Lecture 5

Composition of Hybrid Automata

Karl Henrik Johansson

So far we have only studied autonomous, or *closed*, hybrid automata. In many situations, however, it is natural to build up a hybrid system from subsystems. It is therefore useful to have a notion of connecting hybrid automata. In this lecture we introduce *open hybrid automata*, i.e., hybrid automata with inputs and outputs. After defining executions for these, we derive a composition of two open hybrid automata. As an illustrative example, we use the car example throughout the lecture.

Example 1. For the hybrid automaton in Example 3 in Lecture 1, which describes a simple model of a car, the two control signals gear shift and throttle angle are not specified. Assume that another hybrid automaton is given that models these two signals out of the position and the speed of the car. A feedback system illustrating the connection between the two hybrid automata is shown in Figure 1. Such connection is, however, not possible in the formalism introduced so far. For instance, the hybrid automaton defined in Lecture 2 does not have a notion of external variables.

Next we introduce a slight modification of Definition 1 in Lecture 2, in order to introduce inputs and outputs for hybrid automata. To distinguish hybrid automata with and without inputs and outputs, we call the former *open hybrid automata*.

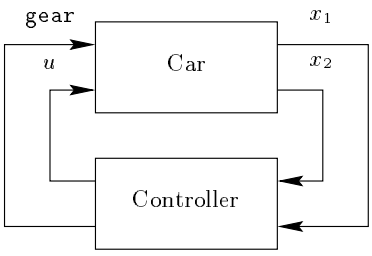

Fig. 1. Block diagram illustrating the interconnection of the car hybrid automaton and the controller hybrid automaton.

Open Hybrid Automata

Definition 1 (Open Hybrid Automaton). *An open hybrid automaton H is a collection $H = (V, X, U, Y, f, \text{Init}, \text{Inv}, \text{Jump}, h)$ with*

Discrete modes *A finite set V.*
Continuous variables *A finite set of real-valued variables $X = \{x_1, \ldots, x_n\}$, $n \geq 0$.*
Input variables *Finite sets U_D and U_C of discrete and continuous variables, respectively. The discrete variables take values in a finite set $\mathbf{U}_D$ and the continuous in $\mathbb{R}^m$, $m \geq 0$. Denote $U = U_D \cup U_C$ and $\mathbf{U} = \mathbf{U}_D \times \mathbb{R}^m$.*
Output variables *Finite sets Y_D and Y_C of discrete and continuous variables, respectively. The discrete variables take values in a finite set $\mathbf{Y}_D$ and the continuous in $\mathbb{R}^p$, $p \geq 0$. Denote $Y = Y_D \cup Y_C$ and $\mathbf{Y} = \mathbf{Y}_D \times \mathbb{R}^p$.*
Vector field *A function $f : V \times \mathbb{R}^n \times \mathbf{U} \to \mathbb{R}^n$.*
Initial condition *A set* $\text{Init} \subset V \times \mathbb{R}^n$.
Invariant condition *A set* $\text{Inv} \subset V \times \mathbb{R}^n \times \mathbf{U}$.
Jump condition *A set-valued function* $\text{Jump} : V \times \mathbb{R}^n \times \mathbf{U} \to P(V \times \mathbb{R}^n)$.
Output map *A function $h : V \times \mathbb{R}^n \to \mathbf{Y}$.*

We impose the standing assumption that both f and h are globally Lipschitz continuous.

Let us make a short comment on the notion of inputs and outputs. By defining input and output variables instead of not only general external variables, we impose a certain signal flow in the model. This simplifies often the analysis, but may be a restriction in some cases because it implies that the causality relations in the model have to be known already at the modeling instance. It is sometimes more efficient to let, for instance, an analysis tool or a computer simulation decide on that later. To see this, consider a model of a resistor. In an input–output framework we have two models of the resistor: $u := Ri$ and $i := u/R$, depending on if u and i are input or output. In a more general setting, we may model the resistor as $u = Ri$ (only specifying that u and i are external variables) and let the causality dependencies wait. Note that in this case it is sufficient to have only one model of the resistor. This idea generalizes to continuous dynamical systems. A differential equation $\dot{x} = f(v, x, u)$, $y = h(v, x)$ (as for instance defined for an open hybrid automaton) implies a signal flow from u to y. Instead of having continuous dynamics defined by an ordinary differential equation, however, it is possible to specify a differential algebraic equation $F(\dot{x}, x, v, u, y) = 0$. Here the causality needs not to be given until the equation is "solved."

