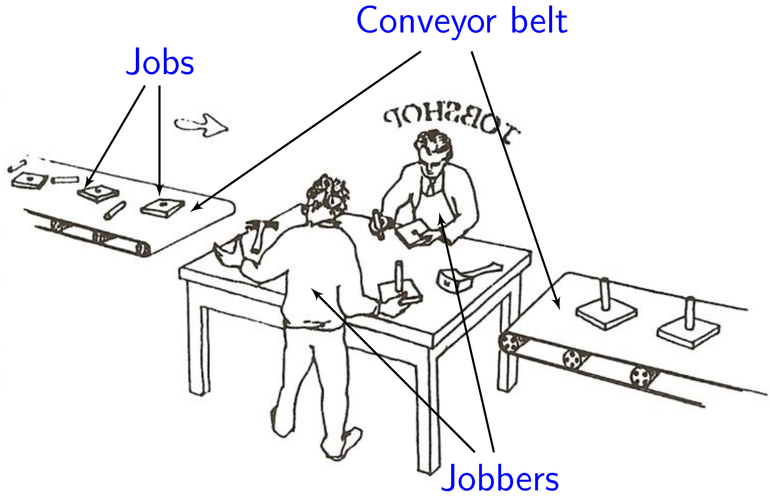
# CIS 4930/6930: Principles of Cyber-Physical Systems

## Timed Automata: A Case Study

Hao Zheng

Department of Computer Science and Engineering
University of South Florida

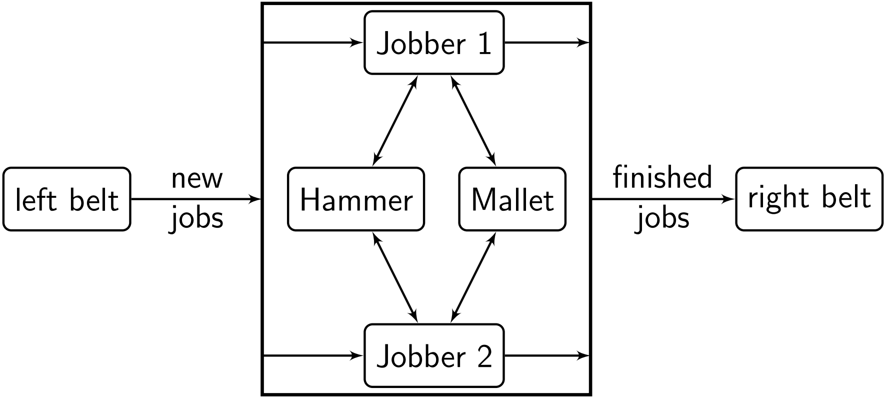
# A Jobshop



# A Jobshop

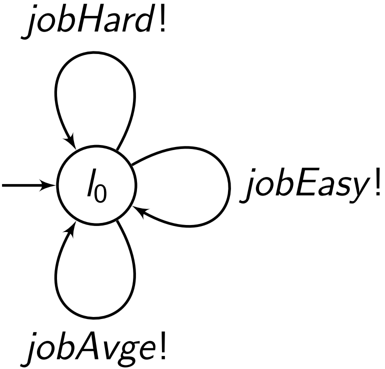
- Assume: two jobbers, and two tools: a hammer and a mallet.
  - These tools are shared by jobbers.
- A job can be easy, hard, or average.
  - If a job is easy, no tool is used.
  - If a job is hard, the hammer is used.
  - Otherwise, either the hammer or the mallet is used.
- The belts run around a constant speed, i.e.
  - jobs appear on one belt from time to time.
- Exact timing will be specified later.

# The Actor Model



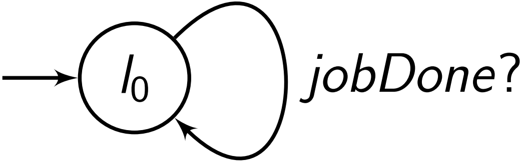
# Modeling Left Belt

This belt keeps sending jobs, `easy`, `hard`, or average, to the job shop.



