

# CS 735: Formal Models for Concurrent and Asynchronous Systems

– Introduction

Instructor: Akshay S.

Jul 30, 2019

Course hours: Slot14,  
Tuesdays and Fridays 5:30-7:00pm  
Office hours: To be announced

# CS 735: Formal Models for Concurrent and Asynchronous Systems

– Introduction

Instructor: Akshay S.  
& and Prof. Krishna (for one module)

Jul 30, 2019

Course hours: Slot14,  
Tuesdays and Fridays 5:30-7:00pm  
Office hours: To be announced

# CS 735: Formal Models for Concurrent and Asynchronous Systems

– Introduction

Instructor: Akshay S.  
& and Prof. Krishna (for one module)

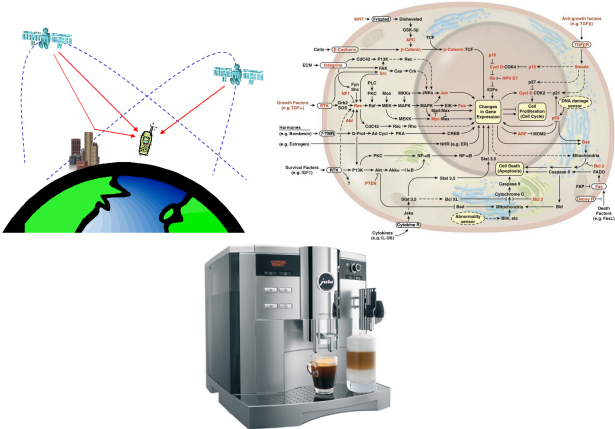
Jul 30, 2019

Course hours: Slot14,  
Tuesdays and Fridays 5:30-7:00pm  
Office hours: To be announced

Queries: Email me with [CS735-2019] in subject line  
[akshayss@cse.iitb.ac.in](mailto:akshayss@cse.iitb.ac.in)

# Goal

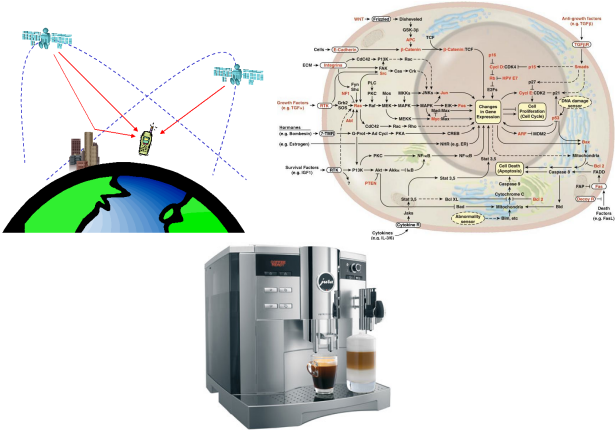
## Formal Models for distributed and infinite-state systems



# Goal

## Formal Models for distributed and infinite-state systems

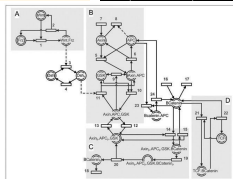
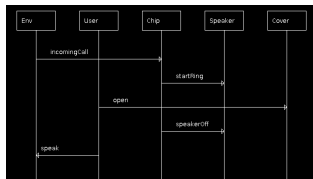
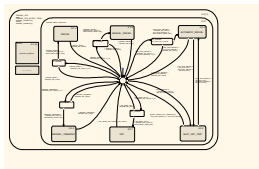
- **Distributed:** Concurrent, asynchronous, communicating,...



# Goal

## Formal Models for distributed and infinite-state systems

- ▶ **Distributed:** Concurrent, asynchronous, communicating,...
- ▶ **Formal models:** Mathematical description, graphical notations, Automata models



# Goal

## Formal Models for distributed and infinite-state systems

- ▶ **Distributed**: Concurrent, asynchronous, communicating,...
- ▶ **Formal models**: Mathematical description, graphical notations, Automata models
- ▶ **Infinite-state**: variables over an infinite domain: counters, channel/queue size, data, time, probabilities

# Goal

## Formal Models for distributed and infinite-state systems

- ▶ **Distributed**: Concurrent, asynchronous, communicating,...
- ▶ **Formal models**: Mathematical description, graphical notations, Automata models
- ▶ **Infinite-state**: variables over an infinite domain: counters, channel/queue size, data, time, probabilities

