National School SIDRA 2017: Formal Methods for the Control of Large-scale Networked Nonlinear Systems with Logic Specifications

Lecture L3: Metric transition systems*

Abstract. In this lecture we introduce the basic material from formal methods to address control of large–scale networked nonlinear systems with logic specifications. We introduce the class of metric transition systems that we use to model continuous processes, modeling the physical part, discrete processes, modeling software and hardware in the cyber part, of Cyber–Physical Systems, and also regular languages, modeling specifications. This lecture is based on [3], [2],[4] and [1].

1 Notation

The symbols $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{R}^+$ and $\mathbb{R}_0^+$ denote the set of nonnegative integer, integer, real, positive real, and nonnegative real numbers, respectively. Given a function $f : X \to Y$ and $X' \subseteq X$ the symbol $f(X')$ denotes the image of X' through f, i.e. $f(X') = \{y \in Y | \exists x \in X' \text{ s.t. } y = f(x)\}$.

2 Metric Transition Systems

We use the mathematical paradigm of metric transition systems as a unifying framework to describe complex heterogeneous processes in Cyber–Physical Systems and logic specifications. We start with the following:

Definition 1. *[3] A transition system is a tuple*

$$T = (X, X_0, U, \longrightarrow, X_m, Y, H),$$

consisting of

- *a set of states X;*
- *a set of initial states $X_0 \subseteq X$;*
- *a set of inputs U;*
- *a transition relation $\longrightarrow \subseteq X \times U \times X$;*
- *a set of marked states $X_m \subseteq X$;*
- *a set of outputs Y and*
- *an output function $H : X \to Y$.*

Remark 1. The definition above slightly extends the notion of systems of [4] because it includes the set of marked states. A transition system enters a marked state whenever it completes some operation or task, see e.g. [1]. Marked states are also instrumental in defining regular languages in lecture no. 4.

We will follow standard practice and denote a transition $(x, u, x') \in \longrightarrow$ of T by

$$x \xrightarrow{u} x'.$$

The following definition will be useful further.

Definition 2. *Given a transition system $T = (X, X_0, U, \longrightarrow, X_m, Y, H)$,*

- *the set of active inputs $U(x)$ of state $x \in X$ is the set*

$$U(x) = \{u \in U | \exists\ x \xrightarrow{u} x'\};$$

- *the set of u–successors of state x, denoted $\mathbf{Post}_u(x)$, is the set:*

$$\mathbf{Post}_u(x) = \{x' \in X | \exists\ x \xrightarrow{u} x'\}.$$

Given a set $Z \subseteq X$, we shall abuse the notation and denote by $\mathbf{Post}_u(Z)$ the set

$$\mathbf{Post}_u(Z) = \bigcup_{x \in Z} \mathbf{Post}_u(x).$$

The evolution of transition systems is captured by the notions of state, input and output runs.

Definition 3. *(Semantics) Given a sequence of transitions of T*

$$x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} \ldots \xrightarrow{u_{l-1}} x_l \tag{1}$$

with $x_0 \in X_0$, the sequences

$$r_X : x_0\, x_1 \ldots x_l,$$
$$r_U : u_0\, u_1 \ldots u_{l-1}, \tag{2}$$
$$r_Y : H(x_0)\, H(x_1) \ldots H(x_l), \tag{3}$$

are called a state run, an input run and an output run of T, respectively.

Definition 4. *(Transition systems classification) Transition system T is said to be*

- *empty, if $X_0 = \varnothing$;*
- *countable, if X and U are countable sets;*
- *symbolic/finite, if X and U are finite sets;*
- *deterministic, if for any $x \in X$ and $u \in U(x)$ there exists a unique transition $x \xrightarrow{u} x^+$ and nondeterministic, otherwise;*
- *alive, if for any $x \xrightarrow{u} x'$ there exists $x' \xrightarrow{u'} x''$;*
- *nonblocking, if for any sequence of transitions (1) of T with $x_0 \in X_0$ either $x_l \in X_m$ or there exists a continuation of it*

  $$x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} \ldots \xrightarrow{u_{l-1}} x_l \xrightarrow{u_l} \ldots \xrightarrow{u_{l'-1}} x_{l'}$$

  such that $x_{l'} \in X_m$, and blocking, otherwise.
- *metric if Y is equipped with a metric $\mathbf{d} : Y \times Y \to \mathbb{R}_0^+$, i.e. a function*

  $$\mathbf{d} : Y \times Y \to \mathbb{R},$$

  satisfying for all $y_1, y_2, y_3 \in Y$:
  - $\mathbf{d}(y_1, y_2) \geq 0$;
  - $\mathbf{d}(y_1, y_2) = 0$ *if and only if* $y_1 = y_2$;
  - $\mathbf{d}(y_1, y_2) = \mathbf{d}(y_2, y_1)$;
  - $\mathbf{d}(y_1, y_3) \leq \mathbf{d}(y_1, y_2) + \mathbf{d}(y_2, y_3)$.

