EE291E/ME290S Lecture Notes 7. Controller Synthesis for Hybrid Systems I: Introduction to Discrete Games

Claire J. Tomlin

March 17, 2021

This lecture is first in a series of lectures on designing control laws for hybrid systems. What is a control law? It is a method by which we steer the automaton using input variables, so that the closed loop exhibits certain desirable properties. A control problem involves:

1. A plant
2. A specification
3. A controller

Controller synthesis involves coming up with a methodology for designing controllers that meet the specification. The discussion in this lecture is based on [1] and [2, 3]. Here, we will deal mostly with *safety* specifications; although the methodology in this part of the course can be extended to a much more general class of properties.

1 Controlled Hybrid Automata

1.1 Fundamental Definitions

We augment the autonomous hybrid automaton definition of Lecture Notes 3 (page 4) with inputs and outputs:

Definition 1 (Hybrid Automaton) *A hybrid automaton H is a collection $H = (Q, X,$ Init, In, $f,$ Dom, $R,$ Out$)$, where*

- $Q = \{q_1, q_2, \ldots\}$ *is a set of* discrete states*;*
- $X = \mathbb{R}^n$ *is a set of* continuous states*;*
- Init $\subseteq Q \times X$ *is a set of* initial states*;*

- In *is a finite collection of* input variables. *We assume that* $\text{In} = \Sigma \cup W$*:* $\Sigma = \Sigma_U \times \Sigma_D$ *where* Σ_U *contains discrete input variables and* Σ_D *contains discrete disturbance input variables;* $W = U \cup D$ *where* U *contains continuous control input variables and* D *contains continuous disturbance input variables;*
- $f : Q \times X \times \text{In} \to \mathbb{R}^n$ *is a* vector field*;*
- $\text{Dom} : Q \to 2^{X \times \text{In}}$ *is a* domain*;*
- $R : Q \times X \times \text{In} \to 2^{Q \times X}$ *is a* reset relation*.*
- Out *is a finite collection of* output variables*. We assume that* $\text{Out} = P \cup Y$*, where* P *contains discrete output variables, and* Y *contains continuous output variables.*

We refer to $(\sigma, w) \in \text{In}$ as the *input* of H, and $(p, y) \in \text{Out}$ as the *output* of H.

The control objective is assumed to be encoded by means of a safety property $\Box F$ (always F) with $F \subseteq Q \times X$.

- The inputs are chosen by a "controller"; typically some input variables can be controlled (their values can be assigned at will by a controller), while others cannot be controlled (their values are determined by the environment);
- Uncontrollable variables (or *disturbances*) typically represent:
  - noise in the sensors, numerical computations, etc.
  - external forces, wind for the aircraft etc.
  - unmodeled dynamics
  - uncertainty about the actions of other agents (such as in the air traffic control and highway system examples)

2 Controllers

A controller can be defined as a map from state executions to sets of allowable inputs:

$$C : (\text{executions of } H) \to 2^{\Sigma_U \cup U}$$

The interpretation is that the controller restricts the valuations of the control input variables that are allowed along the trajectory. The controller may still be able to "cheat" by forcing the execution to be Zeno. Typically one assumes that all loops among discrete states require a non zero amount of time. This assumption may be somewhat restrictive and is difficult to enforce by manipulating the primitives of the model.

Memoryless Controllers

A controller, C, is called *memoryless* (or sometimes pure feedback) if for all executions of the hybrid system ending at the same state, the controller would be the same. A memoryless controllers can be characterized by a feedback map:

$$g : Q \times X \to 2^{\Sigma_U \times U}$$

Given a plant, H, and a memoryless controller, g, we can define the closed loop hybrid automaton, $H = (Q, X, \text{Init}, \text{In}, f, \text{Dom}, R, \text{Out}, g)$.

Controlled Invariant Sets

Typically, for a controller synthesis problem one treats the set of initial conditions, Init, as variable and attempts to establish the largest set of states for which there exists a controller that satisfies the specification. This set of initial conditions turns out to be a "controlled invariant set".

