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

After starting Uppaal, we see the window displayed in Figure 1.4. The Uppaal graphical user interface consists of three main parts, accessible via three tabs in the main window: the system editor, which can be used to construct models, the simulator, in which the behavior of models can be simulated, and the verifier, in which the behavior of models can be analyzed. In this subsection we discuss the system editor, subsection 1.4 will present the simulator, and subsection 1.6 will present the verifier. Upon starting Uppaal, at first the system editor is displayed.

A Uppaal model (called system) is defined as a composition of a number of basic components (called automata or processes). Automata are diagrams with states (called locations) and transitions between states (called edges). The system editor has four drawing tools for building automata, see Figure 1.5, named Select, Location, Edge, and Nail.

- The Select tool is used to select, move, modify and delete elements. Elements can be selected by clicking on them or by dragging a rubber band around one or more elements. Elements can be added or removed from a selection by holding down the control key while clicking on the element. The current selection can be moved by dragging them with the mouse. Double clicking an element brings up a menu in which properties for that element can be specified. Right clicking an element brings up a pop-up menu from which properties of the element can be changed.

- The Location tool is used to add new locations. Simply click with the left mouse button in order to add a new location. Be careful: if one clicks several times new locations will be stacked on top of each other (a common mistake, leading to...
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

left belt \[\rightarrow\text{new jobs}\] Group of actors: 

- Jobber 1
- Hammer
- Mallet
- Jobber 2

finished jobs \[\rightarrow\text{right belt}\]
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

\[ I_0 \rightarrow \text{jobDone?} \]
A tool (hammer or mallet) can be free or taken.

\[\text{get\_hammer}\?\]

\[\text{get\_mallet}\?\]

\[\text{free\_hammer}\?\]

\[\text{free\_mallet}\?\]
Modeling Jobbers

idle

- easy
  - jobEasy?
- avge
  - jobAvge?
- hard
  - jobHard?

work easy

- free_hammer!

work aver1

- get_hammer!

work aver2

- get_hammer!
- get_mallet!

work hard

- free_hammer!
- free_mallet!
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.

```
idle
```

```
<table>
<thead>
<tr>
<th>Job</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>easy</td>
<td>get hammer!</td>
</tr>
<tr>
<td>U</td>
<td>free hammer!</td>
</tr>
<tr>
<td>work</td>
<td>get hammer!</td>
</tr>
<tr>
<td>x := 0</td>
<td>get hammer!</td>
</tr>
<tr>
<td>x ≤ 7</td>
<td>get hammer!</td>
</tr>
<tr>
<td>avge</td>
<td>free mallet!</td>
</tr>
<tr>
<td>jobEasy?</td>
<td>free hammer!</td>
</tr>
<tr>
<td>jobAvge?</td>
<td>get mallet!</td>
</tr>
<tr>
<td>jobHard?</td>
<td>get hammer!</td>
</tr>
<tr>
<td>hard</td>
<td>free hammer!</td>
</tr>
<tr>
<td>work</td>
<td>free hammer!</td>
</tr>
<tr>
<td>hard</td>
<td>free hammer!</td>
</tr>
</tbody>
</table>
```
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

\[ p \xrightarrow{ch!} p' \quad \text{and} \quad q \xrightarrow{ch?} q' \]

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?
What would happen if the left belt is too fast such that jobbers are overwhelmed by too many jobs? **deadlock.**
Or, the bad situation can be modeled explicitly.

Verification Problem 1: Modeling Left Belt

Or, the bad situation can be modeled explicitly.
In UPPAAL, urgent channels cannot be combined with clock constraints!

\[
\begin{align*}
  l_1 & : y \leq 10 \\
  l_2 & : y \leq 12 \\
  l_3 & : y \leq 2 \\
  l_4 & : y \leq 5 \\
  l_5 & : y \leq 4 \\
  l_6 & : y \leq 9 \\
  l_0 & : y = 0
\end{align*}
\]

\[y = 0/\text{jobHard!}\]

\[y = 10/\text{jobHard!}\]

\[y = 2/\text{jobEasy!}\]

\[y = 4/\text{jobAvge!}\]
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

\[ l_0 \]
\[ z \leq 6 \]
\[ z \geq 5, \text{ jobDone?} / \]
\[ z := 0 \]
Verification Problem 2: Modeling Right Belt

\[ l_0 \]
\[ z \leq 6 \]
\[ z \geq 5, \text{jobDone}?/ \]
\[ z := 0 \]
\[ z < 5, \text{jobDone}?/ \]
\[ \text{fail} := \text{false} \]

\[ \text{Bad} \]
Verification Problem 2: Modeling Right Belt

\[ \begin{align*}
  z > 6 & \implies z := 0 \\
  z \leq 6 & \implies \text{Bad} \\
  z \geq 5 & , \text{jobDone}?/ \implies z := 0 \\
  z < 5 & , \text{jobDone}?/ \implies \text{fail} := \text{false}
\end{align*} \]
Verification Problem 3

Given a sequence of jobs, what is the \textbf{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.

```
+---+    +---+    +---+    +---+
| l2 |    | l1 |    | l9 |    | l0 |
| jobAvge! | jobHard! | jobAvge! |
+---+    +---+    +---+    +---+
| l3 |    | l0 |    | l8 |    | l6 |
| jobHard! | jobHard! | jobAvge! |
+---+    +---+    +---+    +---+
| l4 |    | l5 |    | l7 |    | l6 |
| jobHard! | jobEasy! | jobEasy! |
```

H. Zheng (CSE USF)
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
  \[ E<> (left_belt.end && jobber1.idle && jobber2.idle) \]
Need to declare clock now to record the total time when all ten jobs are finished.

