15-424/15-624/15-824 Recitation 5
Event and Time-Triggered Control
15-424/15-624/15-824 Foundations of Cyber-Physical Systems

These notes are based upon notes from repvious years by João Martins and Stefan Mitsch.

These recitation notes explain how to convert a simple event-triggered controller into a time-triggered controller. Important modeling techniques applicable to both types of models will be discussed.

1 The Example

Imagine a ball moving between two lines and consider the problem of ensuring that the ball does not go past either line. The only actuation device available is a paddle that can be used to modify the ball's current velocity. Intuitively, a correct controller should detect when the ball is approaching one of the lines and then tap the ball hard enough to negate its current velocity.

The simplest model of this system treats the ball as a point moving in one dimension – namely, along the interval (l, r) – and models actuation by performing a discrete assignment to the ball's velocity[1]. Here's good start toward a dℒ formula modeling this system:

$$\texttt{Preconds} \rightarrow [\{\texttt{???}; \{x' = v \& \texttt{???}\}\}*]l \leq x \leq r$$

We already know the safety condition ($l \leq x \leq r$) and at least something about the dynamics ($x' = v$ where x is the ball's position and v is the ball's velocity). But the system's controller, domain constraint, and preconditions are all left unspecified.

The remainder of these notes consider both an event-triggered and a time-triggered controller for the system. In addition to changing the control action, we will also need to find appropraite pre-conditions and appropriate modifications to the system's evolution domain constraint.

2 The Event-triggered Controller

An **even-triggered controller** is a controller that performs an action whenever an event occurs. Events are typically chosen carefully so that they are both observable (i.e., a sensor can be used to detect the event without much delay) and useful (i.e., they help establish the pos condition).

Event-triggered systems are modeled in dℒ by interrupting the system's physical dynamics whenever an event occurs so that the controller gets to run whenever the event occurs. This is done by first describing the event using a formula (e.g., "the ball reaches the midpoint of the interval", or $x = (r + l)/2$) and then using this formula as an evolution domain constraint. By interrupting the physical dynamics and putting a $*$ around the entire system, we can ensure that the system's controller will run whenever the dynamics are about to enter a region that makes the triggers the event.

As demonstrated in this week's lectures, merely adding a condition isn't enough. The physical dynamics also have to be modified so that it is physically possible for the system to continue running (both safely or unsafely) after an event triggers. In this case, we choose our event to be the ball reaching either r or l.

$v \geq 0 \land l \leq x \leq r \land l < r \rightarrow$
[{
  $\{?x > (r + l)/2; v := -4 \ \cup \ ?x <= (r + l)/2; v := 4\}$;
  $\{x' = v \& l <= x \land x <= r\} \cup \{x' = v \& x <= l \lor x >= r\}$
}*]$(l \leq x \leq r)$

[1] A realistic model would probably model the system in at least two dimensions and would probably try to build a more realistic model of a paddle hitting a ball. But let's get comfortable with simple models first!

2.1 Two Modeling Mistakes in Event-Triggered Controllers

The most common mistakes in modeling event-triggered controllers are related to the evolution domain constraint. An event-triggered controller should always describe its dynamics in terms of two systems of differential equations with equivalent dynamics but different evolution domains. The first domain should describe the event trigger, and the second domain should describe the entire state space that is not covered by the first domain in addition. Finally, both domains should include each their shared boundaries.

From a modeling perspective, we must to include the second set of dynamics. If the second choice for dynamics is excluded then *any* controller can make the system safe – even a grotesquely unsafe controller! Take a moment to convince yourself that this formula is true even though the controller is obviously unsafe:

$v \geq 0 \land l \leq x \leq r \land l < r \rightarrow [\{\ v := 1000000;\ \{x' = v \& l <= x \land x <= r\}\ \}*] l \leq x \leq r$

We must also include the event trigger boundary in both sets of dynamics. Excluding the boundary in the second ODE allows us to make a bad velocity choice in our original model without preventing us from establishing safety properties. That's because there are regions of space where the system's description now contains no physical dynamics at all (clearly a modeling mistake):

$$
\begin{array}{l}
v \geq 0 \land l \leq x \leq r \land l < r \rightarrow \\
[\{ \\
\quad \{?x > (r+l)/2; v := 4 \ \cup \ ?x <= (r+l)/2; v := 4\}; \\
\quad \{x' = v \& l <= x \land x <= r\} \cup \{x' = v \& x < l \lor x > r\} \\
\}*](l \leq x \leq r)
\end{array}
$$

2.2 Recap and Proving Safety

Recall the model of this system that does not contain errors:

$$
\begin{array}{l}
v \geq 0 \land l \leq x \leq r \land l < r \rightarrow \\
[\{ \\
\quad \{?x > (r+l)/2; v := -4 \ \cup \ ?x <= (r+l)/2; v := 4\}; \\
\quad \{x' = v \& l <= x \land x <= r\} \cup \{x' = v \& x <= l \lor x >= r\} \\
\}*](l \leq x \leq r)
\end{array}
$$

The easiest proof of safety for this system simply solves differential equations after choosing a good loop invariant (see attached file).

3 The Time-Triggered Controller

Next, we want to see how we can make the moving point model easier to implement by switching to a time-triggered system. In a time-triggered system the environment does no longer tell us when a particular event happens. Instead, a time-triggered system checks for events at regular time intervals.

$$
\begin{array}{l}
v \geq 0 \land l \leq x \leq r \land l < r \land T > 0 \rightarrow \Big[\ \big(\ ?\big(x > \frac{r+l}{2}\big) \ ; \ v := -4; \\
\qquad\qquad\qquad\qquad\quad ?\big(x \leq \frac{r+l}{2}\big) \ ; \ v := 4; \\
\qquad\qquad\qquad\qquad\quad \mathrm{t} := 0; \\
\qquad\qquad\qquad\qquad\quad \{x' = v,\ t' = 1,\ t \leq T\} \\
\qquad\qquad\qquad\qquad\quad \big)^* @\mathrm{invariant}(l \leq x \leq r) \\
\qquad\qquad\qquad\qquad \Big](l \leq x \leq r)
\end{array}
$$

However, simply introducing time is not sufficient, since the moving point is now no longer instantaneously informed when it is about to leave the interval. We have to consider the additional distance that the point may move in the worst case until it notices that it is near the boundary. This entails that we need a sufficiently large interval; otherwise, the moving point will not be able to move anywhere.

$$v^2 = 16 \land l \leq x \leq r \land l + 8T < r \land T > 0 \rightarrow \Big[\begin{array}{l} \big(?\big(x > \frac{r+l}{2} \land v \geq 0 \land x + vT > r\big); \, v := -4; \\ \quad ?\big(x \leq \frac{r+l}{2} \land v \leq 0 \land x + vT < l\big) \; ; \, v := 4; \\ \quad \mathrm{t} := 0; \\ \quad \{x' = v, \; t' = 1, \; t \leq T\} \\ \quad \big)^* @\mathrm{invariant}(l \leq x \leq r \land v^2 = 16) \\ \Big](l \leq x \leq r) \end{array}$$