Definition 2 (Controlled Invariant) *A set $W \subseteq \mathbf{Q} \times \mathbf{X}$ is called controlled invariant if there exists a controller that solves the controller synthesis problem $(H', \Box W)$ where H' is identical to H except for* Init' *which is equal to W.*

A controlled invariant set W is called *maximal* if it is not a proper subset of another controlled invariant set. We say a controller *renders W invariant* if it solves the controller synthesis problem $(H', \Box W)$ where $\text{Init}' = W$.

Least Restrictive Controllers

We would like to derive a memoryless controller that solves the problem while imposing minimal restrictions on the controls it allows. There are at least two reasons why such a controller is desirable:

1. As discussed above, safety properties can sometimes be satisfied using trivial controllers (that cause deadlocks or zeno executions for example). Imposing as few restrictions as possible allows us to find a meaningful controller whenever possible.
2. In many cases multiple, prioritized specifications are given for a particular problem. Imposing fewer restrictions on the controls when designing controllers for higher priority specifications allows us greater flexibility when trying to satisfy lower priority specifications.

Memoryless controllers that solve the synthesis problem $(H, \Box F)$ can be partially ordered by the relation:

$$g_1 \preceq g_2 \Leftrightarrow g_1(x) \subseteq g_2(x) \text{ for all } x \in \mathbf{X}$$

Definition 3 *A memoryless controller that solves $(H, \Box F)$ is called* least restrictive *if it is maximal among the controllers that solve $(H, \Box F)$.*

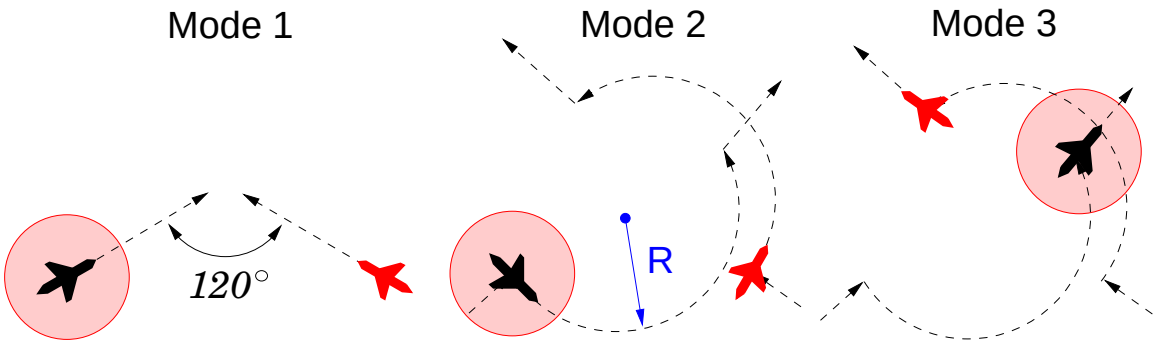

Figure 1: Two aircraft in three modes of operation: in modes 1 and 3 the aircraft follow a straight course and in mode 2 the aircraft follow a half circle. The initial relative heading (120°) is preserved throughout.

Some Remarks on Implementation

The notion of a controller introduced above may be inadequate when it comes to implementation. For one thing, the set valued map g allows non-deterministic choices of control inputs. Since in practice only one input can be applied to the system at any time, this non-determinism has to somehow be resolved when it comes time to implement such a controller. The set valued map can in this sense be thought of as a family of single valued controllers; implementation involves choosing one controller from this family.

Normally, one would implement a controller by using another hybrid automaton, which, when composed with the plant automaton yields the desired behavior.

We assume that the entire state is available to the controller. In general this will not be the case. If a controller is to be implemented by a hybrid automaton, the information the controller has about the plant is obtained through the valuations of the output variables of the plant, which are not necessarily in one to one correspondence with the valuations of the state variables. The controller synthesis problem under partial observation (output feedback) is much more complicated than the full observation (state feedback) problem addressed here and is a topic of current research.

