ELEC843
FIRST LECTURE

Karen Rudie, Ph.D., P.Eng.

Department of Electrical & Computer Engineering

Cross-appointed to School of Computing
What are Discrete-Event Systems (DES)?

• Processes where behaviour is described by sequences of events
• Useful in problems where specifications require prescribed relative ordering of actions, e.g., “close valve” *after* “liquid level reaches 3 cm”
• Abstracts chunks of information for high-level control
• We reason about sequences of events
  • So proofs are typically by induction on length of event sequence
Discrete-Event Systems (cont’d)

- Use discrete math, not continuous-time math (i.e., no real analysis, no calculus, no differential equations)
  - No relationship to classical control systems

- Automata theory, graph theory, mathematical logic courses all helpful (but not prerequisites)
Examples

- Machinery operation in FMS
- Fault detection and isolation: HVAC system
- Advanced manufacturing/automation applications
- Concurrency control of software
- Emergency response protocols
- Telecommunication protocols
DES Control

• Started in 1980s by Peter Ramadge and W. Murray Wonham (Ramadge’s Ph.D. thesis under supervision of Wonham)
  • RW Theory

• Idea:
  • separate plant from supervisor (controller)
  • find conditions and algorithms to guarantee control of plant meets specifications

• Basic Model: finite-state automata
Syncrude Oilsand Extraction Process
How to Model a DES?

The system is made of 1 shovel and 1 truck:

I=Idle, W=working, D=down

E=empty, F=full, B=broken

system DES => 9 states and 24 transitions
How to Control?

• Control objectives are desired sequences of events
  • spells out the legitimate orderings of events and rules out sequences that do not obey those orderings
  • captures constraints
  • ideal for safety-critical specifications

• Desired sequences are result of specifications
  • also translated into finite-state machines

• Disable certain events after certain sequences
How to Control? (cont’d)

- DES Control Theory: develop formal methods for determining which events should be disabled given that
  - some events are not observable
  - and some events cannot be disabled
- Optimal solutions are “minimally restrictive”
  - allow as many event occurrences as possible
- Provides solutions that are correct by construction
Sample Specifications

(i) two buckets needed to fill a truck

(ii) priority on truck repair: if both broken, repair truck 1st

Both specifications => 6 states and 33 transitions
Supervisor Solution
Example: One Legal Path

0 -> dig -> 1 -> load -> 2

0 -> load -> 5

0 -> breakdown

3 -> dig

4

7 -> flat tire

10 -> dig

8 -> repair_t

14
Other Examples

• Dining Philosophers (Mutual Exclusion)
• Data Transmission Problem
• Emergency Response Protocols to Medical Outbreaks in Long-Term Care Facility (LTCF)
• Concurrency Control of Software
Dining Philosophers (Mutual Exclusion)

- Philosopher needs both forks to eat
Dining Philosophers (Mutual Exclusion)

- If Philosopher 1 has fork to left then Philosopher 2 shouldn’t pick up fork to right

![Diagram of Dining Philosophers]
Dining Philosophers (Mutual Exclusion)

- Supervisor disables events leading to deadlock

![Diagram of Dining Philosophers]

- Nodes represent philosophers
- Edges represent actions:
  - P1dropF1
  - P1dropF2
  - P1pickUpF1
  - P1pickUpF2
  - P2pickUpF2
  - P2dropF1
  - P2dropF2
  - P2pickUpF1
  - P2eat
  - P1eat

- Nodes 5, 4, and 3 represent actions P1eat
- Nodes 6 and 8 represent actions P2eat
Data Transmission Problem

- Sender gets arbitrarily long sequence of data from computer and passes data through unreliable channel to receiver
- Receiver must pass data to another computer in correct order and without duplicates
- Channel may lose or (detectably) corrupt messages—no reordering
- Sender may tag on extra information in header of data messages
- Receiver may send acknowledgements
Data Transmission (cont’d)

• Will show in course how DES control can be used to do protocol verification
• Channel losses, timeouts, message receptions are *uncontrollable* events
• Each agent (Sender and Receiver) observes only those events that occur at its physical location
  • All other events are *unobservable* to agent
• Need *decentralized* control
Data Transmission Solution (ABP)
Emergency Response Protocols

• Behaviour of patients, care workers, hospital administrators can be described by discrete-event systems
• Emergency response protocol prescribes actions that agents should take
• Control could amount to preventing certain actions at certain states
DES Model of a Patient in LTCF
Concurrent Control of Software

• Multi-threaded computer code describes event executions (i.e., each step of program is an action)
• Specifications describe ways in which instrumented code should guarantee certain concurrency requirements—e.g., “a process may not write to a space that is currently being read by another process”
• DES theory can be used to determine where in code concurrency control should be added
Applying DES to Concurrency Control
Concurrency Control Example

- All threads start together
- Thread-2, thread-3, and thread-4 (T2, T3, and T4) must wait for thread-1 (T1) to finish before executing their code
- Thread-4 must wait for thread-5 (T5) to finish
Solution
Example: Readers/Writers

1. One writer at a time
2. Many readers at a time
3. Max M readers pending
4. Max N writers pending

(Courtois, 1971)
Code Generation Process

( Dragert et al, 2008)
Code Generation Process

(Dragert et al, 2008)
Code Generation Process

(Dragert et al, 2008)

```
public class Reader {
    ...
    public void run() {
        ...
        Supervisor.next("+r");
        read();
        Supervisor.next("-r");
        ...
    }
}
```
Code Generation Process

(Dragert et al, 2008)
Code Generation Process

(Dragert et al, 2008)
Dynamic Threads
Dynamic Threads
Dynamic Threads
Dynamic Threads
Limited Lookahead

(Chung et al, 1992)
Other Ways to Model DESs

- Petri nets
- Modal logic, temporal logic
- Algebras
- Other types of automata (beyond finite-state automata)
Cat & Mouse

• $c_i$: cat goes through door in specified direction(s)
• $m_i$: mouse goes through door in specified direction
• door $c_7$ cannot be closed
Cat & Mouse

Decide which doors to open *when* s.t.

1. Cat & Mouse never in same room at same time
2. Both can always return home
   - Cat home: Rm 2
   - Mouse home: Rm 4
3. Cat & Mouse can move around as much as possible