15-424/15-624 Recitation 5: Events and Delay
15-424/15-624 Foundations of Cyber-Physical Systems
Notes: Stefan Mitsch(smitsch+fcps@cs)

1. **Discussion: Why must we be careful with evolution domain constraints?**

   Let us suppose that we want to create a controller for a moving point that should stop before reaching a maximum position. The moving point can choose its velocity instantaneously.

   We begin with an event-driven architecture: our controller should be notified when the point is about to exceed the maximum point. Now consider the following controller, which is obviously unsafe.

   $$x \leq max \rightarrow \Big[\big(\text{ v} := 1000; \{x' = v,\ x \leq max\} \big)^* @\text{invariant}(x \leq max) \Big](x \leq max)$$

   The dL formula, however, is still valid, since the evolution domain constrant clips all undesired behavior just for the sake of not missing the event we are interested in. When we trust in this controller (we have proof, so it must be safe, right?) we would still end up in unsafe situations because the real world just does not respect our evolution domain constraint. This means that our evolution domain constraint is too restricted; it is not a good model of the real world.

   KeYmaera can help us detect the bug in the controller, if we model dynamics outside the event as well, as in the following dL formula.

   $$x \leq max \rightarrow \Big[\big(\text{ v} := 1000; (\{x' = v,\ x \leq max\} \cup \{x' = v,\ x \geq max\}) \big)^* @\text{invariant}(x \leq max) \Big](x \leq max)$$

   Here it is crucial to model overlapping evolution domain constraints, since otherwise we could never switch between the two models of continuous dynamics.

   We can fix the controller when we allow the point to move only when it did not yet exceed the maximum position.

   $$x \leq max \rightarrow \Big[\big(\text{ if } (x < max) \text{ then } v := 1000 \text{ else } v := 0 \text{ fi}; (\{x' = v,\ x \leq max\} \cup \{x' = v,\ x \geq max\}) \big)^* @\text{invariant}(l \leq x \leq r) \Big](l \leq x \leq r)$$

2. **Example: How can we turn an event-driven architecture into a time-triggered system?**

   Event-driven models are often easier to verify, but they are hard (if not impossible) to implement, because such models require that someone in the environment detects the event at precisely the correct time. Time-triggered systems, which observe the environment for events in recurring intervals, are often more suitable for implementation purposes.

   We can turn any event-driven architecture in a time-triggered system:

   - Remove the evolution domain constraints for instantaneous event detection
   - Introduce a clock t and a sampling interval T

- Adapt the controller: it may now take up to time T until the event is detected (solve the differential equations to find out how much safety margin you need)

$$x \leq max \wedge T > 0 \rightarrow \Big[\big(\text{ if } (x + 1000T < max) \text{ then } v := 1000 \text{ else } v := 0 \text{ fi}; \\ \text{t} := 0; \\ \{x' = v,\ t' = 1,\ t \leq T\} \\ \big)^* @\text{invariant}(x \leq max) \Big](x \leq max)$$

3. **Further Examples: Moving Point in an Interval**

   How can we create event-triggered control to keep the position of a moving point inside an interval $[l, r]$? The point is moving with constant velocity v to the right (i.e., closer to r) or constant velocity $-v$ to the left (i.e., closer to l). Our moving point is an unsteady one: if its position is closer to the right border, it moves to the left; if it's closer to the left border, the point moves to the right. How can we make the point stay within the interval?

$$v \geq 0 \wedge l \leq x \leq r \rightarrow \Big[\big(\text{ if } \big(x > \tfrac{r+l}{2}\big) \text{ then } v := -4 \\ \qquad\qquad\qquad\qquad \text{else } v := 4 \\ \text{fi}; \\ \{x' = v,\ l \leq x \leq r\} \\ \big)^* @\text{invariant}(l \leq x \leq r) \Big](l \leq x \leq r)$$

   Since the point can change its velocity instantaneously, we do not need extra space when turning around (such as for stopping with some braking force). Thus, the event we are interested in is when the moving point is about to cross either of the two interval bounds. We could simply include an evolution domain constraint that stops continuous dynamics when the point is about to leave the interval. But wait: unless the interval denotes two walls, the evolution domain constraint above cuts off perfectly realistic behavior. Remember that physics is rather non-negotiable. So, if we put in unrealistic physics models we may do just fine during the proof, but still build an unsafe system because the real world behaves differently.

