COL869: Special Topics in Concurrency

– Introduction

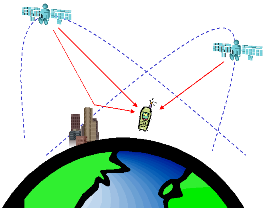
Instructors : S Akshay and Sanjiva Prasad

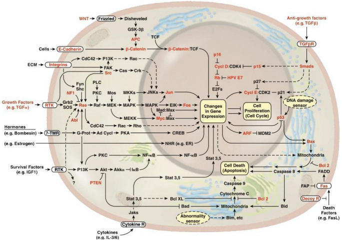
Jul 24, 2018

Course hours: Slot AD,
Tuesdays and Fridays 3:30-5:00pm

Goal

Formal Models for distributed and infinite-state systems

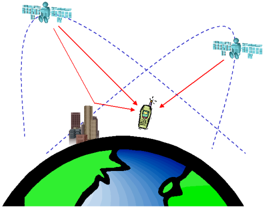


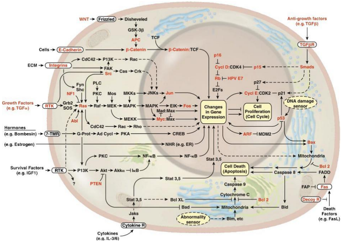


Goal

Formal Models for distributed and infinite-state systems

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

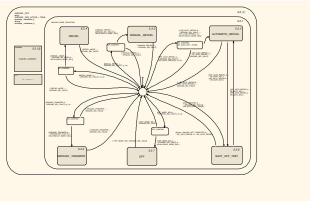


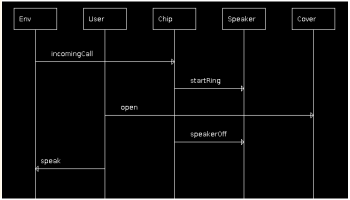


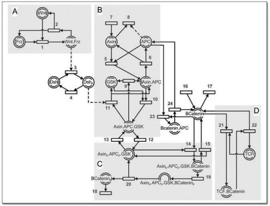
Goal

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

Practical significance

- Finding bugs in programs: Ariane V crash, Intel Pentium bug, Toyota recall...

Practical significance

- Finding bugs in programs: Ariane V crash, Intel Pentium bug, Toyota recall...
- Modeling systems: Networks, Large biological pathway systems

Practical significance

- Finding bugs in programs: Ariane V crash, Intel Pentium bug, Toyota recall...
- Modeling systems: Networks, Large biological pathway systems
- Pictorial representation of Spec: Telecom standards,

Practical significance

- Finding bugs in programs: Ariane V crash, Intel Pentium bug, Toyota recall...
- Modeling systems: Networks, Large biological pathway systems
- Pictorial representation of Spec: Telecom standards,

Application domains

- Hardware (and increasingly software) industry: Intel, Microsoft
- Business processes: web-services, Workflow models
- Safety-critical systems: nuclear reactors, airbus
- Security domain

Course contents

Topics and models that we will cover in this course:

1. Petri nets
2. Process Algebra
3. Automata over a distributed alphabet
4. Well-structured transition systems

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
2. Process Algebra
3. Automata over a distributed alphabet
4. Well-structured transition systems

Course contents

Topics and models that we will cover in this course:

1. Petri nets
2. Process Algebra
   - Communicating sequential processes (CSP)
   - Communicating concurrent systems (CCS)
   - Equivalences and bisimulations
3. Automata over a distributed alphabet
4. Well-structured transition systems

Course contents

Topics and models that we will cover in this course:

1. Petri nets
2. Process Algebra
3. Automata over a distributed alphabet
   - direct product and asynchronous products
   - communicating and message passing automata
4. Well-structured transition systems

Course contents

Topics and models that we will cover in this course:

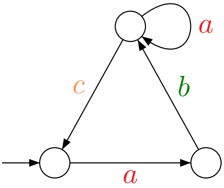
1. Petri nets
2. Process Algebra
3. Automata over a distributed alphabet
   - direct product and asynchronous products
   - communicating and message passing automata
     - Models for scenario-based specification SDL/UML-like spec - used in Telecom/other industries for specification.
     - Lossy channel machines
4. Well-structured transition systems

