The Production Cell Assignment

Modeling and Verifying a Real-Time System using Uppaal

Assignment in DatorSystem II - RealTid
Fall 200-

Lab Assistant

name:	Therese Berg
email:	thereseb@it.uu.se
room:	1438
postbox:	67 (4th floor, building 1)

Introduction

The purpose of this lab is to give students an introduction to formal methods and model checking by modeling and verifying an industrial Production Cell application using Uppaal.

The assignment should be solved individually or in groups of two.

The Production Cell

The Production Cell case study [2, 4] is an attempt to define a realistic industrial application. It has been described in many formalisms, but the original specification of it doesn't contain any temporal issues. In this assignment we will use Uppaal [1, 3, 5] to model and verify some properties of the Production Cell.

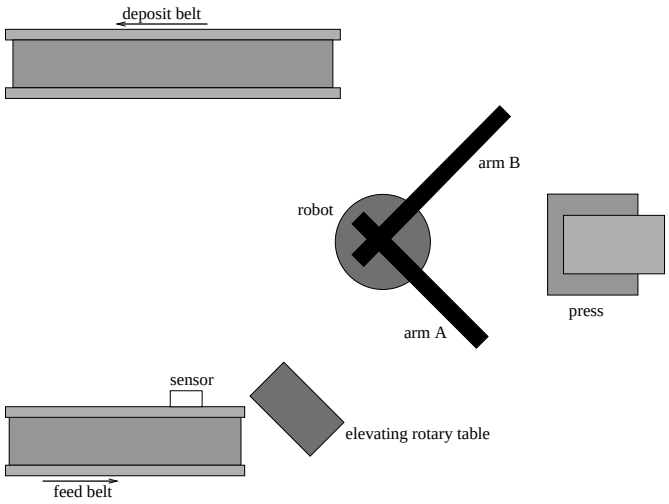


Figure 1: Production Cell

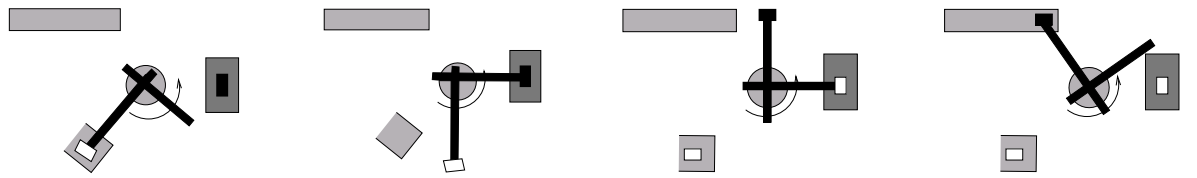
The Production Cell is intended to be a part of an industrial installation. Its purpose is to take metal plates from the feed belt, move them to the press, let the press do its work and then move the plates to a deposit belt. Plates are moved on the belts and by a robot. The

components of the system are shown in Figure 1. Arm A of the robot takes plate from the table (which itself must twist and rise when it gets a plate from the belt) and places the plate on the press. When the plate has been pressed, Arm B of the robot carries the plate to the deposit belt.

The arms of the robot are fixed with respect to each other (i.e. 90 degrees), so the robot controller must coordinate its operations of the two arms. The arms can also move horizontally (back and forth) but cannot move up or down. Instead the table and the press can move up and down and the belts are on different levels. You are not required to treat all of these movements in detail.

As all actions in the Production Cell take time it will be necessary to communicate to the robot that a plate has arrived at the Cell. This is achieved by means of a sensor.

The simplification in this model, compared with the original model, is the removal of a crane which moves plates from the deposit belt back to feed belt (it was an artificial element to produce a closed system). Instead of this we capture the continuous stream of plates by modeling a fixed number of plates circulating around the system, i.e. after leaving the deposit belt they become available for the feed belt again.



(a) Arm A at table (b) Arm B at press (c) Arm A at press (d) Arm B at deposit

Figure 2: Robot positions

Robot Movements

When a plate is moved from the Feed Belt to the table the table must be in its lowest position. The table then rises to a level where arm A can pick up the plate. The robot movements are then as follows (see Figure 2):

`arm A at table` - Arm A extends and pick up the plate from the table, see Figure 2(a).

`arm B at press` - The robot rotates counterclockwise until arm B points towards the press. Arm B is extended until it reaches the press. Arm B picks up the forged plate and retracts, see Figure 2(b).

`arm A at press` - The robot rotates counterclockwise until arm A can reach the press. Arm A extends, deposits its plate, and retracts again, see Figure 2(c).

`arm B at deposit` - The robot rotates counterclockwise until arm B points towards the deposit belt. Arm B extends and places the forged plate on the belt, see Figure 2(d).