## Questions that we will tackle

- ▶ **Analysis** of such models
- ▶ **Characterization**, relations
- ▶ **Underlying** properties, generalizations



# Course contents

Topics and models that we will cover in this course:

1. Petri nets
2. Well-structured transition systems
3. Distributed automata models and their behaviors
4. Applications to concurrent programs

# Course contents

Topics and models that we will cover in this course:

## 1. Petri nets

- ▶ Elementary nets, Place/Transition nets
- ▶ Behaviors - traces, posets, unfoldings.
- ▶ Decision problems - reachability, coverability
- ▶ Tools, implementations and case-studies (optional)

## 2. Well-structured transition systems

## 3. Distributed automata models and their behaviors

## 4. Applications to concurrent programs

# Course contents

Topics and models that we will cover in this course:

## 1. Petri nets

- ▶ Elementary nets, Place/Transition nets
- ▶ Behaviors - traces, posets, unfoldings.
- ▶ Decision problems - reachability, coverability
- ▶ Tools, implementations and case-studies (optional)

## 2. Well-structured transition systems

- ▶ A generalized abstraction for infinite-state systems
- ▶ Well-quasi orders and well-founded systems
- ▶ Applications to show termination of infinite systems
- ▶ Theoretical bounds on complexity (optional)

## 3. Distributed automata models and their behaviors

## 4. Applications to concurrent programs

# Course contents

Topics and models that we will cover in this course:

## 1. Petri nets

- ▶ Elementary nets, Place/Transition nets
- ▶ Behaviors - traces, posets, unfoldings.
- ▶ Decision problems - reachability, coverability
- ▶ Tools, implementations and case-studies (optional)

## 2. Well-structured transition systems

- ▶ A generalized abstraction for infinite-state systems
- ▶ Well-quasi orders and well-founded systems
- ▶ Applications to show termination of infinite systems
- ▶ Theoretical bounds on complexity (optional)

## 3. Distributed automata models and their behaviors

- ▶ Asynchronous automata
- ▶ Message passing automata and Lossy channel machines

## 4. Applications to concurrent programs

# Course contents

Topics and models that we will cover in this course:

## 1. Petri nets

- ▶ Elementary nets, Place/Transition nets
- ▶ Behaviors - traces, posets, unfoldings.
- ▶ Decision problems - reachability, coverability
- ▶ Tools, implementations and case-studies (optional)

## 2. Well-structured transition systems

- ▶ A generalized abstraction for infinite-state systems
- ▶ Well-quasi orders and well-founded systems
- ▶ Applications to show termination of infinite systems
- ▶ Theoretical bounds on complexity (optional)

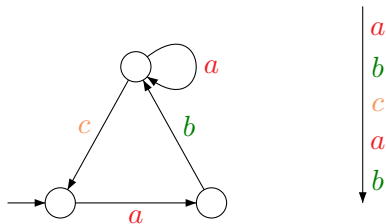
## 3. Distributed automata models and their behaviors

- ▶ Asynchronous automata
- ▶ Message passing automata and Lossy channel machines

## 4. Applications to concurrent programs

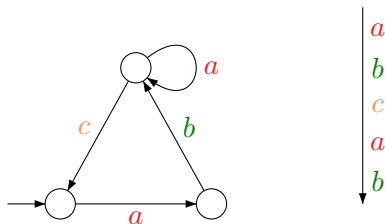
- ▶ Modeling “weak” memory in programs: variants, algorithms
- ▶ Towards efficiency: (Dynamic) partial order reductions

# Automata



- ▶ Behaviours are words, i.e., sequences of actions over a finite alphabet  $\Sigma = \{a, b, c\}$ .

# Automata

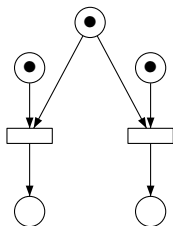


- ▶ Behaviours are words, i.e., sequences of actions over a finite alphabet  $\Sigma = \{a, b, c\}$ .

## Questions

- ▶ How shall we distribute it?
- ▶ How shall we add concurrent behaviors?

# Petri Nets



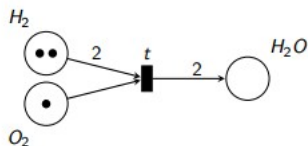
- ▶ An old model for distributed systems
  - ▶ invented by Carl Petri (-at the age of 13- in 1939? or '62)
  - ▶ to model resource consumption and so on...