   What can we do to fix our model?

$$v \geq 0 \wedge l \leq x \leq r \rightarrow \Big[\big(\text{ if } \big(x > \tfrac{r+l}{2}\big) \text{ then } v := -4 \\ \qquad\qquad\qquad\qquad \text{else } v := 4 \\ \text{fi}; \\ \{x' = v,\ l \leq x \leq r\} \ \cup \ \{x' = v,\ x \leq l \vee x \geq r\} \\ \big)^* @\text{invariant}(l \leq x \leq r) \Big](l \leq x \leq r)$$

   The only reason why we added the evolution domain constraint in the first place was that we did not want to miss the event when the moving point crossed either of the interval's boundaries. So, we can simply add alternative continuous dynamics with an evolution domain constraint that watches for interval bound crossing from the outside. The evolution domain constraints of both alternatives ensure that we can follow either of the two evolutions and still won't miss the boundary crossing. Note: It is crucial to use overlapping evolution domain constraints. Otherwise, the program could never switch between the two alternatives.

   Is this model valid? We use proof (this time with KeYmaera) to find out. We apply the proof rules of $\mathsf{d}\mathcal{L}$ to syntactically decompose our model and find a counter-example.

   What does this counter-example tell us and how can we fix the model?

$$v \geq 0 \land l \leq x \leq r \land l < r \rightarrow \Big[\big(\text{if } \big(x > \tfrac{r+l}{2}\big) \text{ then } v := -4 \text{ else } v := 4 \text{ fi}; \{x' = v,\ l \leq x \leq r\} \cup \{x' = v,\ x \leq l \lor x \geq r\} \big)^* @\text{invariant}(l \leq x \leq r) \Big](l \leq x \leq r)$$

The counter-example showed us that the moving point cannot be strictly to the right of the interval center when the interval is empty. So, we introduce the additional precondition to fix our model.

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.

$$v \geq 0 \land l \leq x \leq r \land l < r \land T > 0 \rightarrow \Big[\big(\text{if } \big(x > \tfrac{r+l}{2}\big) \text{ then } v := -4 \text{ else } v := 4 \text{ fi}; \text{t} := 0; \{x' = v,\ t' = 1,\ t \leq T\} \big)^* @\text{invariant}(l \leq x \leq r) \Big](l \leq x \leq r)$$

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.

$$v^2 = 16 \land l \leq x \leq r \land l + 8T < r \land T > 0 \rightarrow \Big[\big(\text{if } \big(x > \tfrac{r+l}{2} \land v \geq 0 \land x + vT > r\big) \text{ then } v := -4 \text{ fi}; \text{if } \big(x \leq \tfrac{r+l}{2} \land v \leq 0 \land x + vT < l\big) \text{ then } v := 4 \text{ fi}; \text{t} := 0; \{x' = v,\ t' = 1,\ t \leq T\} \big)^* @\text{invariant}(l \leq x \leq r \land v^2 = 16) \Big](l \leq x \leq r)$$

As an alternative solution, we could let the moving point choose a velocity that fits its current distance to the boundary. We also get rid of the requirement to cross the interval center to make our arithmetic a bit easier.

$$v \geq 0 \land l \leq x \leq r \land l < r \land T > 0 \rightarrow \Big[\big(\text{if } (v \geq 0 \land x + vT > r) \text{ then } v := \tfrac{l-x}{T} \text{ fi}; \text{if } (v \leq 0 \land x + vT < l) \text{ then } v := \tfrac{r-x}{T} \text{ fi}; \text{t} := 0; \{x' = v,\ t' = 1,\ t \leq T\} \big)^* @\text{invariant}(l \leq x \leq r) \Big](l \leq x \leq r)$$

Up to now our models used only deterministic choices. For proving safety it is often beneficial to allow more flexibility with nondeterministic choice.

$$v \geq 0 \land l \leq x \leq r \land l < r \land T > 0 \rightarrow \Big[\Big(\big(?v \geq 0 \land x + vT > r;\ v := \tfrac{l-x}{T}\big) \cup \big(?v \leq 0 \land x + vT < l;\ v := \tfrac{r-x}{T}\big); \mathrm{t} := 0; \{x' = v,\ t' = 1,\ t \leq T\} \Big)^* @\text{invariant}(l \leq x \leq r) \Big](l \leq x \leq r)$$

4. **Quiz**

   Suppose you have an event-triggered architecture as in the following $\mathsf{d}\mathcal{L}$ formula:

$$k \geq 0 \land x \leq max \rightarrow \Big[\Big(\big((p := 0;\ (v := 0 \cup v := 1)) \cup (?x \leq max;\ p := 1;\ v := 0)\big); \{x' = (p - v)k,\ 0 \leq x \leq max\} \Big)^* @\text{invariant}(x \leq max) \Big](x \leq max)$$

   Fix the event-driven architecture, then turn the model into a time-triggered system.