\square \neg(p_1 = inCS \ \wedge \ p_2 = inCS)$$

Starvation freedom

If process 1 requests entry, then it eventually enters the critical section.

$$\square \ p_1 = reqCS \Rightarrow \Diamond p_1 = inCS$$

LTL Model Checking

Given transition system and LTL formula we have

LTL model checking

Determine if $T \models \varphi$

System verified

Counterexample

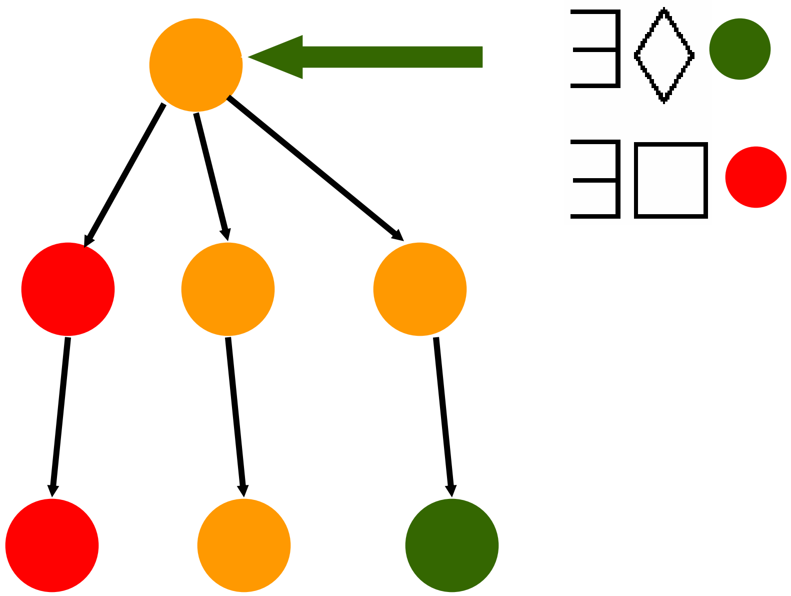
The transition system satisfies the formula if all executions satisfy it.
LTL model checking is decidable for finite T

Complexity : $$O(n+m)2^{O(k)}$$

states transitions formula length

Computational tree logic (CTL)

CTL is based on branching time. Its formulas are evaluated over the tree of trajectories generated from a given state of the transition system.



Computation tree logic

CTL syntax

The CTL formulas are defined inductively as follows

Atomic propositions

All observation symbols p are formulas

Boolean operators

If φ_1 and φ_2 are formulas then

$$\varphi_1 \vee \varphi_2 \qquad \neg\varphi_1$$

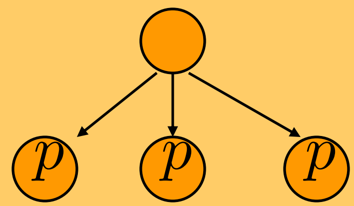
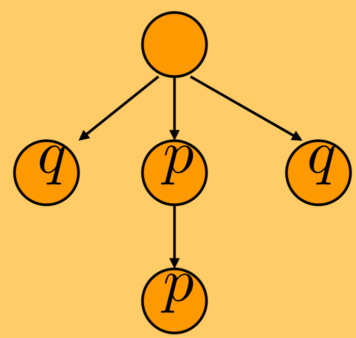
Temporal operators

If φ_1 and φ_2 are formulas then

$$\varphi_1 \exists U \varphi_2 \qquad \exists \bigcirc \varphi_1 \qquad \exists\Box\varphi_1$$

Computation tree logic (informally)

Express specifications in computation trees (branching time)

Informally	Syntax	Semantics
Inevitably next p	$\forall \bigcirc p$	
Possibly always p	$\exists \Box p$	

Penn
University of Pennsylvania

CTL Model Checking

Given transition system and CTL formula we have

CTL model checking

Determine if $T \models \varphi$

System verified

Counterexample

The transition system satisfies the formula if all initial states satisfy it.
CTL model checking is decidable for finite T

