CS 522 Assignment 3 – Statecharts

Spencer Borland
(sborla@cs.mcgill.ca)

1 Introduction

The objective of this assignment is to model the dynamics of a compact disc player software application using the Statechart formalism. Your solution should meet all the application requirement specified in the assignment distribution.

You may wish to have a look at the COMP522 Statecharts projects where slightly modified requirements were actually implemented, linking a GUI to a CD device driver.

One possible solution can be seen in Figure 1 (dia source).

The `CD` state has been zoomed-out and its inner dynamics can be seen in Figure 2 (dia source).

One advantage of modelling a system with Statecharts is that it is easy to verify that each requirement for the system has been met. Let us now go through each requirement in the assignment specification and ensure that each has been fulfilled.

Requirement 1 The `CLOSE_WIN` event is generated by the GUI when closing the application window. The system ends in the `DEAD` state.

Requirement 2 There are 7 events generated by the 7 corresponding buttons: `PLAY-PAUSE`, `STOP`, `PREVIOUS`, `NEXT`, `REVERSE`, `FORWARD`, `EJECT`.

Requirement 3 There is a `Display` object which has a method `setTime(time)` to set the time text field.

Requirement 4 There is a `Display` object which has a method `setTrack(track)` to set the track text field.

Requirement 5 The state `No CD` is entered via the condition `not cd_loaded()` which is within the `Tray In` contour.

Requirement 6 If `cd_loaded()` is true, the system moves to within the `CD` state, which consists of either `PLAYING`, `PAUSED` or `STOPPED`.

Requirement 7 The transition from the `Tray In` state to the `Tray Out` state is activated by the `EJECT` event.

Requirement 8 The transition from the `Tray Out` state to the `Tray In` state is activated by the `EJECT` event. Once in the `Tray In` state, the system will determine if a CD is present via the condition `C`.

Requirement 9 The only transition leading out of the `No CD` state is the transition from the `Tray In` state to the `Tray Out` state. Thus, no other event will have an effect on the system while in this state.

Requirement 10 There is a transition from the `Stop` state (which contains `Stopped`) to the `Playing` state. This transition is activated by the `PLAY-PAUSE` event.

Requirement 11 There is a transition from the `Playing` state to the `Stop` state. The default transition in the `Stop` contour leads to the `Stopped` state. This transition is activated by the `STOP` event.

Requirement 12 There is a transition from the `Playing` state to the `Paused` state which is activated by the `PLAY-PAUSE` event.

Requirement 13 There is a transition from the `Paused` state to the `Playing` state which is activated by the `PLAY-PAUSE` event.

Requirement 14 The transitions leading into the `Stop` state cause the default transition to `Stopped` to fire. One of the actions of this default transition is to set the time to 0 and the track number to 1. Notice that this requirement conflicts with requirements **20** and **21**. If the system is in the `Stopped` state, then the display should have the track number set to 1. However, while in the `Stopped` state, the `Next` button may be pressed. Once the system returns to the `Stopped` state, the track number should now read 2. Thus, it is necessary to move the state of the system to the `Paused` state, otherwise we will violate this requirement.

Requirement 15 The default transition in the `Stop` state has the action `StopButton.disable()`.