Ask UPPAAL to check the following property

\[ E<> (\text{left\_belt}.\text{end} \&\& \text{jobber1}.\text{idle} \&\& \text{jobber2}.\text{idle}) \]

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 \( \texttt{now} \) to record the total time when all ten jobs are finished.
- Ask UPPAAL to check the following property
  \( \texttt{E<> (left\_belt.end && jobber1.idle && jobber2.idle)} \)
- UPPAAL will return a trace showing the satisfaction of the above property.
  - The trace includes the value of \( \texttt{now} \), but not necessarily the minimal.
- Go to Menu → Diagnostic Trace, and select the option Fastest.
  - UPPAAL will produce a trace including \( \texttt{now} \) with the minimal value.
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 \texttt{now} can be done indirectly.
- Check the property
  \[
  \text{A[]} \text{ now}\geq 200 \implies \text{(left\_belt.end && jobber1.idle && jobber2.idle)}
  \]
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 $\text{now}$ can be done indirectly.
- Check the property
  \[
  A[] \text{ now} \geq 200 \implies (\text{left_belt.end} \land \text{jobber1.idle} \land \text{jobber2.idle})
  \]
- 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 $\text{now}$ can be done indirectly.
- Check the property
  \[ A[] \text{ now} \geq 200 \implies (left\_belt.end \&\& \text{jobber1.idle} \&\& \text{jobber2.idle}) \]
- 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
  \[ (left\_belt.end \&\& \text{jobber1.idle} \&\& \text{jobber2.idle}) \]
After showing the satisfaction of the property

\[ A[] \text{ now}\geq 200 \implies (left\textunderscore belt\textunderscore end && jobber1\textunderscore idle && jobber2\textunderscore idle) \]

Next, check

\[ A[] \text{ now}\geq 150 \implies (left\textunderscore belt\textunderscore end && jobber1\textunderscore idle && jobber2\textunderscore idle) \]
• After showing the satisfaction of the property
  \[ A[] \text{ now} \geq 200 \text{ imply} \]
  \[ (\text{left\_belt\_end} \&\& \text{jobber1\_idle} \&\& \text{jobber2\_idle}) \]
• Next, check
  \[ A[] \text{ now} \geq 150 \text{ imply} \]
  \[ (\text{left\_belt\_end} \&\& \text{jobber1\_idle} \&\& \text{jobber2\_idle}) \]
• Sat’ed, then check
  \[ A[] \text{ now} \geq 120 \text{ imply} \]
  \[ (\text{left\_belt\_end} \&\& \text{jobber1\_idle} \&\& \text{jobber2\_idle}) \]
Verification Problem 4

- After showing the satisfaction of the property
  \[ A[] \text{ now} \geq 200 \implies (\text{left_belt.end} \&\& \text{jobber1.idle} \&\& \text{jobber2.idle}) \]

- Next, check
  \[ A[] \text{ now} \geq 150 \implies (\text{left_belt.end} \&\& \text{jobber1.idle} \&\& \text{jobber2.idle}) \]

- Sat’ed, then check
  \[ A[] \text{ now} \geq 120 \implies (\text{left_belt.end} \&\& \text{jobber1.idle} \&\& \text{jobber2.idle}) \]

- Unsat’ed, then check
  \[ A[] \text{ now} \geq 135 \implies (\text{left_belt.end} \&\& \text{jobber1.idle} \&\& \text{jobber2.idle}) \]
Eventually, we will find out that

\[
A[] \text{ now} \geq 127 \quad \text{imply} \quad \\
(left\_belt.\text{end} \&\& \text{jobber1.} \text{idle} \&\& \text{jobber2.} \text{idle})
\]

is satisfied, but

\[
A[] \text{ now} \geq 126 \quad \text{imply} \quad \\
(left\_belt.\text{end} \&\& \text{jobber1.} \text{idle} \&\& \text{jobber2.} \text{idle})
\]

is not satisfied.
Eventually, we will find out that
\[ A[] \text{ now} \geq 127 \implies (\text{left\_belt\_end} \land \text{jobber1\_idle} \land \text{jobber2\_idle}) \]
is satisfied, but
\[ A[] \text{ now} \geq 126 \implies (\text{left\_belt\_end} \land \text{jobber1\_idle} \land \text{jobber2\_idle}) \]
is not satisfied.

This indicates that the maximal amount of time for all ten jobs to be finished is 126.