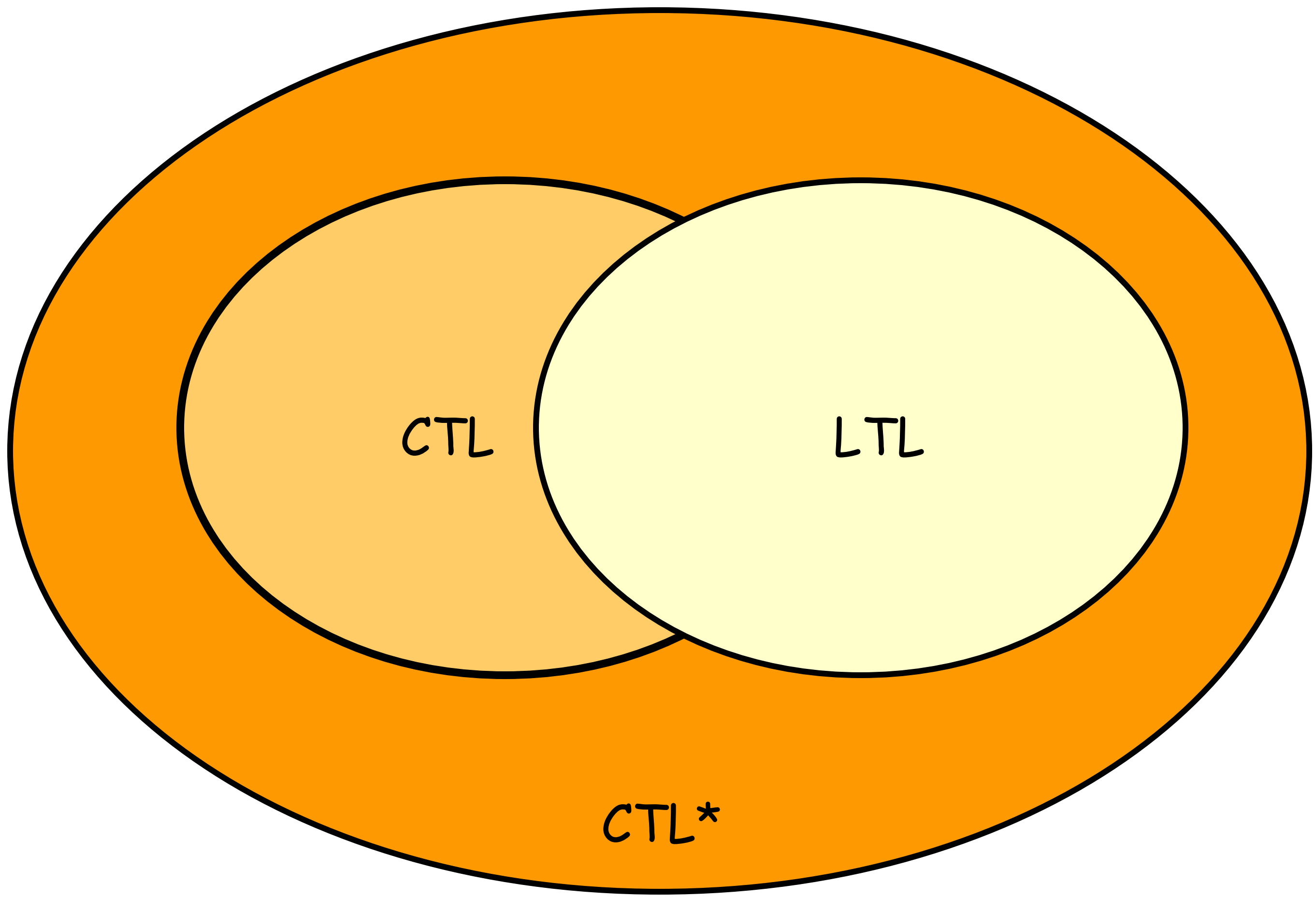
Complexity : $O((n+m)k)$

states transitions

formula length

Penn
UNIVERSITY of PENNSYLVANIA

Comparing logics



Dealing with complexity

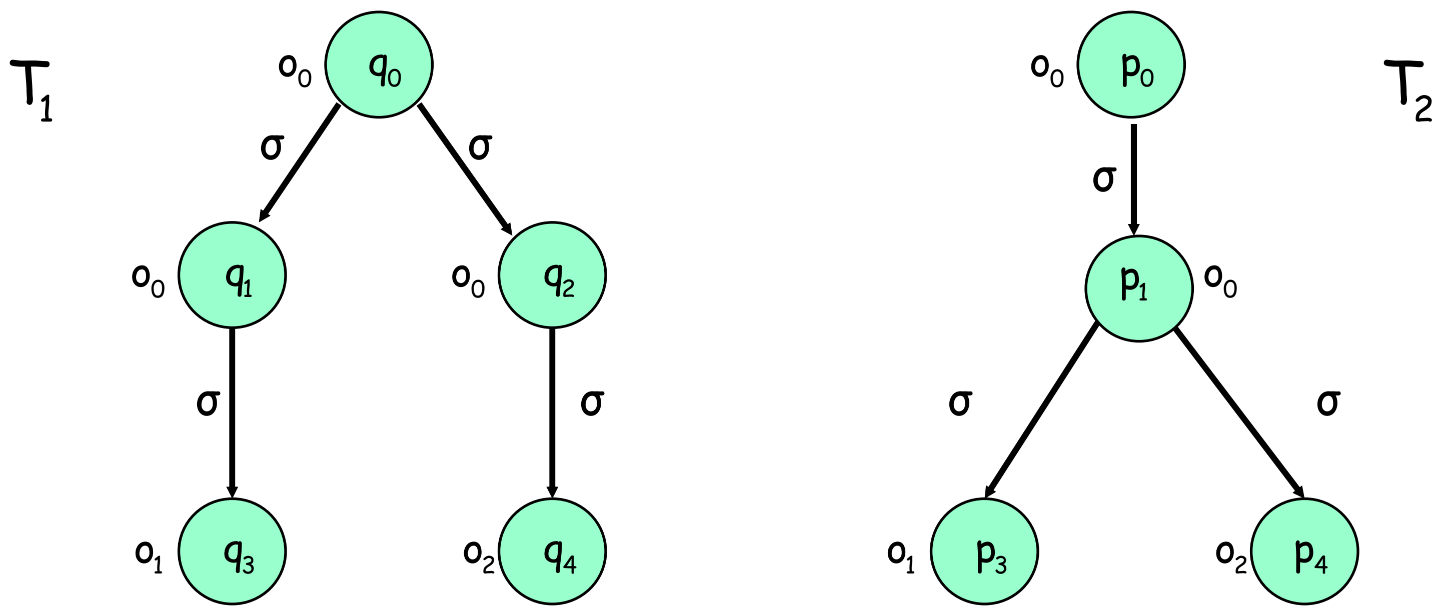
Bisimulation

Simulation

Language Inclusion

Language Equivalence

Consider two transition systems T_1 and T_2 over same Σ and O



Languanges are equivalent $L(T_1)=L(T_2)$

LTL equivalence

Consider two transition systems T_1 and T_2 and an LTL formula

Language equivalence

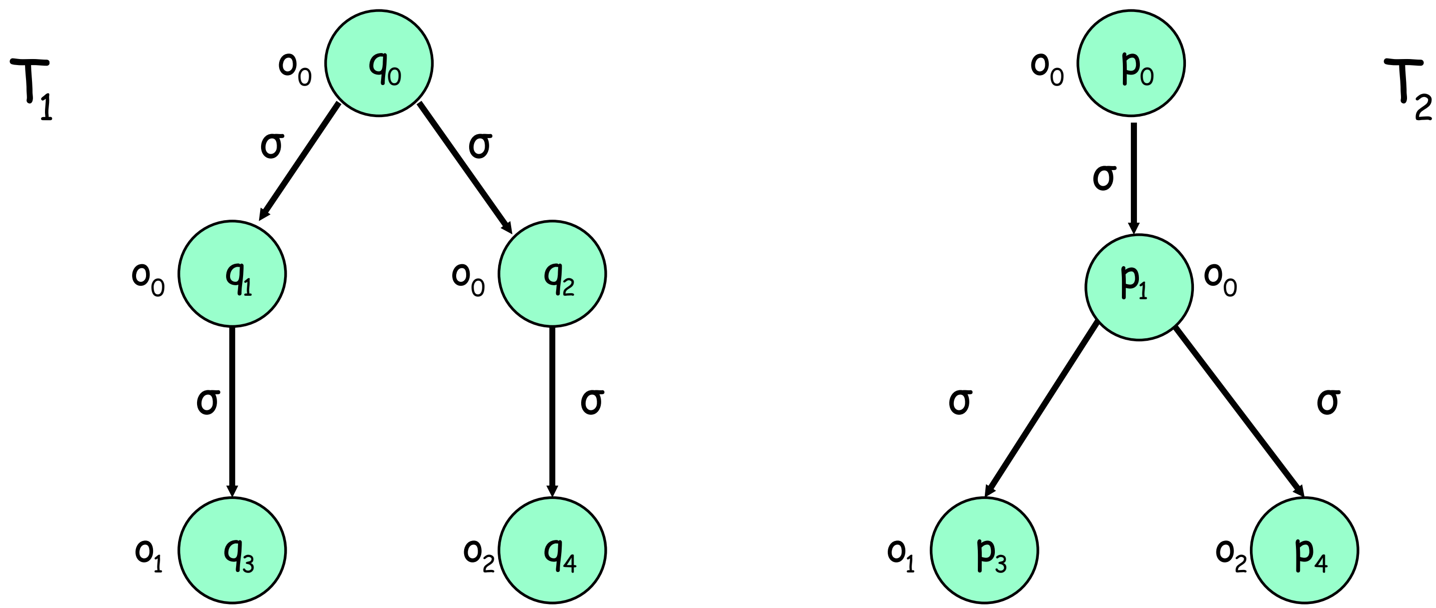
$$\text{If} \quad L(T_1) = L(T_2) \quad \text{then} \quad T_1 \models \varphi \;\Leftrightarrow T_2 \models \varphi$$

Language inclusion

$$\text{If} \quad L(T_1) \subseteq L(T_2) \quad \text{then} \quad T_2 \models \varphi \;\Rightarrow T_1 \models \varphi$$

Language equivalence and inclusion are difficult to check

Language Equivalence $\not\Rightarrow$ CTL equivalence



false

$$\forall \bigcirc \exists \Diamond o_1$$

true

Simulation Relations

Consider two transition systems

$$T_1 = (Q_1, \Sigma, \rightarrow_1, O, \langle \cdot \rangle_1)$$
$$T_2 = (Q_2, \Sigma, \rightarrow_2, O, \langle \cdot \rangle_2)$$

over the same set of labels and observations. A relation $S \subseteq Q_1 \times Q_2$ is called a simulation relation (of T_1 by T_2) if it

1. Respects observations

$$\text{if } (q,p) \in S \ \text{ then } \langle q \rangle_1 = \langle p \rangle_2$$