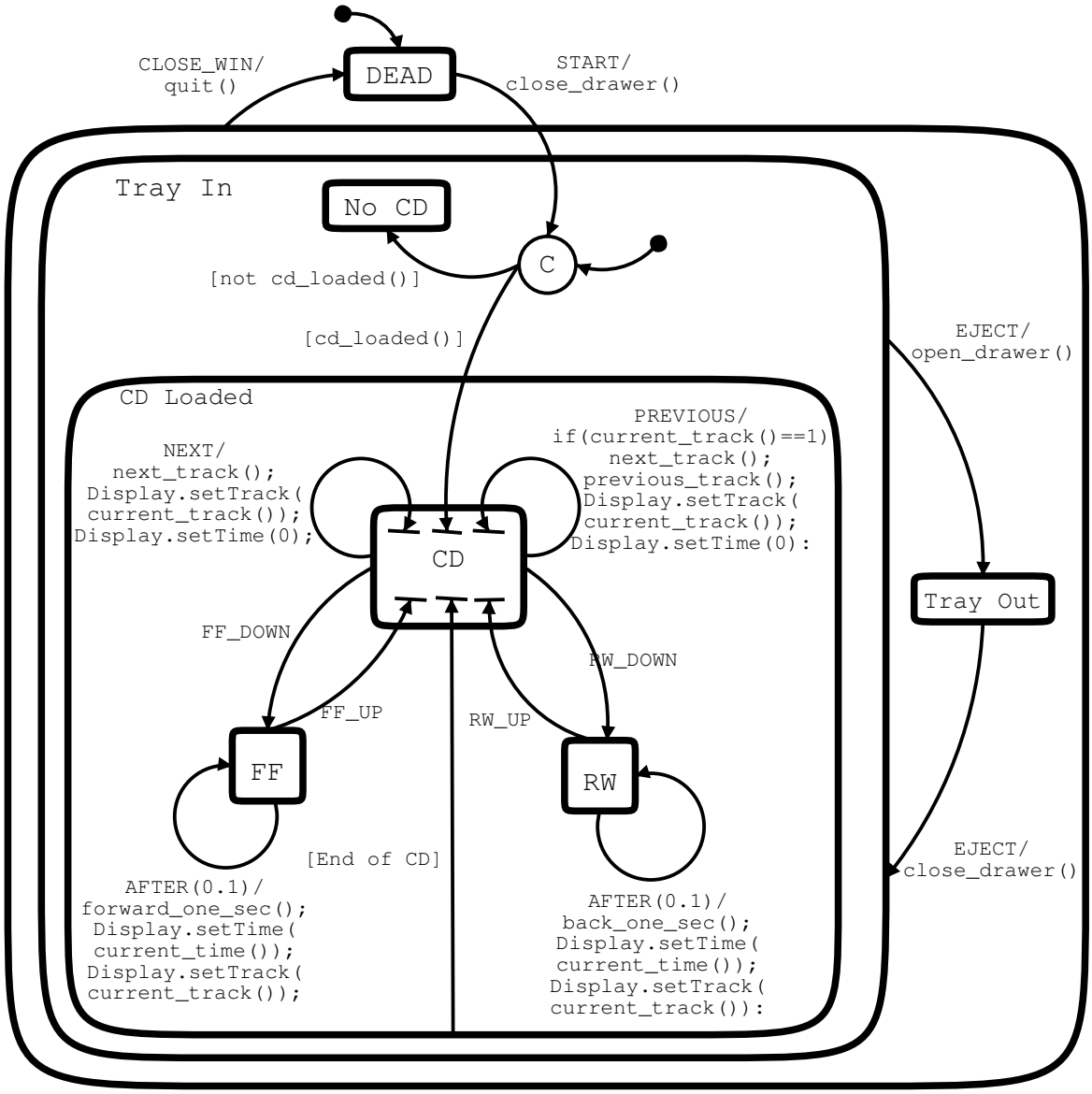

Figure 1: The CD player Statechart model

Requirement 16

Requirement 17 There is a transition from the `CD` contour to within `CD` activated by the `NEXT` event. This transition has the action `next_track()` which increments the track number. If the current track is the last track, then `next_track()` automatically set the track number to 1, which is the desired effect.

Requirement 18

Requirement 19 There is a transition from the `CD` contour to within `CD` activated by the `PREVIOUS` event. This transition has the action `previous_track()` which decrements the track number. If the current track is the first track, then `next_track()` is called before the call to `previous_track()`. This ensures that we get back to the beginning of the first track.

Requirement 20

Requirement 21 Without loss of generality, we will only discuss the re-wind action for these two requirements. There is a transition from the `CD` contour to the `RW` state (re-wind), activated by the `RW_DOWN` event. `RW` has a timer set on it which expires after 0.1 seconds. After this time, the transition fires and several actions are executed. One if these actions is to move the hardware's clock `back_one_sec()`. We also update the display accordingly. Notice that at each iteration of the re-wind cycle, the system checks if the end of the CD has been reached. This check is