Note that:

- A transition system may be alive and blocking (consider the transition system in Fig. 1);

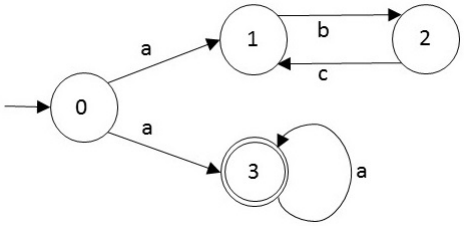

Fig. 1. A transition system that is alive and blocking. Outputs are not represented, meaning that the output function is the identity.

– A transition system may be nonblocking and not alive (consider the transition system composed only of an initial state that is also a marked state);

Metric transition systems are general enough to describe continuous processes modeling physical systems and discrete processes modeling software and hardware at the implementation level, in Cyber–Physical Systems, see [4], as illustrated in the following two examples.

Example 1. A physical process is often described by differential equations. Consider the following nonlinear control system:

$$\Sigma : \begin{cases} \dot{x}(t) = f(x(t), u(t)), \\ y(t) = h(x(t)), \\ x(t) \in \mathbf{X} \subseteq \mathbb{R}^n, \\ x(0) \in \mathbf{X}_0 \subseteq \mathbf{X}, \\ u(t) \in \mathbf{U} \subseteq \mathbb{R}^m, \\ y(t) \in \mathbf{Y} \subseteq \mathbb{R}^p, \\ t \in \mathbb{R}_0^+, \end{cases} \tag{4}$$

where $x(t)$ is the state, $u(t)$ is the input and $y(t)$ is the output at time $t \in \mathbb{R}_0^+$. Control inputs u are assumed to belong to the class $\mathcal{U}$ of piecewise continuous functions from $\mathbb{R}_0^+$ to $\mathbf{U}$. For simplicity we assume that function f and sets $\mathbf{X}$ and $\mathbf{U}$ are such that Σ admits a unique solution for any initial state $x(0) \in \mathbf{X}_0$ and for any control input function u and it is forward complete, i.e. starting from any initial state $x(0) \in \mathbf{X}_0$ and for any control input function $u \in \mathcal{U}$, the solution $\mathbf{x}(\cdot, x_0, u)$ to the differential equation (4) exists for any time $t \in \mathbb{R}_0^+$. How to model Σ through the formalism of transition systems? Given Σ define

$$T(\Sigma) = (X, X_0, U, \longrightarrow, X_m, Y, H),$$

where

– $X = \mathbf{X}$;
– $X_0 = \mathbf{X}_0$;

– U is the collection of restrictions of functions in $\mathcal{U}$ to intervals $[0, \tau[$, for some $\tau \in \mathbb{R}^+$;
– $x \xrightarrow{u|_{[0,\tau[}} x'$ if $x' = \mathbf{x}(\tau, x, u)$;
– $X_m = \mathbf{X}$;
– $Y = \mathbf{Y}$ and
– $H(x) = h(x)$ for all $x \in X$.

What are the connections between Σ and $T(\Sigma)$?

– Σ and $T(\Sigma)$ have an infinite number of states;
– Σ admits a unique solution for any initial state $x(0) \in \mathbf{X}_0$ and for any control input function u and $T(\Sigma)$ is deterministic;
– Σ is forward complete and $T(\Sigma)$ is alive;
– $T(\Sigma)$ preserves reachability properties of Σ, that is, a state x_f is reached by a trajectory of Σ starting from $\mathbf{X}_0$ if and only if there exists a state run in $T(\Sigma)$ ending in x_f.

Example 2. Any software can be rewritten in the Assembler language. For example, Matlab is written in the C language that is in turn, written in the Assembler language. Assembler language is a low-level programming language for a computer, or other programmable device, in which there is a very strong (generally one-to-one) correspondence between the language and the architecture's machine code instructions. The key operation of the Assembler language is to change the contents of the memory depending on the inputs given by the user. How to model a program in Assembler language through the formalism of transition systems? Let U be the finite set of inputs of the user, X be the finite set corresponding to all possible configurations of the memory (for example for a memory of 2 bits we have $X = \{00, 01, 10, 11\}$), let transition relation $\longrightarrow$ describe how memory contents change in the program and depending on the inputs of the user. Set X_0 as the initial memory configuration, X_m as the final memory configuration, $Y = X$ and H be the identity function. We have then defined $T = (X, X_0, U, \longrightarrow, X_m, Y, H)$. Fig. 2 illustrates a transition system describing a part of the Arithmetic Logic Unit (ALU) devoted to the calculation of the sum operation on a machine with a memory of two bits.

We also recall some unary operations on transition systems naturally adapted from the ones given for discrete–event systems [1].