Course contents

Topics and models that we will cover in this course:

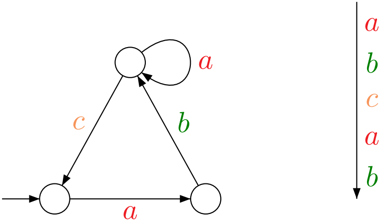
1. Petri nets
2. Process Algebra
3. Automata over a distributed alphabet
4. Well-structured transition systems
   - A generalized abstraction for infinite-state systems
   - Well-quasi orders and well-founded systems
   - Applications to show termination of infinite systems
   - Theoretical bounds on complexity

Transition systems



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

Transition systems

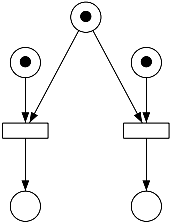


- Behaviours are words, i.e., sequences of actions over a finite alphabet $\Sigma = \{a, b, c\}$.

Questions

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

Petri Nets

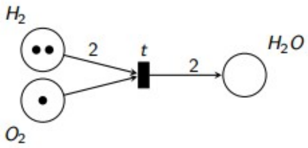


- An old model for distributed systems
  - invented by Carl Petri (-at the age of 13- in 1939? or '62)
  - to model resource consumption and so on...

Examples of Petri nets

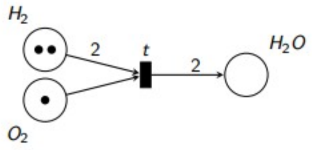
- A chemical reaction: $2H_2 + O_2 \to 2H_2O$.
- A library
- A producer-consumer example
- A coffee machine

Examples of Petri nets



- A chemical reaction: $2H_2 + O_2 \to 2H_2O$.
- 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

Many many extensions: with time, data, colors!

An algebraic theory of processes

- Can we model how processes interact as algebraic operations?
- Think of regular expressions!

An algebraic theory of processes

- Can we model how processes interact as algebraic operations?
- Think of regular expressions!
- Can you sequentially or parallely compose processes?

An algebraic theory of processes

- Can we model how processes interact as algebraic operations?
- Think of regular expressions!
- Can you sequentially or parallely compose processes?
- Fix few symbols and define operations to build/model complicated structures!

An algebraic theory of processes

- Can we model how processes interact as algebraic operations?
- Think of regular expressions!
- Can you sequentially or parallely compose processes?
- Fix few symbols and define operations to build/model complicated structures!

$$CM := coin.(\overline{coffee}.CM + \overline{tea}.CM)$$

An algebraic theory of processes

- Can we model how processes interact as algebraic operations?
- Think of regular expressions!
- Can you sequentially or parallely compose processes?
- Fix few symbols and define operations to build/model complicated structures!

$$CM := coin.(\overline{coffee}.CM + \overline{tea}.CM)$$

- Use this to study equivalence and difference between systems!

An algebraic theory of processes

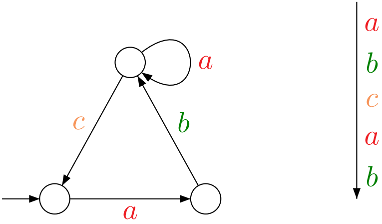
- Can we model how processes interact as algebraic operations?
- Think of regular expressions!
- Can you sequentially or parallely compose processes?
- Fix few symbols and define operations to build/model complicated structures!

$$CM := coin.(\overline{coffee}.CM + \overline{tea}.CM)$$

- Use this to study equivalence and difference between systems!

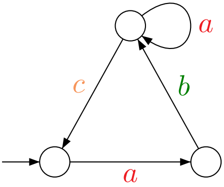
$$CS := coffee.\overline{pub.CS}$$

Transition systems and automata



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

Transition systems and automata

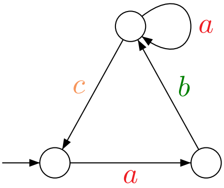


- Behaviours are words, i.e., sequences of actions over a finite alphabet $\Sigma = \{a, b, c\}$.