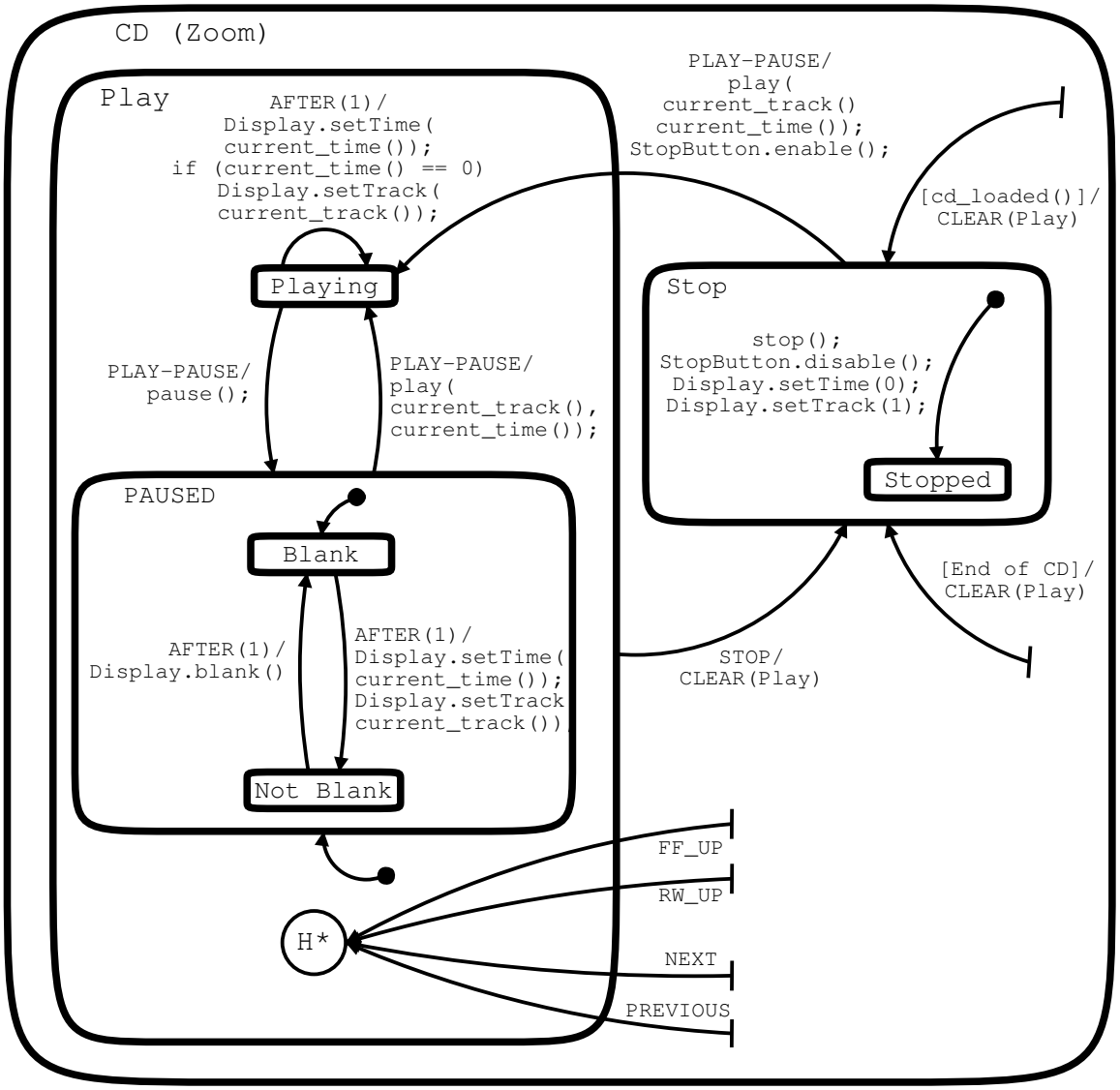

Figure 2: The CD contour zoomed

implemented in the transition from the CD Loaded contour to within the CD contour. Once the re-wind button is released, the RW_UP event is generated. This causes the system to move to last known state within the Play contour. If no history is available, then we revert to the default state, Paused. This ensures that after the re-wind button is pushed and released, and if the system is in the Stopped state, we end up in the Paused state so that requirement 14 is not violated.

Requirement 22 The Playing state has a transition from itself to itself which is activated on a 1 second timer. When the timer expires the time and track number displays are updated accordingly.

Requirement 23 The internal dynamics of the Paused state alternate between the Blank and Not Blank state. This alternation first calls the blank() method of the display object, then later sets the time and track number to the current_time() and current_track(), respectively.

1.1 Added Behaviour

Since most CD players allow you to press stop while the player is paused, we have added the contour Play around the Playing and Paused states to allow the user to stop the playback if the system is either playing or paused.

1.2 Extra Notes

Note that we must reset the history for the `Play` contour when moving to the `Stopped` state since we may leave the `CD` state via the re-winding/fast-forward or track increment/decrement events. We do this because, if we reach the `Stopped` state, then a fast forward action occurs, we want the system to return to the `Paused` state upon the release of the fast forward button so that requirement 14 is not violated.