

In this lab we will be working on a state diagram for the Production Cell controller (see lab 1).

Attendance/Demo

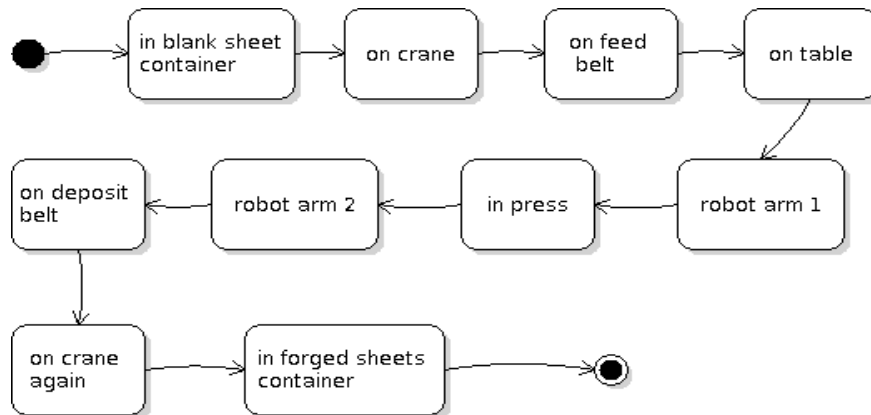
To receive credit for this lab, you must make reasonable progress towards completing the exercises. When you have finished all the exercises, call your TA, who will review your work. For those who don't finish early, the TA will ask you to show whatever diagrams you have completed, starting at about 15 minutes before the end of the lab period. Finish any exercises that you don't complete on your own time.

Introduction

Recall the first lab during which you conducted an initial requirement elicitation for the Production Cell system.

During this laboratory you will specify the overall behaviour of the control software with a state machine. We will assume that the system only handles one sheet at a time: considering sheets in multiple locations will make it overly complex (although in a real system, this would be the case).

We will take the process from the beginning. A metal sheet goes through a process that we can simply describe as a sequence of states:



These steps gives us an idea of what the control software must do: we can imagine that at the transitions between these states, the controller software must activate or deactivate machines: for example, as the feed belt goes from the feed belt to the table, it must switch off the feed belt, and activate the table to make it rotate and go up.

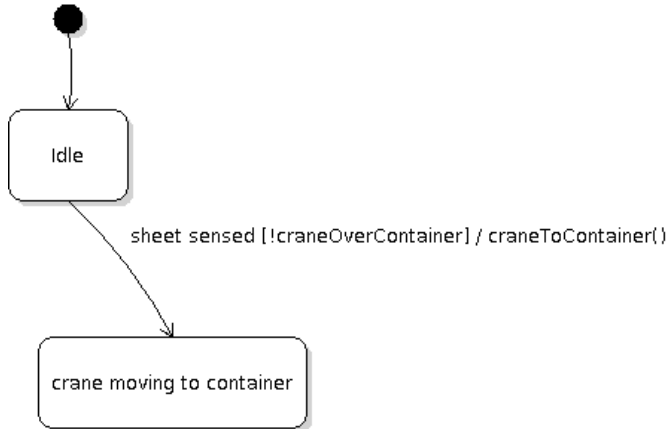
So this suggests that the controller software will be in “waiting” states while machines perform a process, such as conveying the sheet, lifting it, or forging it.

In order to coordinate the overall process, the controller must receive signals indicating the status of the machines and the sheet, e.g. a sheet is present at the end of a belt, or the crane is at a specific position. These signals will be events and conditions that will allow the control software to change state (and initiate the next step of the process).

At state transitions, the software will also have to send signals to the actors (machines): these signals are **commands** to the actors to perform tasks such as starting or stopping (sent to feed belts), moving to the right height and at the right angle (sent to the table), pressing (sent to the press)...

As an example, at the beginning of the process, the controller must send commands to the crane to pick up a sheet from the blank sheets container and place it on the feed belt.