3 Motivating Example

We present as motivating example a model for the kinematic motions of two aircraft, labeled 1 and 2, at a fixed altitude. Let $(x_r, y_r, \psi_r) \in \mathbb{R}^2 \times [-\pi, \pi)$ represent the relative position and orientation of aircraft 2 with respect to aircraft 1. In terms of the absolute positions and orientations of the two aircraft x_i, y_i, ψ_i for $i = 1, 2$, it may be verified that $x_r = \cos\psi_1(x_2 - x_1) + \sin\psi_1(y_2 - y_1), y_r = -\sin\psi_1(x_2 - x_1) + \cos\psi_1(y_2 - y_1), \psi_r = \psi_2 - \psi_1$, and

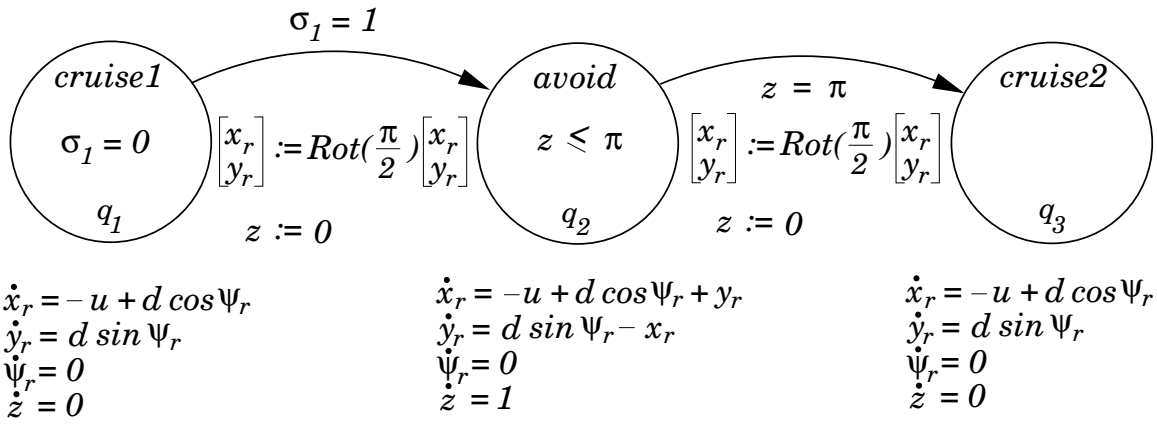

Figure 2: In q_1 both aircraft follow a straight course, in q_2 a half circle, and in q_3 both aircraft return to a straight course.

it is easy to derive that

$$\begin{aligned} \dot{x}_r &= -v_1 + v_2 \cos\psi_r + \omega_1 y_r \\ \dot{y}_r &= v_2 \sin\psi_r - \omega_1 x_r \\ \dot{\psi}_r &= \omega_2 - \omega_1 \end{aligned} \tag{1}$$

where v_i is the linear velocity of aircraft i and ω_i is its angular velocity. The protected zone of aircraft 2 may be translated to the origin of this relative frame, and thus the relative position (x_r, y_r) must remain outside of the disk $\{(x_r, y_r, \psi_r) : x_r^2 + y_r^2 < 5^2\}$. The flight modes for this system of two aircraft are based on the linear and angular velocities of the aircraft. We consider two possibilities: $\omega_i = 0$, meaning that aircraft i follows a straight line, and $\omega_i = 1$, meaning that aircraft i follows an arc of a circle if v_i is kept constant. These maneuvers approximate closely the behavior of pilots flying aircraft: straight line segments (constant heading) and arcs of circles (constant bank angle) are easy to fly both manually and on autopilot. Consider a maneuver in which there are three modes in sequence: a *cruise* mode in which both aircraft follow a straight path; an *avoid* mode in which both aircraft follow a circular arc path; and a second *cruise* mode in which the aircraft return to the straight path. The protocol of the maneuver is that as soon as the aircraft are within a certain distance of each other, each aircraft turns 90° to its right and follows a half circle. Once the half circle is complete, each aircraft returns to its original heading and continues on its straight path (Figure 1). We assume that both aircraft switch modes simultaneously, so that the relative orientation ψ_r is constant, and we assume that both aircraft fly an arc with the same radius at the same velocity. These assumptions simply allow us to display the evolution of the continuous state in two dimensions, making the results easier to present: in a true conflict resolution scenario these assumptions would be removed. This maneuver generalizes to n-aircraft as a "roundabout" maneuver, discussed in [4].