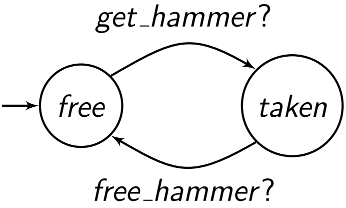

Three different channels have to be used as UPPAAL does not support passing values through channels.

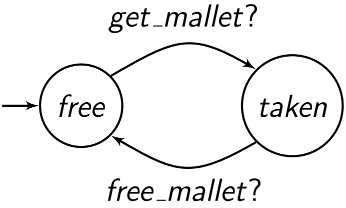
# Modeling Right Belt



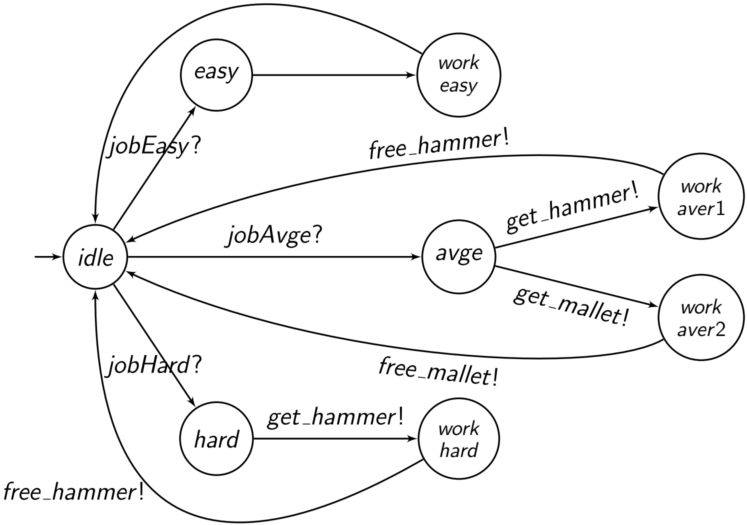
# Modeling Tools

A tool (hammer or mallet) can be *free* or *taken*.



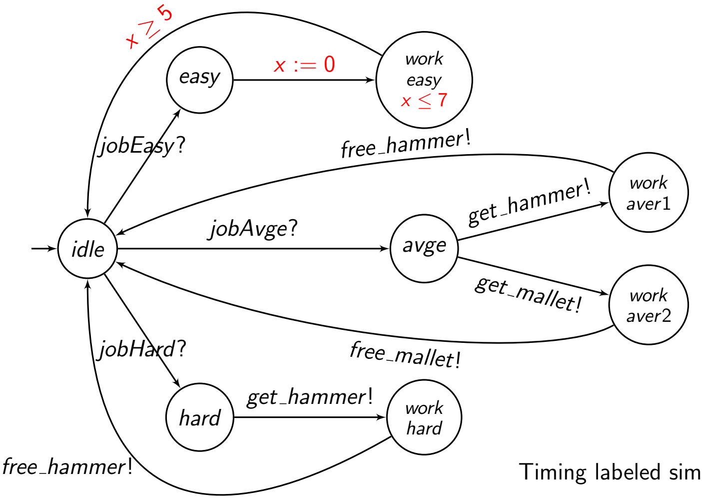
# Modeling Jobbers



# Timing for Jobbers

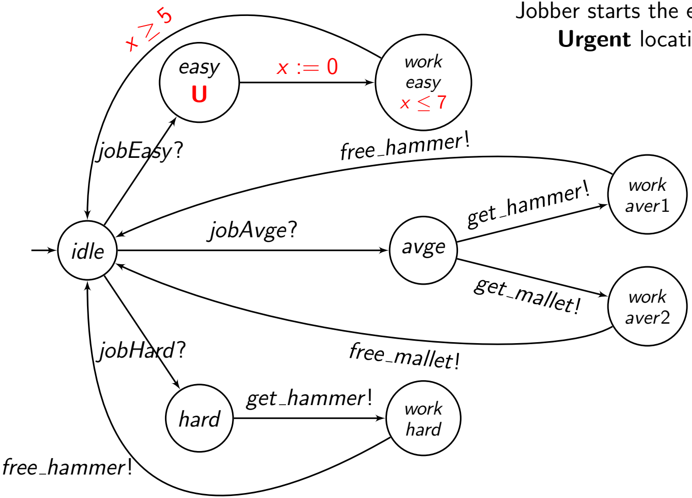
- $[5, 7]$ seconds to finish an easy job.
- $[10, 12]$ seconds to finish an average job with the hammer.
- $[15, 17]$ seconds to finish an average job with the mallet.
- $[20, 22]$ seconds to finish a hard job.