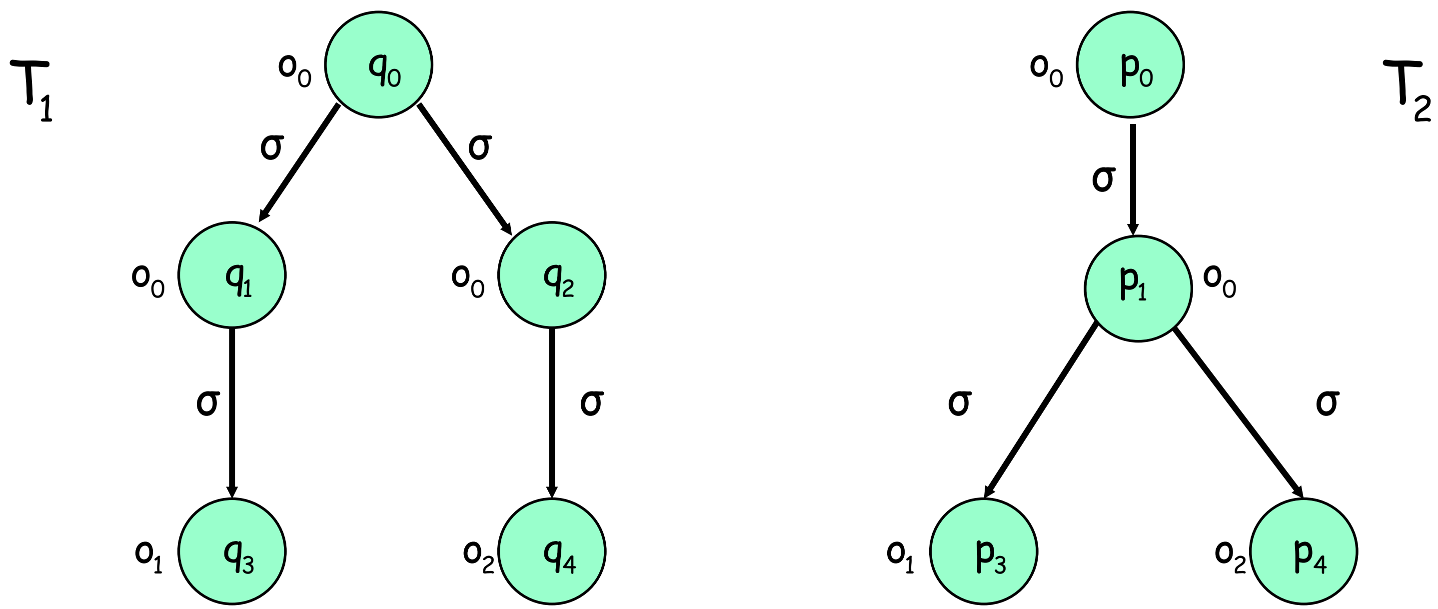
2. Respects transitions

$$\text{if } (q,p) \in S \ \text{ and } \ q \xrightarrow{\sigma} q', \ \text{ then } p \xrightarrow{\sigma} p' \ \text{ for some } (q',p') \in S$$

If a simulation relation exists, then $T_1 \leq T_2$

Game theoretic semantics

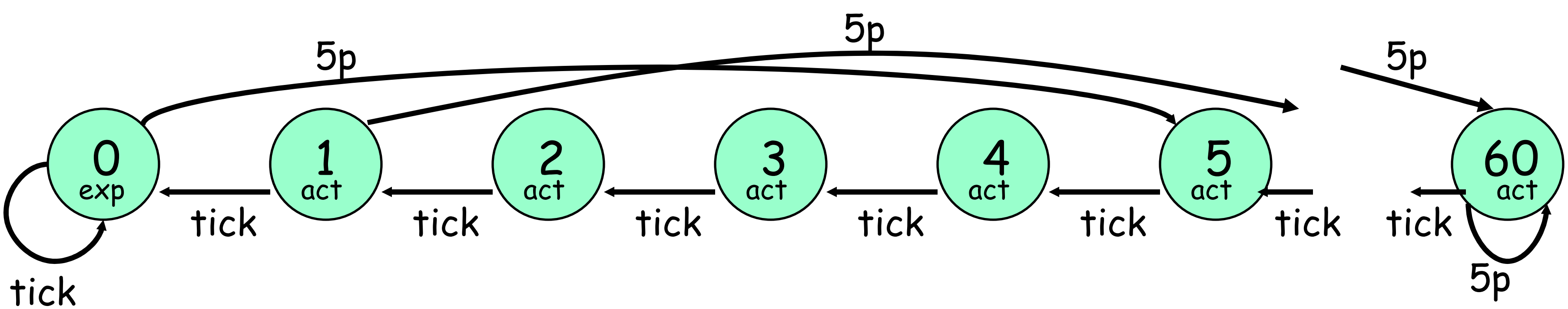
Simulation is a matching game between the systems