The definition of an execution for an open hybrid automaton is a slight adjustment of the corresponding definition for a hybrid automaton (Definition 3 in Lecture 2).

Definition 2 (Execution of Open Hybrid Automaton). *An execution χ of an open hybrid automaton H is a collection $\chi = (\tau, v, x, u, y)$ with $\tau \in \mathcal{T}$, $v : \tau \to V$, $x : \tau \to \mathbb{R}^n$, $u : \tau \to \mathbf{U}$, and $y : \tau \to \mathbf{Y}$, satisfying*

Initial condition $\big(v(\tau_0), x(\tau_0)\big) \in \text{Init}$,

Continuous evolution *For all i with $\tau_i < \tau_i'$, $v(\cdot)$, $x(\cdot)$, $u(\cdot)$, and $y(\cdot)$ are continuous functions*[1] *over $[\tau_i, \tau_i']$, and for all $t \in [\tau_i, \tau_i')$, $x(\cdot)$ is a solution to the differential equation $\dot{x}(t) = f\big(v(t), x(t), u(t)\big)$ and $\big(v(t), x(t), u(t)\big) \in \mathrm{Inv}$.*

Discrete evolution *For all i, either $(v(\tau_{i+1}), x(\tau_{i+1})) \in \mathrm{Jump}\big(v(\tau_i'), x(\tau_i'), u(\tau_i')\big)$ or $(v(\tau_{i+1}), x(\tau_{i+1})) = \big(v(\tau_i'), x(\tau_i')\big)$.*

Output evolution *For all $t \in \tau$, $y(t) = h\big(v(t), x(t)\big)$.*

Note that a discrete transition may either take place due to a (conventional) discrete transition or due to that a transition takes place in some hybrid automaton connected to H, and, thus, possibly affecting the continuous part of the input variables of H.

Both the continuous and the discrete evolution may directly depend on both the continuous and the discrete part of the input variables. In many applications, it is natural to let the continuous part of the input variable affect only the vector field and the discrete part affect only the invariant and jump conditions.

Example 2. Consider a simplified version of the car model in Example 3 in Lecture 1. Assume that there are only two gears.[2] An open hybrid automaton $H_1 = (V_1,\ X_1,\ U_1,\ Y_1,\ f_1,\ \mathrm{Init}_1,\ \mathrm{Inv}_1,\ \mathrm{Jump}_1,\ h_1)$ modeling the car is given by

- $V_1 = \{\texttt{zero}, \texttt{low}, \texttt{high}\}$,
- $X_1 = \{x_1, x_2\}$,
- $U_{D1} = \{\texttt{gear}\}$ and $U_{C1} = \{u\}$ with $\mathbf{U} = \mathbf{U}_{D1} \times \mathbb{R}^{m_1} = \{0, 1, 2\} \times \mathbb{R}$,
- $Y_{D1} = \emptyset$ and $Y_{C1} = \{x_1, x_2\}$ with $\mathbf{Y} = \mathbb{R}^{p_1} = \mathbb{R}^2$,

and for all $(v, z, w) = \big(v, (z_1, z_2), (w_1, w_2)\big) \in V \times \mathbb{R}^2 \times \mathbf{U} = V \times \mathbb{R}^2 \times \{0, 1, 2\} \times \mathbb{R}$,

- $$f_1\big(v, (z_1, z_2), (w_1, w_2)\big) = \begin{cases} \begin{bmatrix} 0 \\ 0 \end{bmatrix}, & \text{if } v = \texttt{zero} \\ \begin{bmatrix} z_2 \\ \alpha_1(z_2) w_2 \end{bmatrix}, & \text{if } v = \texttt{low} \\ \begin{bmatrix} z_2 \\ \alpha_2(z_2) w_2 \end{bmatrix}, & \text{if } v = \texttt{high}, \end{cases}$$
- $\mathrm{Init}_1 = \{\big(\texttt{zero}, (0, 0)\big)\}$,
- $$\begin{aligned} \mathrm{Inv}_1 = \{\big(v, (z_1, z_2), (w_1, w_2)\big) \in V \times \mathbb{R}^2 \times \{1, 2\} \times \mathbb{R} : \\ (v = \texttt{zero} \wedge w_1 = 0) \vee (v = \texttt{low} \wedge w_1 = 1) \\ \vee (v = \texttt{high} \wedge w_1 = 2)\}, \end{aligned}$$