We can work from the use case Load Belt (see lab 1 solution).:

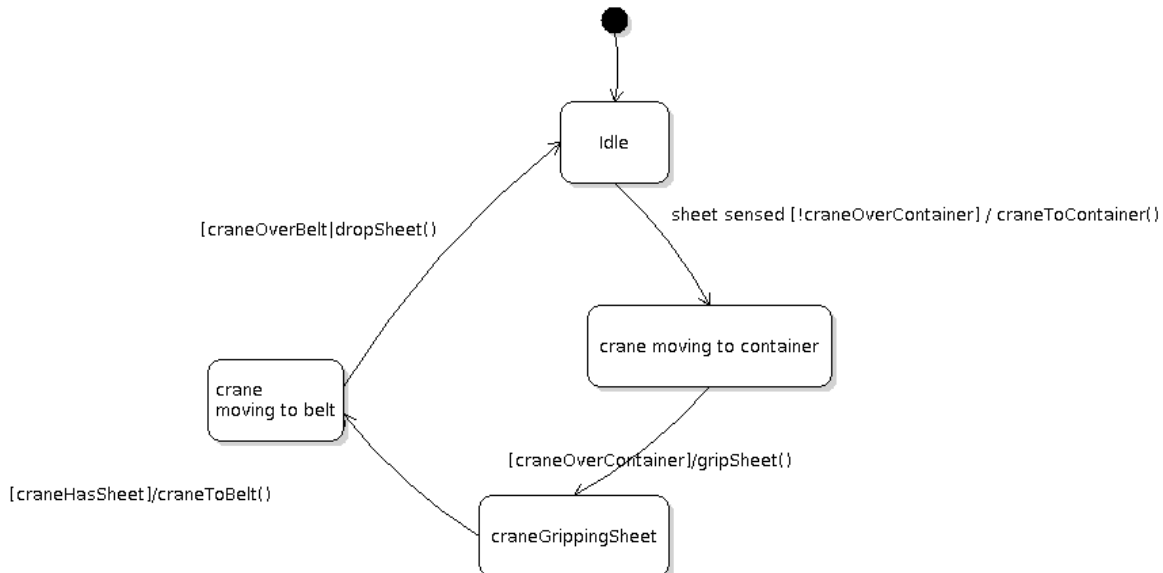


The controller software is initially idle. The system must check (through a photoelectric sensor) whether sheets are available in the blank sheet container. This is the event “sheet sensed”, that triggers the transition.

If the crane is not over the container, then the software must send a command to move towards the container: The condition is `[!craneOverContainer]` (the square brackets indicate a condition, the exclamation mark is logical negation), and the command is `craneToContainer()`. We use (round) brackets to denote a command.

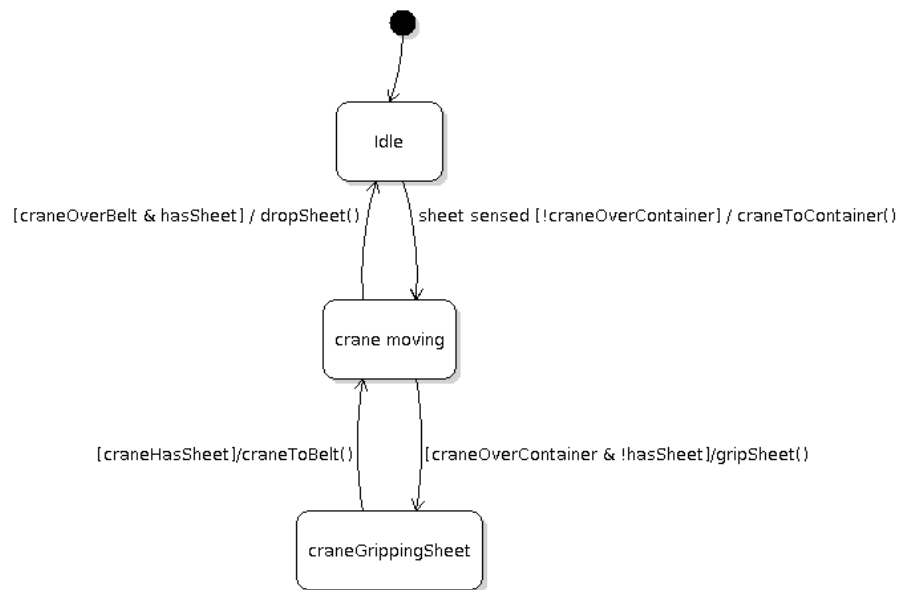
the exclamation mark is logical negation), and the command is `craneToContainer()`. We use (round) brackets to denote a command.

Following the same idea, we can extend this state machine to the next steps of the use case: once the crane is in position, the controller sends a command to grip a sheet. After that the crane must move to the belt, and release the sheet. If we limit ourselves to controlling the crane, we obtain a state machine like the one below.



Then we can observe that the states where the crane is moving can perhaps be merged. We then obtain the state machine on the next page.

We added a condition to the transitions out of the new state “crane moving”, just to make sure the crane doesn't try gripping a sheet when it already has one, or dropping it when it doesn't have one.



Your turn! Following this approach, extend this state machine to handle the next steps of the process. You may not have time to fully complete the state machine during this lab, but it will be a good exercise to finish on your own time.