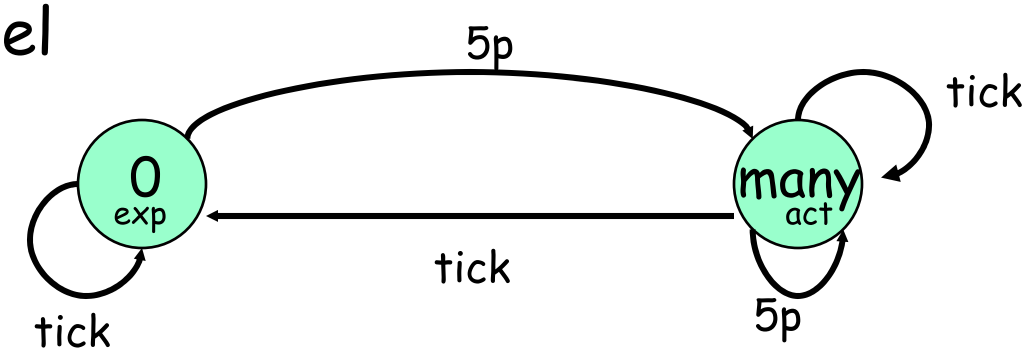
Check that $T_1 \leq T_2$ but it is not true that $T_2 \leq T_1$

The parking example

The parking meter



A coarser model



$$S = \{(0,0),(1,many),\ldots,(60,many)\}$$

Simulation relations

Consider two transition systems T_1 and T_2

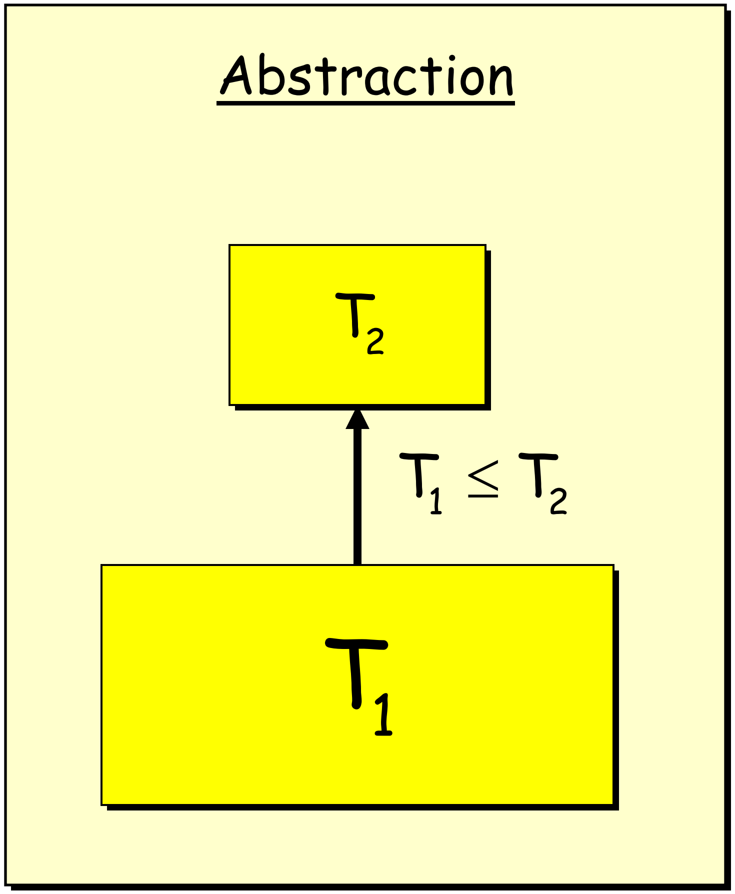
Simulation implies language inclusion

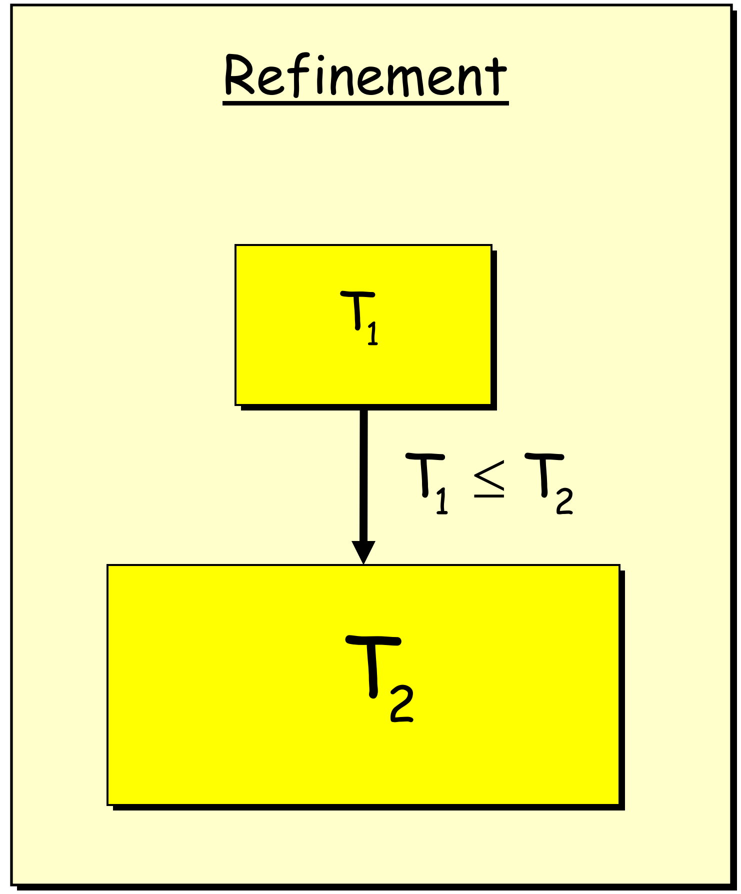
$$\text{If } \; T_1 \leq T_2 \; \text{ then } \; L(T_1) \subseteq L(T_2)$$