[1] Of course, a constant function (like $v(t)$ for $t \in [\tau_i, \tau_i']$) is trivially continuous.
[2] Here α_1 corresponds to α_1 in Lecture 1 and α_2 corresponds to α_4 in Lecture 1.

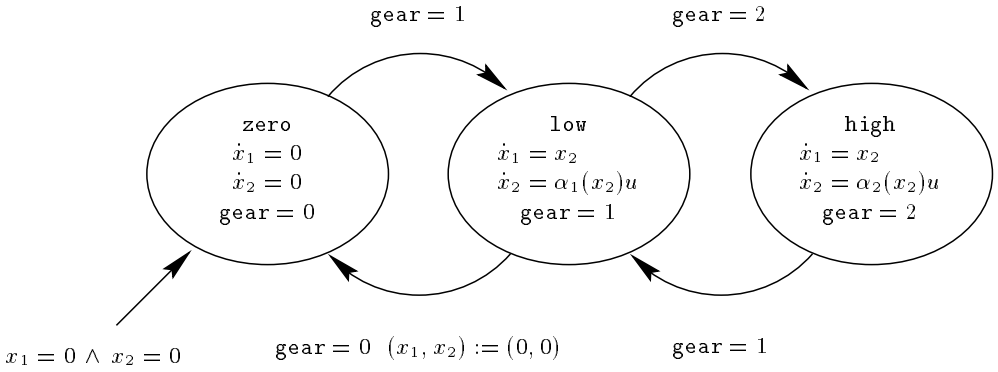

Fig. 2. Open hybrid automaton H_1 for car model. Note that the input variables $U_{D1} = \{\texttt{gear}\}$ and $U_{C1} = \{u\}$ and the output variables $Y_{C1} = \{x_1, x_2\}$ are not explicitly specified in the graph.

–

$$\mathrm{Jump}_1\big(v, z, (w_1, w_2)\big) = \begin{cases} \{(\texttt{zero}, 0)\}, & \text{if } v = \texttt{low} \wedge w_1 = 0 \\ \{(\texttt{low}, z)\}, & \text{if } (v = \texttt{high} \wedge w_1 = 1) \\ & \quad \vee (v = \texttt{zero} \wedge w_1 = 1) \\ \{(\texttt{high}, z)\}, & \text{if } v = \texttt{low} \wedge w_1 = 2 \\ \emptyset, & \text{otherwise,} \end{cases}$$

– $h_1(v, z) = z$.

The open hybrid automaton H_1 is illustrated in Figure 2.

Let the controller be designed such that it takes the car from the position $x_1 = 0$ to $x_1 = 100$. The controller has three discrete modes $\{\texttt{stop}, \texttt{gear1}, \texttt{gear2}\}$. The drive cycle is such that we wait first for 10 s in `stop`, then accelerate in `gear1` until the speed $x_2 = 20$ is reached, and, finally, continues in `gear2` until the position $x_1 = 100$ is reached and the car is stopped. Figure 3 shows the drive cycle. An open hybrid automaton $H_2 = (V_2, X_2, U_2, Y_2, f_2, \mathrm{Init}_2, \mathrm{Inv}_2, \mathrm{Jump}_2, h_2)$ modeling the controller is given by

– $V_2 = \{\texttt{stop}, \texttt{gear1}, \texttt{gear2}\}$,
– $X_2 = \{t, u\}$,
– $U_{D2} = \emptyset$ and $U_{C2} = \{x_1, x_2\}$ with $\mathbf{U} = \mathbb{R}^{m_2} = \mathbb{R}^2$,
– $Y_{D2} = \{\texttt{gear}\}$ and $Y_{C2} = \{u\}$ with $\mathbf{Y} = \mathbf{Y}_{D2} \times \mathbb{R}^{p_2} = \{0, 1, 2\} \times \mathbb{R}$,

and for all $(v, z, w) = \big(v, (z_1, z_2), w\big)\big) \in V \times \mathbb{R}^2 \times \mathbf{U} = V \times \mathbb{R}^2 \times \mathbb{R}^2$,