Questions

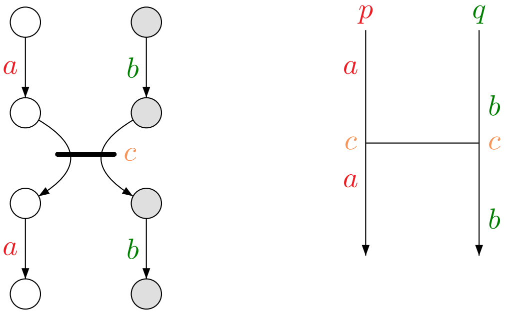
- How shall we distribute it?
- How shall we add concurrent behaviors?

Asynchronous Automata



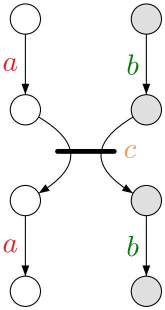
- To distribute the behaviors we need to first distribute the actions.

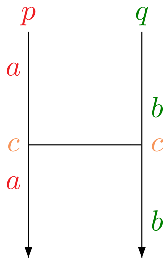
Asynchronous Automata



- 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.
- Define behaviors and automata over this distributed alphabet

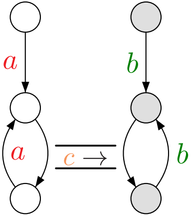
Asynchronous Automata

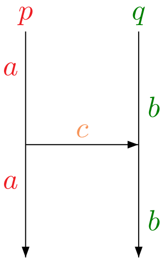




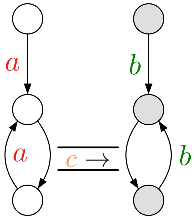
- What are the properties of languages accepted by such automata? E.g. above accepts $\{\underline{abcab}, bacab, bacba, abcba\}$.
- Given a language L, (when) can it be accepted by such an asynchronous automaton?

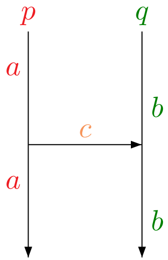
Message Passing Automata





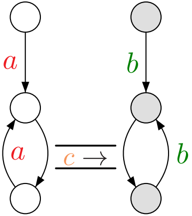
Message Passing Automata

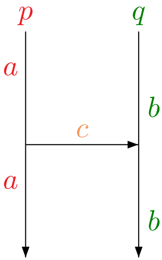




- In fact, this formalism is extremely (Turing complete for those who know!) powerful.
- We will consider algorithmic and decidability issues.

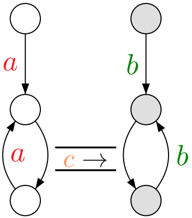
Message Passing Automata

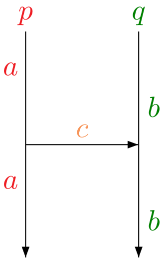




- In fact, this formalism is extremely (Turing complete for those who know!) powerful.
- We will consider algorithmic and decidability issues.
- (Surprising fact: If you are allowed to lose messages randomly then it is decidable!)

Message Passing Automata





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

Generalizing...

- In general, these are examples of concurrent structures, but also often involve infinite-state objects.

Generalizing...

- In general, these are examples of concurrent structures, but also often involve infinite-state objects.
- If you don't like "state objects", think of them as infinite discrete structures!

Generalizing...

- In general, these are examples of concurrent structures, but also often involve 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 concurrent structures, but also often involve 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 concurrent and infinite discrete structures!

Generalizing...

- In general, these are examples of concurrent structures, but also often involve 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 concurrent and infinite discrete structures!

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.

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.

Prerequisities

- Discrete structures (hard req!)
- Automata theory (soft req)

Logistics

Evaluation (tentative and flexible... upto a point)

- Assignments/minor exam: 20%
- Programming assignment: 20%
- Paper presentation: 25 %
- Exam (Major): 35 %

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

Logistics

Evaluation (tentative and flexible... upto a point)

- Assignments/minor exam: 20%
- Programming assignment: 20%
- Paper presentation: 25 %
- Exam (Major): 35 %

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`
- Moodle/Piazza will be set up soon