Complexity of $L(T_1) \subseteq L(T_2)$ $\quad O((n_1 + m_1)2^{n_2})$

Complexity of $T_1 \leq T_2$ $\quad O((n_1 + m_1)(n_2 + m_2))$

Two important cases





Penn
University of Pennsylvania

Bisimulation Relations

Consider two transition systems

$$T_1 = (Q_1, \Sigma, \rightarrow_1, O, \langle\cdot\rangle_1)$$
$$T_2 = (Q_2, \Sigma, \rightarrow_2, O, \langle\cdot\rangle_2)$$

over the same set of labels and observations. A relation $S \subseteq Q_1 \times Q_2$ is called a bisimulation relation if it

1. Respects observations

$$\text{if } (q,p) \in S \text{ then } \langle q\rangle_1 = \langle p\rangle_2$$

2. Respects transitions

$$\text{if } (q,p) \in S \text{ and } q \xrightarrow{\sigma} q', \text{ then } p \xrightarrow{\sigma} p' \text{ for some } (q',p') \in S$$

$$\text{if } (q,p) \in S \text{ and } p \xrightarrow{\sigma} p', \text{ then } q \xrightarrow{\sigma} q' \text{ for some } (q',p') \in S$$

If a simulation relation exists, then $T_1 \equiv T_2$

Penn
University of Pennsylvania

Bisimulation

Consider two transition systems T_1 and T_2

Bisimulation

$$T_1 \equiv T_2 \ \text{ if } \ T_1 \leq T_2 \ \wedge \ T_2 \leq T_1$$

Bisimulation is a symmetric simulation

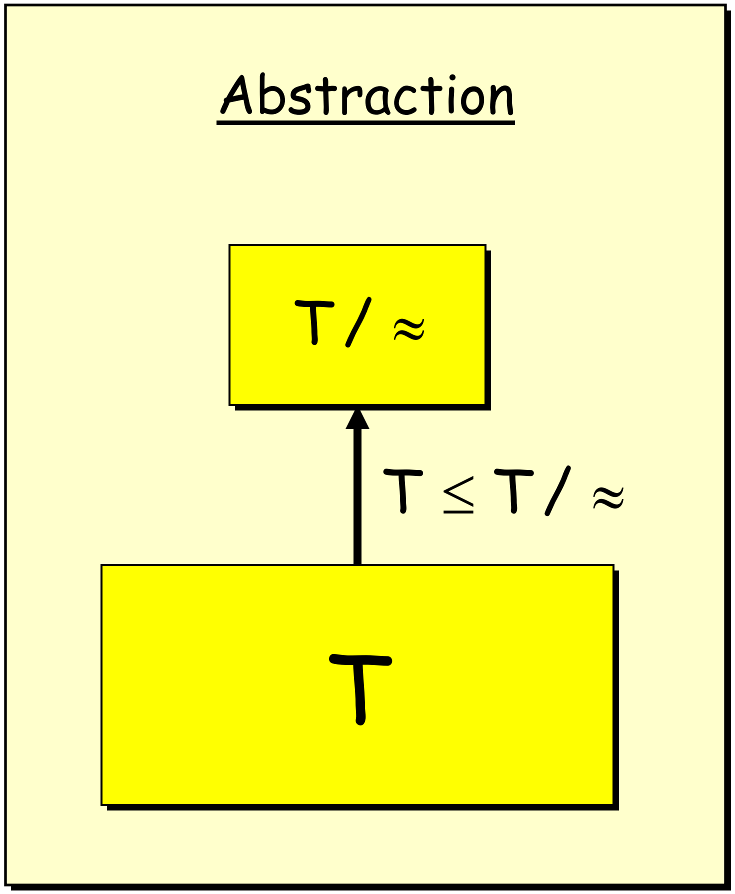
Strong notion of equivalence for transition systems

CTL* (and LTL) equivalence

$$\text{If } T_1 \equiv T_2 \ \text{ then } \ T_1 \models \varphi \ \Leftrightarrow T_2 \models \varphi$$

$$\text{If } T_1 \equiv T_2 \ \text{ then } \ L(T_1) = L(T_2)$$

Special quotients



When is the quotient language equivalent or bisimilar to T ?