# Jobbers with Timing




Timing labeled similarly for other jobs.

# Jobbers with Timing (1)




Jobber starts the easy job immediately.
**Urgent** locations in UPPAAL.

# Communications

- Whenever a job is ready and a jobber is ready for the next job, the job is transferred immediately.
- Whenever a tool is free and a jobber needs it, the tool is transferred immediately.

**Urgent channels** in UPPAAL: whenever two edges

$$p \xrightarrow{ch!} p' \text{ and } q \xrightarrow{ch?} q'$$

are enabled, they take place immediately.

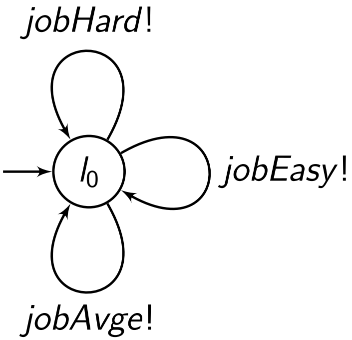
In our model,

```
urgent jobEasy, jobHard, jobAvge, get_hammer,
    get_mallet, free_hammer, free_mallet
```

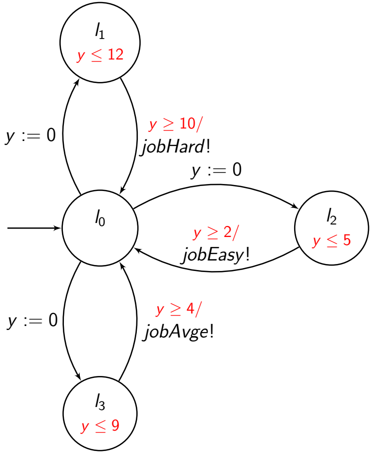
# Verification Problem 1

Is it possible that the left belt delivers jobs too fast for the jobbers to handle with the following timing parameters?

- An easy job is delivered within $[2, 5]$ seconds since last delivered job.
- An average job is delivered within $[4, 9]$ seconds since last delivered job.
- A hard job is delivered within $[10, 12]$ seconds since last delivered job.

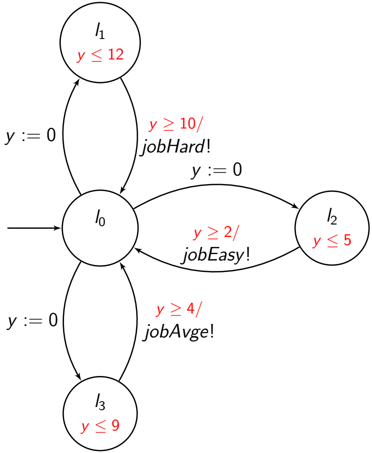


# Verification Problem 1: Modeling Left Belt




What would happen if the left belt is too fast such that jobbers are overwhelmed by too many jobs?

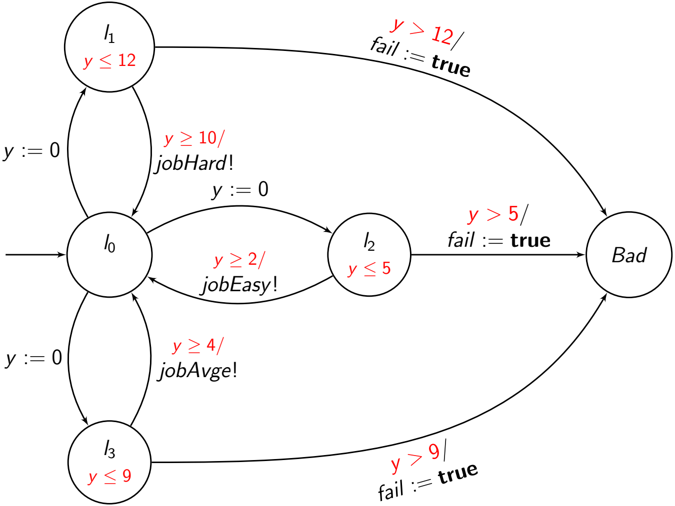
# Verification Problem 1: Modeling Left Belt