## Examples of Petri nets

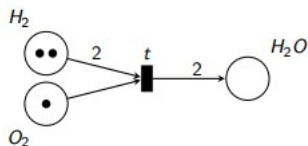
- ▶ A chemical reaction:  $2H_2 + O_2 \rightarrow 2H_2O$ .
- ▶ A library
- ▶ A producer-consumer example
- ▶ A coffee machine

## Examples of Petri nets



- ▶ A chemical reaction:  $2H_2 + O_2 \rightarrow 2H_2O$ .
- ▶ A library
- ▶ A producer-consumer example
- ▶ A coffee machine

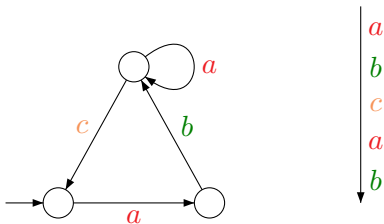
## Examples of Petri nets



- ▶ A chemical reaction:  $2H_2 + O_2 \rightarrow 2H_2O$ .
- ▶ A library
- ▶ A producer-consumer example
- ▶ A coffee machine

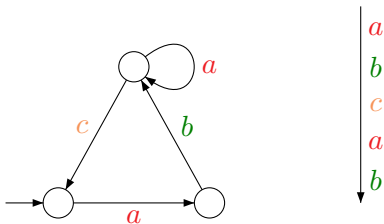
Applications: Business process models, stochastic processes, biological networks and so on

# Automata



- ▶ Behaviours are words, i.e., sequences of actions over a finite alphabet  $\Sigma = \{a, b, c\}$ .

# Automata

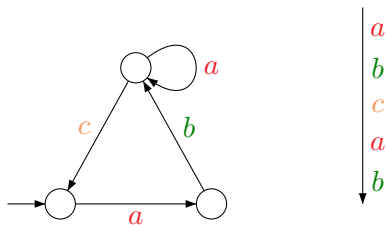


- ▶ Behaviours are words, i.e., sequences of actions over a finite alphabet  $\Sigma = \{a, b, c\}$ .

## Questions

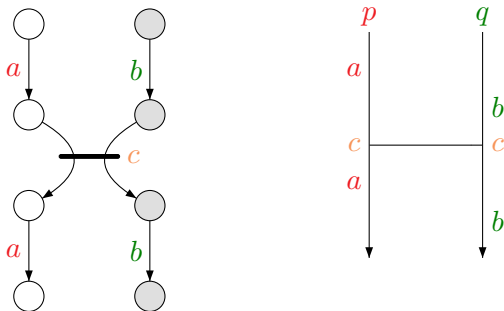
- ▶ How shall we distribute it?
- ▶ How shall we add concurrent behaviors?

# Asynchronous Automata



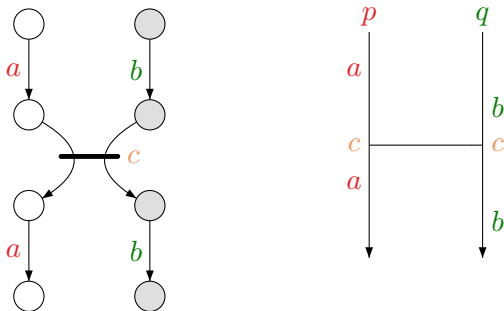
- Behaviours are words, i.e., sequences of actions over a finite alphabet  $\Sigma = \{a, b, c\}$ .

# Asynchronous Automata



- ▶ Actions are distributed across processes (with sharing!)
- ▶ Some actions are shared, e.g.,  $c$  is allowed only if both  $p$  and  $q$  move on  $c$ .

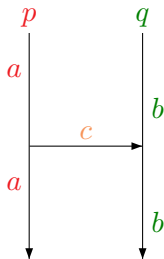
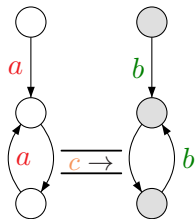
# Asynchronous Automata



- ▶ What are the properties of languages accepted by such automata? E.g. above accepts  $\{\underline{abcab}, bacab, bacba, abcba\}$ .
- ▶ Given a language  $L$ , (when) can it be accepted by such an asynchronous automaton?