Quotient Transition Systems

Given a transition system

$$T = (Q, \Sigma, \rightarrow, O, \langle\cdot\rangle)$$

and an observation preserving partition $\approx \subseteq Q \times Q$, define

$$T/\approx = (Q/\approx, \Sigma, \rightarrow_{\approx}, O, \langle\cdot\rangle_{\approx})$$

naturally using

1. Observation Map

$$\langle P \rangle_{\approx} = o \text{ iff there exists } p \in P \text{ with } \langle p \rangle = o$$

2. Transition Relation

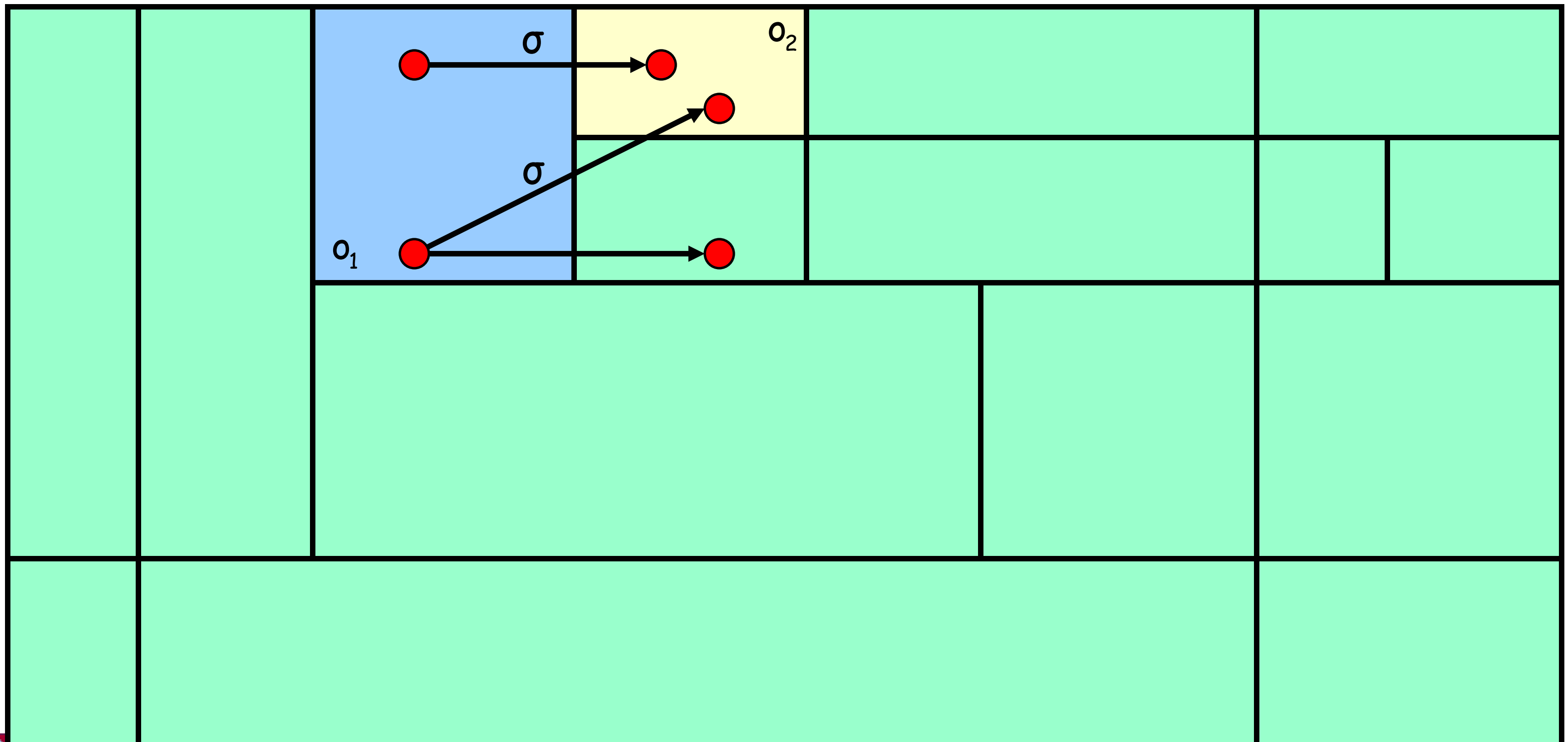
$$P \overset{\sigma}{\rightarrow}_{\approx} P' \text{ iff there exists } p \in P, p' \in P' \text{ with } p \overset{\sigma}{\rightarrow} p'$$