Definition 5. *A transition system*

$$T' = (X', X'_0, U', \longrightarrow', X'_m, Y', H')$$

is said to be a subsystem of transition system

$$T = (X, X_0, U, \longrightarrow, X_m, Y, H),$$

denoted

$$T' \sqsubseteq T,$$

if $X' \subseteq X$, $X'_0 \subseteq X_0$, $U' \subseteq U$, $\longrightarrow' \subseteq \longrightarrow$, $X'_m \subseteq X_m$, $Y' \subseteq Y$ *and* $H'(x) = H(x)$ *for all* $x \in X'$.

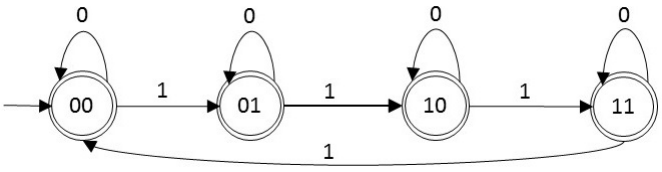

Fig. 2. Transition system describing part of the ALU devoted to the calculation of the sum operation on a machine with a memory of two bits. Outputs are not represented, meaning that the output function is the identity.

Remark 2. Binary operator $\sqsubseteq$ in the definition above is a pre–order on the set of transition systems because it enjoys the reflexivity property, i.e. $T \sqsubseteq T$ for any transition system T, and the transitivity property, i.e. $T_1 \sqsubseteq T_2$ and $T_2 \sqsubseteq T_3$ implies $T_1 \sqsubseteq T_3$ for any transition systems T_1, T_2 and T_3.

We can now give the following

Definition 6. *The accessible part of a nonempty transition system T, denoted* $\mathrm{Ac}(T)$*, is the unique maximal*[1] *subsystem T' of T such that for any state x' of T' there exists a state run of T' ending in x'. A nonempty transition system T is accessible if $T = \mathrm{Ac}(T)$.*

By definition, $\mathrm{Ac}(T)$ if not empty is accessible.

Definition 7. *The co–accessible part of a nonempty transition system T, denoted* $\mathrm{Coac}(T)$*, is the unique maximal*[2] *subsystem T' of T such that for any state $x' \in X'$ there exists a sequence of transitions of T' starting from x' and ending in a marked state of T'. A nonempty transition system T is co–accessible if $T = \mathrm{Coac}(T)$.*

By definition, $\mathrm{Coac}(T)$ if not empty is co–accessible, and therefore also nonblocking. Conversely, T may be nonblocking but not co–accessible.
We now introduce one more unary operator that is used in the next lectures to address control design with logic specifications.

Definition 8. *The trim of a nonempty transition system T, denoted* $\mathrm{Trim}(T)$*, is defined as*

$$\mathrm{Trim}(T) = \mathrm{Coac}(\mathrm{Ac}(T)) = \mathrm{Ac}(\mathrm{Coac}(T)).$$

By definition, $\mathrm{Trim}(T)$, if not empty, is accessible and co–accessible and hence, nonblocking.

[1] Here, maximality is with respect to the pre–order naturally induced by the binary operator $\sqsubseteq$, see Remark 2.

[2] Here, maximality is with respect to the pre–order naturally induced by the binary operator $\sqsubseteq$, see Remark 2.

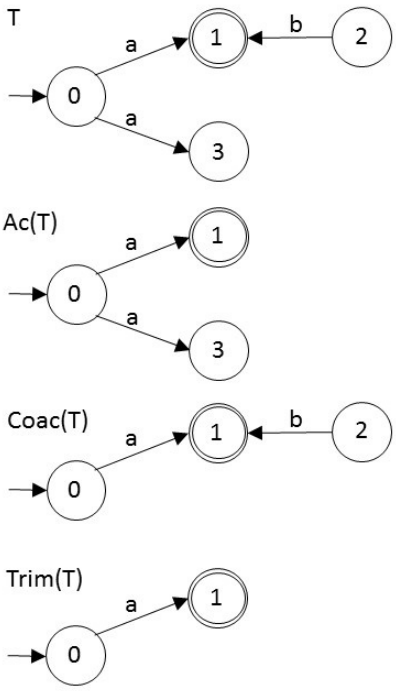

Fig. 3. A transition system T, its accessible part $\mathrm{Ac}(T)$, co–accessible part $\mathrm{Coac}(T)$, and $\mathrm{Trim}(T)$.

Example 3. In Fig. 3 we illustrate the use of the unary operators introduced above. Output function of transition system T is assumed to be the identity and therefore output symbols are not represented in Fig. 3.

References

1. C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.
2. A. Girard and G.J. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, 52(5):782–798, 2007.
3. G. Pola, P. Pepe, and M. D. Di Benedetto. Decentralized approximate supervisory control of networks of nonlinear control systems. *IEEE Transactions on Automatic Control*, 2017. Submitted for publication. Available online at arxiv.org/abs/1606.04647 [math.OC].
4. P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.