–

$$f_2(v, z, w) = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix}, & \text{if } v = \texttt{stop} \\ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, & \text{if } v = \texttt{gear1} \vee v = \texttt{gear2}, \end{cases}$$

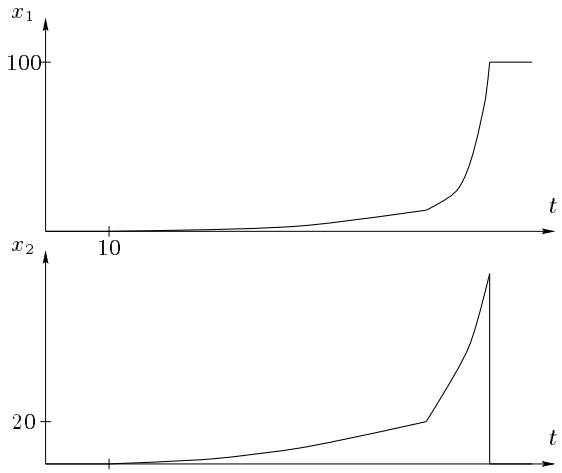

Fig. 3. Drive cycle for the car.

– $\text{Init}_2 = \{(\texttt{stop}, (0,0))\}$,

–
$$\text{Inv}_2 = \{(v, (z_1, z_2), (w_1, w_2)) \in V \times \mathbb{R}^2 \times \{1,2\} \times \mathbb{R} : (v = \texttt{stop} \wedge z_1 \leq 10) \vee (v = \texttt{gear1} \wedge w_2 \leq 20) \vee (v = \texttt{gear2} \wedge w_1 \leq 100)\}$$

–
$$\text{Jump}_2(v, (z_1, z_2), (w_1, w_2)) = \begin{cases} \{(\texttt{gear1}, (z_1, 0))\}, & \text{if } (v = \texttt{gear2} \wedge w_1 > 100) \vee \\ & \quad (v = \texttt{stop} \wedge 10 < z_1 < 100) \\ \{(\texttt{gear2}, (z_1, 0))\}, & \text{if } v = \texttt{gear1} \wedge z_1 < 100 \wedge \\ & \quad w_2 > 20 \\ \{(\texttt{stop}, (z_1, 0))\}, & \text{if } v = \texttt{gear1} \wedge w_1 > 100 \\ \emptyset, & \text{otherwise,} \end{cases}$$

–
$$h_2(v, (z_1, z_2)) = \begin{cases} (v, z_2), & \text{if } v = \texttt{gear1} \vee v = \texttt{gear2} \\ \emptyset, & \text{otherwise.} \end{cases}$$

The open hybrid automaton H_2 is illustrated in Figure 4.

Composition

Definition 3 (Compatible Open Hybrid Automata). *Two open hybrid automata* $H_1 = (V_1, X_1, U_1, Y_1, f_1, \text{Init}_1, \text{Inv}_1, \text{Jump}_1, h_1)$ *and* $H_2 = (V_2, X_2, U_2, Y_2, f_2, \text{Init}_2, \text{Inv}_2, \text{Jump}_2, h_2)$ *are called compatible if* $Y_1 \cap Y_2 = \emptyset$.