What would happen if the left belt is too fast such that jobbers are overwhelmed by too many jobs? **deadlock.**

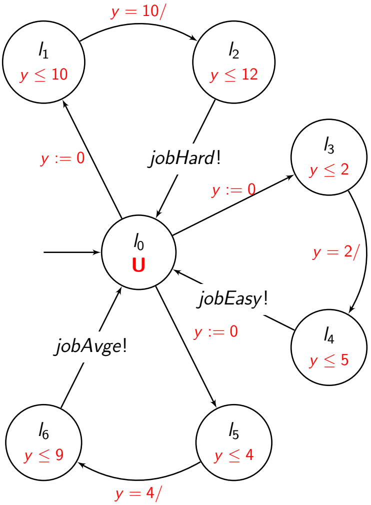
# Verification Problem 1: Modeling Left Belt

Or, the bad situation can be modeled explicitly.



# Modeling Left Belt: Another versioin

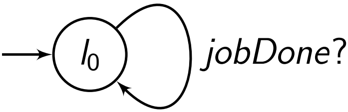
In UPPAAL, urgent channels cannot be combined with clock constraints!



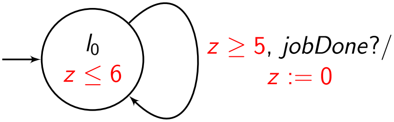
# Verification Problem 2

Suppose that the right belt runs in a speed such that it can take the finished jobs in every 5 to 6 seconds.

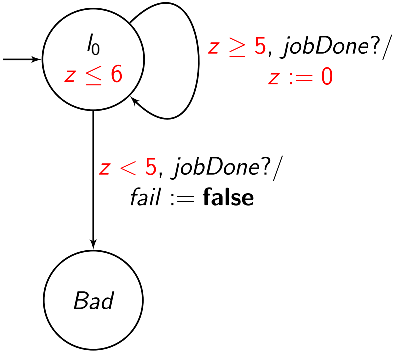
Can it take every finished jobs from the jobbers?



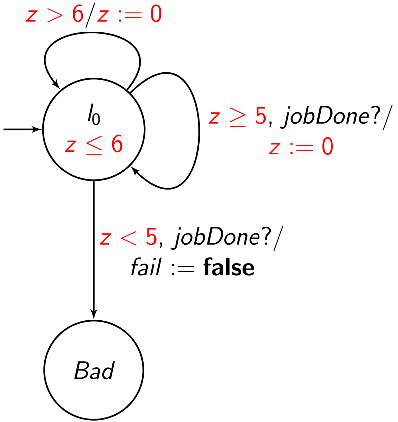
# Verification Problem 2: Modeling Right Belt



# Verification Problem 2: Modeling Right Belt



# Verification Problem 2: Modeling Right Belt



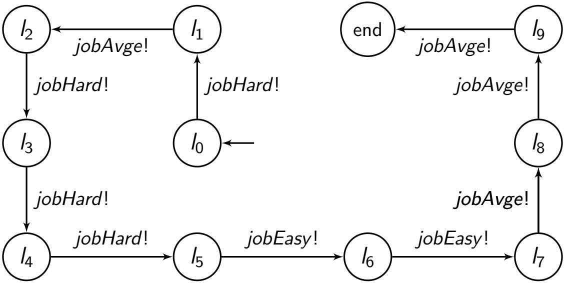
# Verification Problem 3

Given a sequence of jobs, what is the **minimal** amount time that all jobs are finished?

# Verification Problem 3

Given a sequence of jobs, what is the **minimal** amount time that all jobs are finished?

A new model for the left belt.



# Verification Problem 3

- Need to declare `clock now` to record the total time when all ten jobs are finished.