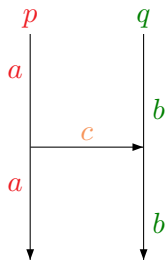
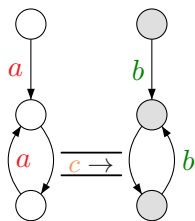


# Message Passing Automata



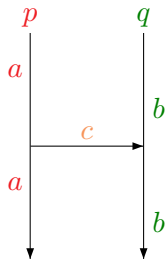
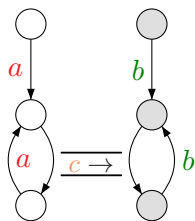
- ▶ In fact, this formalism is Turing powerful!
- ▶ We will consider decidability issues.

# Message Passing Automata



- ▶ In fact, this formalism is Turing powerful!
- ▶ We will consider decidability issues.
- ▶ (Surprising fact: If you are allowed to lose messages randomly then it is decidable!)

# Message Passing Automata



- ▶ In fact, this formalism is Turing powerful!
- ▶ We will consider decidability issues.
- ▶ (Surprising fact: If you are allowed to lose messages randomly then it is decidable!) These are called Lossy channel systems.

# Applications to Concurrent programs

What are good formal models for concurrent programs?

- ▶ Automata or transition systems

# Applications to Concurrent programs

What are good formal models for concurrent programs?

- ▶ Automata or transition systems
- ▶ Distributed/Asynchronous/Message-passing automata??
- ▶ Petri nets

# Applications to Concurrent programs

What are good formal models for concurrent programs?

- ▶ Automata or transition systems
- ▶ Distributed/Asynchronous/Message-passing automata??
- ▶ Petri nets

Does this capture reality of programs in today's world?

# Modeling concurrent programs

## Two issues

- ▶ In the multi-processor world: memory access is no longer atomic!

# Modeling concurrent programs

## Two issues

- ▶ In the multi-processor world: memory access is no longer atomic!
- ▶ There is no non-determinism! How to avoid exploring runs

Leads to:

1. Weak memory models
2. Partial order reduction techniques



## Generalizing...

- ▶ In general, these are examples of **infinite-state objects**.

## Generalizing...

- ▶ In general, these are examples of **infinite-state objects**.
- ▶ If you don't like “state objects”, think of them as **infinite discrete structures!**

## Generalizing...

- ▶ In general, these are examples of **infinite-state objects**.
- ▶ If you don't like “state objects”, think of them as **infinite discrete structures!**
- ▶ Why?

## Generalizing...

- ▶ In general, these are examples of **infinite-state objects**.
- ▶ If you don't like “state objects”, think of them as **infinite discrete structures!**
- ▶ Why?

Another title for this course:

**Reasoning about infinite (discrete) structures!**

## Generalizing...

- ▶ In general, these are examples of **infinite-state objects**.
- ▶ If you don't like “state objects”, think of them as **infinite discrete structures!**
- ▶ Why?

Another title for this course:

**Reasoning about infinite (discrete) structures!**

- ▶ Theory of well-structured transition systems
- ▶ Under-approximate verification
- ▶ Fixed-point approaches

## Generalizing...

- ▶ In general, these are examples of **infinite-state objects**.
- ▶ If you don't like “state objects”, think of them as **infinite discrete structures!**
- ▶ Why?

Another title for this course:

**Reasoning about infinite (discrete) structures!**

- ▶ Theory of well-structured transition systems
- ▶ Under-approximate verification
- ▶ Fixed-point approaches

## Pictures and Mathematics

- ▶ How do you write these objects mathematically?
- ▶ Why write them mathematically?

## Some take-aways from this course

- ▶ Different formal models for distributed systems
- ▶ Mathematical formalisms that reason about (the infinite) behaviors of such systems.
- ▶ Techniques to automatically analyze such systems.
- ▶ How to use them and where they are applied.

# Logistics

## Evaluation (flexible... upto a point)

- ▶ Continuous evaluation - assignments/quizzes : 40%
- ▶ Exam (Midsem/Endsem): 35 %
- ▶ Paper presentation: 25 %

There will be guest lectures, research directions given along the way.



# Logistics

## Evaluation (flexible... upto a point)

- ▶ Continuous evaluation - assignments/quizzes : 40%
- ▶ Exam (Midsem/Endsem): 35 %
- ▶ Paper presentation: 25 %

There will be guest lectures, research directions given along the way.

Course material, references will be posted at

- ▶ <http://www.cse.iitb.ac.in/~akshayss/teaching.html>
- ▶ Piazza will be set up soon?