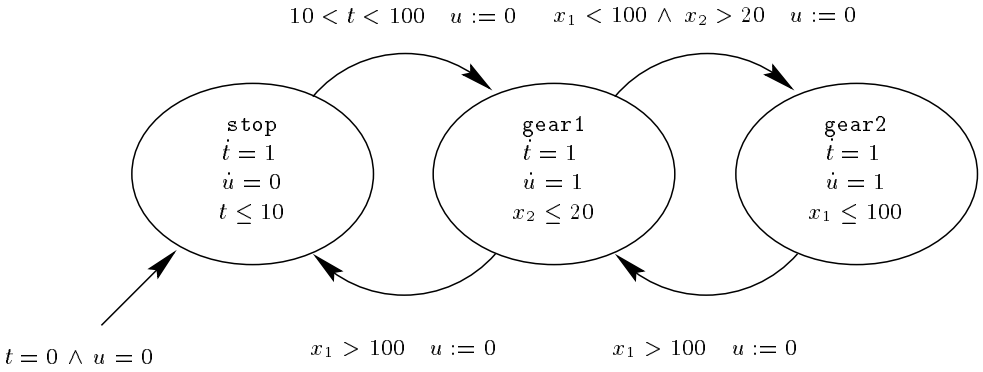

Fig. 4. Open hybrid automaton H_2 for controller. The input variables are $U_{C2} = \{x_1, x_2\}$ and the output variables $Y_{D2} = \{\texttt{gear}\}$ and $Y_{C2} = \{u\}$.

Two open hybrid automata are hence compatible if their sets of output variables are disjoint.

Next we present composition of two compatible open hybrid automata H_1 and H_2. We make the assumption that all inputs and outputs of both H_1 and H_2 are connected, i.e., $Y_1 = U_2$ and $Y_2 = U_1$. We also make the assumption that X_1 and X_2 are disjoint. None of these assumptions are restrictive, but they simplify notation considerably. The general case is straightforward and left as an exercise.

Definition 4 (Composition of Two Open Hybrid Automata). *Consider two compatible open hybrid automata* $H_1 = (V_1, X_1, U_1, Y_1, f_1, \text{Init}_1, \text{Inv}_1, \text{Jump}_1, h_1)$ *and* $H_2 = (V_2, X_2, U_2, Y_2, f_2, \text{Init}_2, \text{Inv}_2, \text{Jump}_2, h_2)$. *Assume that* $X_1 \cap X_2 = \emptyset$, $Y_1 = U_2$, *and* $Y_2 = U_1$. *The composition of* H_1 *and* H_2 *is an open hybrid automaton* $H = (V, X, U, Y, f, \text{Init}, \text{Inv}, \text{Jump}, h)$, *where*

- $V = V_1 \times V_2$,
- $X = X_1 \cup X_2$ *in* $\mathbb{R}^n$, $n = n_1 + n_2$,
- $U = (U_1 \cup U_2) \setminus (Y_1 \cup Y_2) = \emptyset$,
- $Y = Y_1 \cup Y_2$,

and, for all $\big((v_1, v_2), (z_1, z_2), (w_1, w_2)\big) \in V_1 \times V_2 \times \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \times \mathbf{U}_1 \times \mathbf{U}_2$, *with* $w_1 = h_2(v_2, z_2)$ *and* $w_2 = h_1(v_1, z_1)$, *it holds that*

- $f : V \times \mathbb{R}^n \times \mathbf{U} \to \mathbb{R}^n$ *such that*

$$f\big((v_1, v_2), (z_1, z_2), (w_1, w_2)\big) = \begin{bmatrix} f_1(v_1, z_1, w_1) \\ f_2(v_2, z_2, w_2), \end{bmatrix}$$

- $\text{Init} \subset V \times \mathbb{R}^n$ *such that*

$$\text{Init} = \{\big((v_1, v_2), (z_1, z_2)\big) : (v_1, z_1) \in \text{Init}_1 \wedge (v_2, z_2) \in \text{Init}_2\}$$

– $\mathrm{Inv} \subset V \times \mathbb{R}^n \times \mathbf{U}$ *such that*

$$\mathrm{Inv} = \{\big((v_1, v_2), (z_1, z_2), (w_1, w_2)\big) : \ (v_1, z_1, w_1) \in \mathrm{Inv}_1 \wedge (v_2, z_2, w_2) \in \mathrm{Inv}_2\}$$

– $\mathrm{Jump} : V \times \mathbb{R}^n \times \mathbf{U} \to P(V \times \mathbb{R}^n)$ *such that*

$$\big((v_1', v_2'), (z_1', z_2')\big) \in \mathrm{Jump}\big((v_1, v_2), (z_1, z_2), (w_1, w_2)\big),$$

where either $\mathrm{Jump}_1(v_1, z_1, w_1) \neq \emptyset$ *or* $\mathrm{Jump}_2(v_2, z_2, w_2) \neq \emptyset$*, and for* $i \in \{1, 2\}$, $\mathrm{Jump}_i(v_i, z_i, w_i) \neq \emptyset$ *implies* $(v_i', z_i')\mathrm{Jump}_i(v_i, z_i, w_i)$ *and else* $(v_i', z_i') = (v_i, z_i)$,