Penn
University of Pennsylvania

Bisimulation Algorithm

Quotient system $T/\approx$ always simulates the original system T

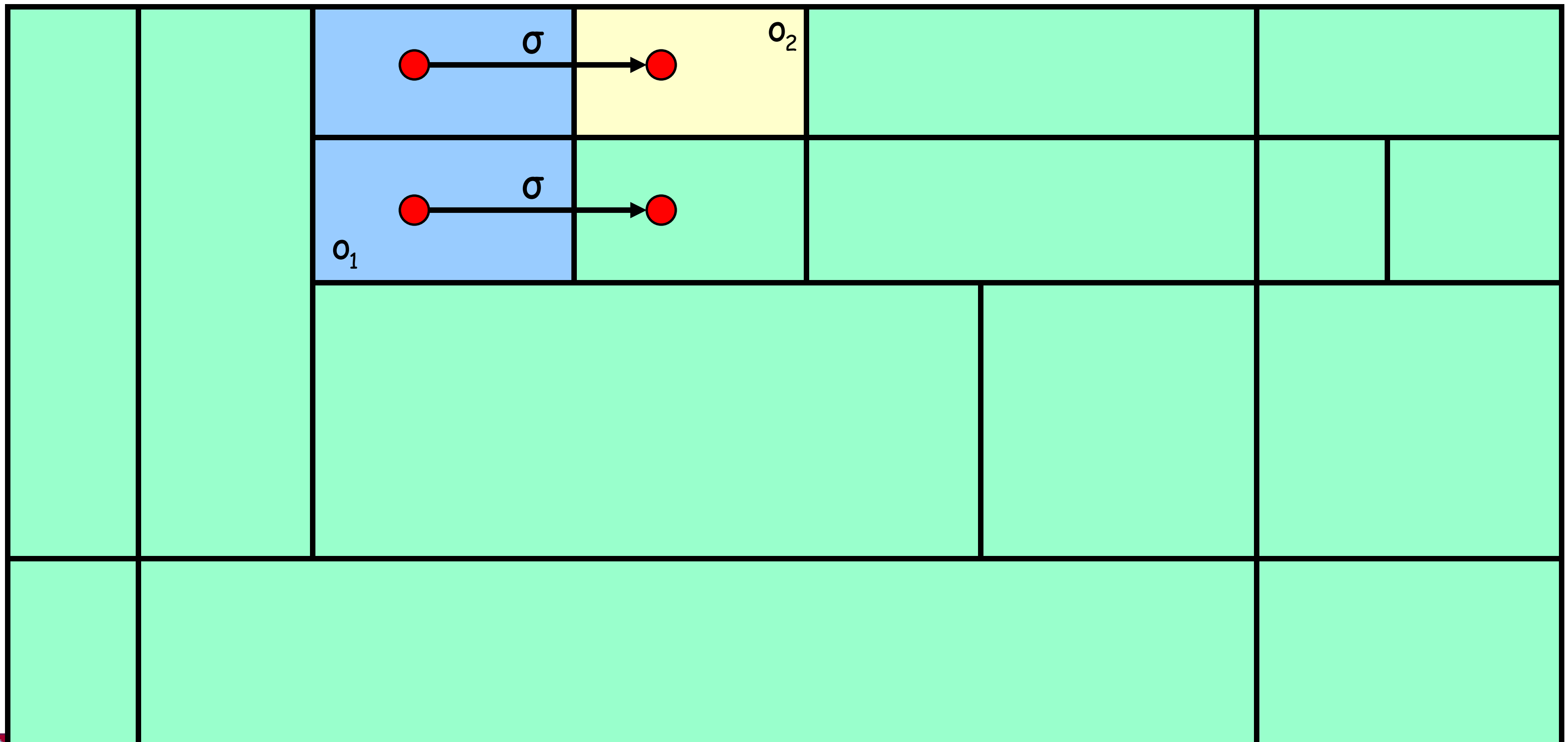
When does original system T simulate the quotient system $T/\approx$?



Bisimulation Algorithm

Quotient system $T/\approx$ always simulates the original system T

When does original system T simulate the quotient system $T/\approx$?



Transition Systems

A region is a subset of states $P \subseteq Q$

We define the following operators

$$Pre_\sigma(P) = \{q \in Q \mid \exists p \in P \quad q \xrightarrow{\sigma} p\}$$

$$Pre(P) = \{q \in Q \mid \exists \sigma \in \Sigma \quad \exists p \in P \quad q \xrightarrow{\sigma} p\}$$

$$Post_\sigma(P) = \{q \in Q \mid \exists p \in P \quad p \xrightarrow{\sigma} q\}$$

$$Post(P) = \{q \in Q \mid \exists \sigma \in \Sigma \quad \exists p \in P \quad p \xrightarrow{\sigma} q\}$$

Penn
University of Pennsylvania

Bisimulation algorithm

Bisimulation Algorithm

initialize $Q/_{\sim} = \{p \sim q \ \text{ iff } \ <q> = <p>\}$
while $\exists P, P' \in Q/_{\sim}$ **such that** $\emptyset \neq \subseteq P \cap Pre(P') \neq \subseteq P$

$$P_1 := P \cap Pre(P')$$

$$P_2 := P \setminus Pre(P')$$

$$Q/_{\sim} := (Q/_{\sim} \setminus \{P\}) \cup \{P_1, P_2\}$$

end while

If T is finite, then algorithm computes coarsest quotient.

If T is infinite, there is no guarantee of termination

Relationships

Bisimulation

Strongest, more properties, easiest to check

Simulation

Weaker, less properties, easy to check

Language Inclusion

Weakest, less properties, difficult to check

Complexity comparisons

Bisimulation

$$O(m \cdot \log(n))$$

↓

Simulation

$$O(m \cdot n)$$

↓

Language Equivalence

$$O(m \cdot 2^n)$$