When there is little to do for the robot it can move to a `wait`-position whose position is located somewhere between the `arm A at table`-position and `arm B at press`-position. The decision procedure for deciding on when to go to the `wait`-position is something you have to implement.

Device	Action	Time Units
Feed belt	Move to sensor	3
Feed belt	Move to table	1
Table	Raise and twist	2
Table	Return and twist	2
Press	Press plate	22–25
Press	Prepare for new plate	18–20
Deposit belt	Move plate out of cell	4
Robot	`Arm A at table` to `Arm B at press`	5
Robot	`Arm B at press` to `Arm A at press`	15
Robot	`Arm A at press` to `Arm B at deposit`	5
Robot	`Arm B at deposit` to `Arm A at table`	25
Robot	`Arm B at deposit` to `Wait position`	25 - X
Robot	`Arm A at press` to `Wait position`	20 - X
Robot	`Wait position` to `Arm A at table`	X
Robot	`Wait position` to `Arm B at press`	5 - X
Robot	At `Wait position`	2

Figure 3: Temporal values of Production Cell Activities

Temporal Parameters

In a realistic model all actions take time. Some operations take fixed time and other actions have maximum and minimum time bounds. The values which will be used in the study are given in the table in Figure 3. Actual time values are not significant, hence values are given in time units. From the table we can e.g. see that it takes the robot five time units to move from the `arm A at table`-position to the `arm B at press`-position.

The press requires some time to prepare for a new plate. The times for pressing and preparing for a new plate are not fixed but will vary.

To simplify the model we assume that all taking and leaving of plates are instantaneous, i.e. they will take no time to perform. Instead it is the movement between states, where plates can be taken and leaved, which takes time.

The movements given in Figure 3 are the only movements that the robot can perform, e.g. it is not possible to go from the `Arm A at press`-position directly to the `Arm B at press`-position. Also, no movement can be stopped or reversed, i.e. if a robot starts to move from one position to another it must move all the way and cannot stop on the way.

The X-value introduces some variability in the model and allow you to investigate the best location for the `Wait`-position. By changing the value of X we can change the location of the `Wait`-position. For example, if we want the `Wait`-position close to the `arm A at table`-position then X should be small. To make sure that the robot doesn't interfere with the rest of the devices when in `Wait`-position the possible values of X are limited to: $1 \leq X \leq 4$.

Modeling the Production Cell

Start by open the model of the system that you can find on:
`/stud/docs/kurs/realtid/production_cell/`
in Uppaal 2k. All different objects of the system are modeled as separate processes in Uppaal. The feeding belt, the sensor, the table, the robot, the press, the deposit belt and one or more plate processes. Of these all but the robot are given. Your task is to model a robot process and include it in the system so that the overall requirements are fulfilled.

Open the model of the system in the simulator, study what happens when you step through it. The already given parts of the system allows you to see how a plate moves on the feed belt to the table. But there it stops since there are no robot process there to handle it. Verify that the system will finally get stucked by stepping through the system using the simulator.

Study what channels the different given parts of the cell uses, what integer variables are declared. Your robot model will need local variables, e.g. to model that an arm has a plate or that there is a plate in the press, but you do not need to change any of the other models in the system. You might have to restrict some movements so they are not possible unless some specific properties, e.g. that `Arm B` must be holding a plate, are fulfilled.

The channels calls your robot will use for synchronization are:

- `plate_taken?` - to take a plate to `Arm A` from table.
- `get?` - to leave a plate from `Arm A` to press.
- `take?` - to get a pressed plate from the press to `Arm B`.
- `put?` - to put a plate from `Arm B` on deposit belt.
- `read!` - to get tell the `Sensor` that the robot is ready for getting a new plate.

Observe that all channels are declared `urgent`. This means that the transition must be taken immediately if both participating automata are ready for the synchronization (and all other guards are fulfilled). Observe that it is not allowed to have clock guards on synchronization transitions.

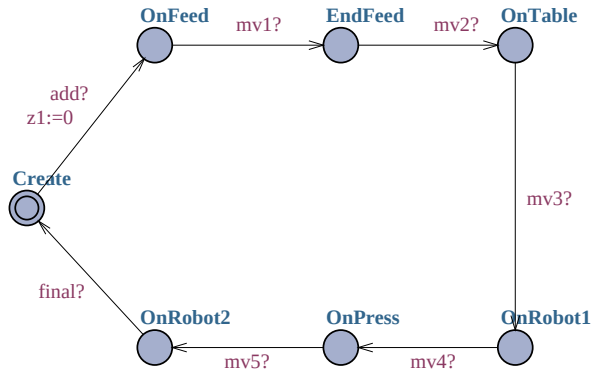


Figure 4: Template for the Plate Process

An Example

A template for the circulating plate process is shown in Figure 4. `Create` is an initial state. Process in the whole model is made by communicating on urgent channels. Note, that the plate is a passive component of the system, it only circulates in it, so it cannot force the

progress. Because of this, each plate has only a single local clock, that is used for checking the bounded liveness. The meanings of the (urgent) synchronization channels for this Finite State Machine (FSM) are as follows:

- `add` is linked to FSM representing the feed belt, by this channel the plate moves to the state OnFeed.
- `mv1` - moving the plate to the end of the feed belt.
- `mv2` - plate movement to the table.
- `mv3` - from the table to the robot (the Arm A).
- `mv4` - by the arm A to the press.
- `mv5` - from the press to the robot (the Arm B).
- `final` - to the deposit belt.

Observe that the plate never will synchronize with the Robot directly but only with the other devices in the system.

Modelling Tips

Some tips for doing the modelling in Uppaal are:

- If you want to model that certain amount of time should pass, for example for going between the `Arm_A_at_Table-` and `Arm_B_at_Press`-position, you do it most easily by creating a new state, say `A_Table_to_B_Press`, which symbolize the transition. Let the in-arrow to the transition-state reset a clock, say `c:=0`, which will be used to constrain the time we could stay in the state. By setting a guard on how long we can stay in the transition-state, `c<=5`, and a guard on the out-arrow on when we can leave the state, `c==5`, we can make sure that the transition lasts for exactly `5` time units, see Figure 5.

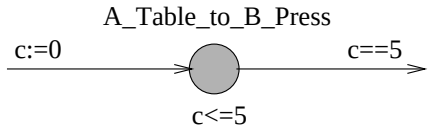


Figure 5: Time passage

- Since the robot can arrive to a state before the corresponding device it shall synchronize with is ready, you might sometimes have to model that the robot are at the same position, but in two different states. See for example Figure 6 where the `Arm B at Deposit`-position has been divided into two states, `Arm_B_at_Deposit_Wait` and `Arm_B_at_Deposit`, since the Robot can't deliver the plate until the Deposit Belt is ready for receiving it.

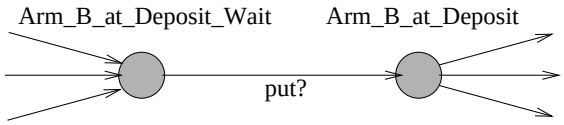


Figure 6: Waiting state

- To guarantee progress in your model you might have to set some states as `urgent`. This tells Uppaal that the Robot is not allowed to stay in that state and must therefore immidiately leave it.

Requirements for the Model

After you have constructed the Robot FSM it is time to verify some properties of the system. For this you will be using the verifier part of the Uppaal tool. You can give a question about some property of your system to the verifier, and if the analysis shows that it is not fulfilled you can step through a generated counterexample in the simulator.

A verification of a system implementation involves predicting the temporal worst-case behavior of the system when all delays are accounted for, and the subsequent assertion that in such circumstances all deadlines are satisfied. For each model we could make sure that two properties are satisfied: *safety* of the model (i.e. it must be possible to show that undesirable state cannot be reached from any legal initial conditions) and *bounded liveness* of the model (i.e. it can be asserted that all deadlines are feasible).

To capture the continuous stream of plates a fixed number of plates should be modeled as circulating around the system, i.e.:

- After leaving the Deposit Belt the plates become available for the Feed Belt again.
- There is a maximum of only one plate on the Feed Belt at a time.

What You Should Verify

The following properties should be verified (and answers given for) your Production Cell model:

1. That your system is deadlock free. Currently there is no support for deadlock checking available using Uppaal's graphical interface[1], instead we have to use Uppaal's stand-alone verifier `verifyta`. The procedure for doing deadlock checks are as follows:
   i. Use the `Export As..` option in the `File` menu to generate a `.ta`-file of your system, e.g. `MyProdCellModel.ta`.
   ii. Create a question that will make the verifier to investigate the whole search space, for example: `A[] (1 == 1)` (which asks if it is always true, for every possible execution, that 1 is equal to 1). Create the question in the verifier and save it in a file, e.g. `MyDeadlockQuestion.q`.
   iii. Call `verifyta` in a shell with an option that display (deadlock) warnings as queries[2], and save the generated warnings in a file, e.g. `MyDeadlockWarnings.q`:
   `verifyta -Q MyProdCellModel.ta MyDeadlockQuestion.q > MyDeadlockWarnings.q`
   iv. Check if you got any deadlocks by investigating the generated file `MyDeadlock-Warnings.q`. The easiest way to do this is to open the query file in Uppaal's graphical interface. If you got deadlocks in your system you will have (a number of queries) looking e.g. like `E<> (P1.OnPress and ...)`. You can now, by pushing the `Model Check` button generate a trace for the deadlock which you can step through in the simulator. To generate traces make sure that you have the checked the `Diagnostic Trace` box in the `Options` menu.