# Verification Problem 3

- Need to declare `clock now` to record the total time when all ten jobs are finished.
- Ask UPPAAL to check the following property

```
E<> (left_belt.end && jobber1.idle && jobber2.idle)
```

# Verification Problem 3

- Need to declare `clock now` to record the total time when all ten jobs are finished.
- Ask UPPAAL to check the following property

```
E<> (left_belt.end && jobber1.idle && jobber2.idle)
```

- UPPAAL will return a trace showing the satisfaction of the above property.
  - The trace includes the value of `now`, but not necessarily the minimal.

# Verification Problem 3

- Need to declare `clock now` to record the total time when all ten jobs are finished.
- Ask UPPAAL to check the following property

```
E<> (left_belt.end && jobber1.idle && jobber2.idle)
```

- UPPAAL will return a trace showing the satisfaction of the above property.
  - The trace includes the value of `now`, but not necessarily the minimal.
- Go to *Menu* → *Diagnostic Trace*, and select the option *Fastest*.
  - UPPAAL will produce a trace including `now` with the minimal value.

# Verification Problem 4

Given the same sequence of jobs for Problem 3, what is the **maximal** amount of time to finish all ten jobs?

- Computing the largest value for `now` can be done indirectly.
- Check the property

```
A[] now>=200 imply
     (left_belt.end && jobber1.idle && jobber2.idle)
```

# Verification Problem 4

Given the same sequence of jobs for Problem 3, what is the **maximal** amount of time to finish all ten jobs?

- Computing the largest value for `now` can be done indirectly.
- Check the property

```
A[] now>=200 imply
      (left_belt.end && jobber1.idle && jobber2.idle)
```

- If satisfied, what does it mean?

# Verification Problem 4

Given the same sequence of jobs for Problem 3, what is the **maximal** amount of time to finish all ten jobs?

- Computing the largest value for `now` can be done indirectly.
- Check the property

```
A[] now>=200 imply
     (left_belt.end && jobber1.idle && jobber2.idle)
```

- If satisfied, what does it mean?
  - It does not necessarily mean the maximal amount of time to finish all ten jobs. Time keeps passing by when the system is in

```
(left_belt.end && jobber1.idle && jobber2.idle)
```

# Verification Problem 4

- After showing the satisfaction of the property

```
A[] now>=200 imply
     (left_belt.end && jobber1.idle && jobber2.idle)
```

- Next, check

```
A[] now>=150 imply
     (left_belt.end && jobber1.idle && jobber2.idle)
```

# Verification Problem 4

- After showing the satisfaction of the property

```
A[] now>=200 imply
      (left_belt.end && jobber1.idle && jobber2.idle)
```

- Next, check

```
A[] now>=150 imply
      (left_belt.end && jobber1.idle && jobber2.idle)
```

- Sat'ed, then check

```
A[] now>=120 imply
      (left_belt.end && jobber1.idle && jobber2.idle)
```

# Verification Problem 4

- After showing the satisfaction of the property

```
A[] now>=200 imply
      (left_belt.end && jobber1.idle && jobber2.idle)
```

- Next, check

```
A[] now>=150 imply
      (left_belt.end && jobber1.idle && jobber2.idle)
```

- Sat'ed, then check

```
A[] now>=120 imply
      (left_belt.end && jobber1.idle && jobber2.idle)
```

- Unsat'ed, then check

```
A[] now>=135 imply
      (left_belt.end && jobber1.idle && jobber2.idle)
```

# Verification Problem 4

- Eventually, we will find out that

```
A[] now>=127 imply
      (left_belt.end && jobber1.idle && jobber2.idle)
```

  is satisfied, but

```
A[] now>=126 imply
      (left_belt.end && jobber1.idle && jobber2.idle)
```

  is not satisfied.

# Verification Problem 4

- Eventually, we will find out that

```
A[] now>=127 imply
      (left_belt.end && jobber1.idle && jobber2.idle)
```

  is satisfied, but

```
A[] now>=126 imply
      (left_belt.end && jobber1.idle && jobber2.idle)
```

  is not satisfied.
- This indicates that the maximal amount of time for all ten jobs to be finished is 126.