– $h : V \times \mathbb{R}^n \to \mathbf{Y}$ *such that*

$$h\big((v_1, v_2), (z_1, z_2)\big) = \begin{bmatrix} w_2 \\ w_1 \end{bmatrix}.$$

We denote the composition of H_1 and H_2 by $H = H_1 \parallel H_2$.

Roughly speaking, a connection between two open hybrid automata H_1 and H_2 is done by simply choosing the same variable names for some of the input variables of H_1 as for some of the output variables of H_2. This was done for the car example. Next we illustrate (part of) the composition in that example.

Example 3. Let us connect the car and the controller by deriving the composition of H_1 and H_2. First note that they are compatible and satisfy the assumptions of Definition 4.

– $V = V_1 \times V_2 = \{\texttt{low}, \texttt{high}\} \times \{\texttt{stop}, \texttt{gear1}, \texttt{gear2}\}$,
– $X = X_1 \cup X_2 = \{x_1, x_2, t, u\}$,
– $U = (U_1 \cup U_2) \setminus (Y_1 \cup Y_2) = \emptyset$,
– $Y = Y_1 \cup Y_2 = \{x_1, x_2, \texttt{gear}, u\}$,

and, for all $\big((v_1, v_2), (z_1, z_2), (w_1, w_2)\big) \in V_1 \times V_2 \times \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \times \mathbf{U}_1 \times \mathbf{U}_2$, with $w_1 = h_2(v_2, z_2)$ and $w_2 = h_1(v_1, z_1)$, it holds that

– $f : V \times \mathbb{R}^n \times \mathbf{U} \to \mathbb{R}^n$ such that

$$f\big((v_1, v_2), (z_1, z_2), (w_1, w_2)\big) = \begin{bmatrix} f_1(v_1, z_1, w_1) \\ f_2(v_2, z_2, w_2) \end{bmatrix},$$

where

$$f_1(v_1, z_1, w_1) = f_1\big(v_1, (z_{11}, z_{12}), (w_{11}, w_{12})\big) = \begin{cases} \begin{bmatrix} 0 \\ 0 \end{bmatrix}, & \text{if } v_1 = \texttt{zero} \\ \begin{bmatrix} z_{12} \\ \alpha_1(z_{12})\, w_{12} \end{bmatrix}, & \text{if } v_1 = \texttt{low} \\ \begin{bmatrix} z_{12} \\ \alpha_2(z_{12})\, w_{12} \end{bmatrix}, & \text{if } v_1 = \texttt{high} \end{cases}$$

and

$$f_2(v_2, z_2, w_2) = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix}, & \text{if } v_2 = \texttt{stop} \\ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, & \text{if } v_2 = \texttt{gear1} \vee v_2 = \texttt{gear2}, \end{cases}$$

- Init $= \{(\texttt{low}, \texttt{stop}, (0,0), (0,0))\}$,
- Etc.

Even if the Lectures 2–4 have been on autonomous hybrid automata, several of the concepts discussed generalize to open hybrid automata. Though, one has to be careful about the class of inputs.

Background

The lecture is mainly based on Lecture 8 in [2]. Several terms are borrowed from [1]. Note, however, that many of them have a slightly different meaning, due to that our definition of a hybrid automaton is not the same as in that paper.

References

1. R Alur and T A Henzinger. Modularity for timed and hybrid systems. In A. Mazurkiewicz and J. Winkowski, editors, *CONCUR 97: Concurrency Theory*, Lecture Notes in Computer Science 1243, pages 74–88. Springer-Verlag, 1997.
2. J. Lygeros et al. Hybrid systems: Modeling, analysis and control—eecs 291e Lecture notes and class projects. Technical Report UCB/ERL M99/34, Department of Electrical Engineering and Computer Sciences, UC Berkeley, 1999.