   Validate that your system is deadlock free both for one as well as several number of plates (see below).

[1] The Uppaal gang has promised to add deadlock checking in their next version of Uppaal :)
[2] Run `verifyta` without any options to get a listing of the features it support.

2. That the worst-case time for a plate from arrival on the feed belt to removal from the deposit belt must be less than 300 time units. This temporal requirements on the Cell concern the end-to-end movement of any plate. This is a deadline for the model and hence represents a bounded liveness property to be verified. In Uppaal this can verified as as the property:
   `A[] (not P1.Create imply P1.z1<=300)`,
   where `Create` is a location of a plate process and `z1` is a clock for the plate P1 (see Figure 4).

   Determine the worst-case time for the plate within the system by systematically decreasing the time value. What is the worst-case time for a plate in your system?

3. It should also be investigated, by experimenting with the model, how many plates that must be in the system to keep it fully busy and within the worst-case end-to-end requirement for all plates.

   If too many plates are added to the system you might experience that plates get stucked on a location forever (and the worst-case requirement will not be fulfilled).

   If to few plates are added to the system the robot will be idle waiting for plates to serve, (formally it means that you should verify that whenever the feed belt is ready to take a plate, a plate is ready to be added to the feed belt).

   It must also be proved that plates cannot overtake each other, i.e. it is necessary to show that their locations in the system are occupied under mutual exclusion (the safety property). In Uppaal this can be said as:
   `A[] (P1.OnFeed imply not (P2.OnFeed or P3.OnFeed or P4.OnFeed))`[3].

   Determine how many plates that must circulate in your system to keep it fully busy and within the worst-case requirement.

4. Finally, there are open issues that the design must address. In particular, when a plate has been placed on the press (which has a relatively long operating period) where should the robot wait, i.e. what is an optimal value for X? If it is close to the press it will take time to move to the table if the new plate arrives. But if it moves to the table and no plate enters the system it will have further to move to get back to the press. Having decided on a resting place, the duration of this wait has to be fixed. Eventually the robot must service the press in order to ensure liveness but it must also attempt to minimize the end-to-end time that any plate spends in the system. By manipulating the parameters of the models it is possible to investigate the impact of these design decisions. This is an important secondary purpose of verification: it allows design alternatives to be explored.

   Determine the optimal value for X by experimenting with your model. Is there a difference in the importance of X depending on the number of plates in the system?

[3]Note that this property is only partial, you will have to fill in the details for the right number of plate processes.

Report

A proper report is to be handed in for the assignment. It should include at least the following:

- The cover page provided at the end of this document.
- A printout of your constructed Robot FSM and a detailed description on how it is supposed to work. Make sure that your states and variables are descriptively named after what they are supposed to do.
- Answers and discussions on the properties that should be verified.
- Discussion of the concepts involved, i.e. well thought out and coherent comments regarding the topics covered by the assignment, as well as experimental results (if any) and reasonable conclusions.

This assignment must be handed in no later than 8h00 on Januari 16, 2006.

Some general guidelines of how to make a report:

- Use a word processor, text editor or typewriter to type your solutions. Print your constructed Robot FSM by saving it as postscript file (in the `Template` menu).
- Use English in your report.
- Do not print on both sides of the pages.
- Staple your report thoroughly.
- Staple your report with the pages in correct order.
- Leave room in the upper left corner of each paper for the staple.

References

[1] Johan Bengtsson, Kim G. Larsen, Fredrik Larsson, Paul Pettersson, Wang Yi, and Carsten Weise. New Generation of UPPAAL. In *International Workshop on Software Tools for Technology Transfer. Aalborg, Denmark*, 1998. `http://www.docs.uu.se/docs/rtmv/papers/bllpww-sub98-1.ps.gz`.

[2] A. Burns. How to Verify a Safe Real-Time System. The Application of Model Checking and a Timed Automata to the Production Cell Case Study. Technical report, Real-Time System Research Group, Department of Computer Science, University of York, 1998.

[3] Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a Nutshell. *Springer International Journal of Software Tools for Technology Transfer 1(1+2)*, 1997. `http://www.docs.uu.se/docs/rtmv/papers/lpw-sttt97.ps.gz`.

[4] C. Lewerentz and T Lindner. Case Study Production Cell, Formal Development of Reactive Systems. In *LNCS, Springer Verlag*, volume 891, 1995.

[5] The Uppaal Toolkit, `http://www.uppaal.com`.