The dynamics of the maneuver can be encoded by the hybrid automaton of Figure 2, where q_1 corresponds to cruising before the avoid maneuver, q_2 corresponds to the avoid mode, and q_3 corresponds to cruising after the avoid maneuver has been completed. There is one discrete control input σ_1, such that the switch from $\sigma_1 = 0$ to $\sigma_1 = 1$ triggers the transition from q_1 to

q_2. The transition from q_2 to q_3 is required to take place after the aircraft have completed a half circle: note that with $\omega_i = 1$, for $i = 1, 2$, it takes π time units to complete a half circle. The continuous state space is augmented with a timer $z \in \mathbb{R}^+$ to force this transition. Let $x = (x_r, y_r, \psi_r, z)^T$. At each transition, both aircraft change heading instantaneously by $\pi/2$ radians; we represent this with the standard rotation matrix $Rot(\frac{\pi}{2})$. Assuming computation in the flight management system of aircraft 1, we assume that v_1 is controllable, and v_2 is known to within some uncertainty. Safety is defined in terms of the relative distance between the two aircraft:

$$F^c = G = \{q_1, q_2, q_3\} \times \{x \in X \ : \ x_r^2 + y_r^2 \leq 5^2\} \tag{2}$$

Thus the state space of this two aircraft system is $Q \times X = \{q_1, q_2, q_3\} \times (\mathbb{R}^2 \times [-\pi, \pi) \times \mathbb{R}^+)$. The discrete input space is $\Sigma = \Sigma_U = \{0, 1\}$ ($\Sigma_D = \emptyset$), and the continuous input space is $W = U \times D$, where $U = [v_{1_{min}} v_{1_{max}}]$ and $D = [v_{2_{min}} v_{2_{max}}]$. We assume Init $= q_1 \times (G^\circ)^c$, that f is described by the relative aircraft dynamics (1) augmented with a timer, as shown in Figure 2, and that Dom is given as follows:

$$\text{Dom}(q_1) = (X, \{\text{In} : \sigma_1 = 0\}); \text{Dom}(q_2) = (\{x \in X \mid 0 \leq z \leq \pi\}, \text{In}); \text{Dom}(q_3) = (X, \text{In}) \tag{3}$$

The map R which resets x_r, y_r in transitions from q_1 to q_2 and q_2 to q_3 is described in Figure 2. Output variables are the same as the state variables, in this example.

The controller synthesis problem is therefore to generate continuous control inputs for aircraft 1, as well as the conditions under which aircraft 1 is allowed to switch modes, so that that the 5 nautical mile separation is maintained, despite the disturbance actions of aircraft 2 (unknown but within known bounds).

Controllers for Safety for Discrete Systems

Before solving this hybrid control problem, let's start with something simpler: a finite automaton model of a discrete system. Recall from Lecture 2:

- **Definition:** A *finite automaton* M is a mathematical model of a system represented as

  $$(Q, \Sigma, \text{Init}, R) \tag{4}$$

  where

  - Q is a finite set of *discrete state variables*
  - $\Sigma = \Sigma_1 \cup \Sigma_2$ is a finite set of *discrete input variables*, where Σ_1 contains the controller's inputs and Σ_2 contains the environment's inputs, which cannot be controlled
  - Init $\subseteq Q$ is a set of *initial states*
  - $R : Q \times \Sigma \to 2^Q$ is a *transition relation* which maps the state and input space to subsets of the state space and thus describes the transition logic of the finite automaton

- Now consider a set of states $F \subseteq Q$.

  Problem statement: Design a control law to guarantee $\Box F(\chi)$ (where χ represents executions of the discrete system) despite possible worst case action of the disturbance inputs.

  There could be many such control laws: we would like to establish the largest set of states for which there exists a controller that manages to keep all executions inside F. This is called *least restrictive* control; its practical importance is that one can compose any other control design with this least restrictive control law, and the least restrictive control law will only come into effect when one is in danger of violating $\Box F(\chi)$.

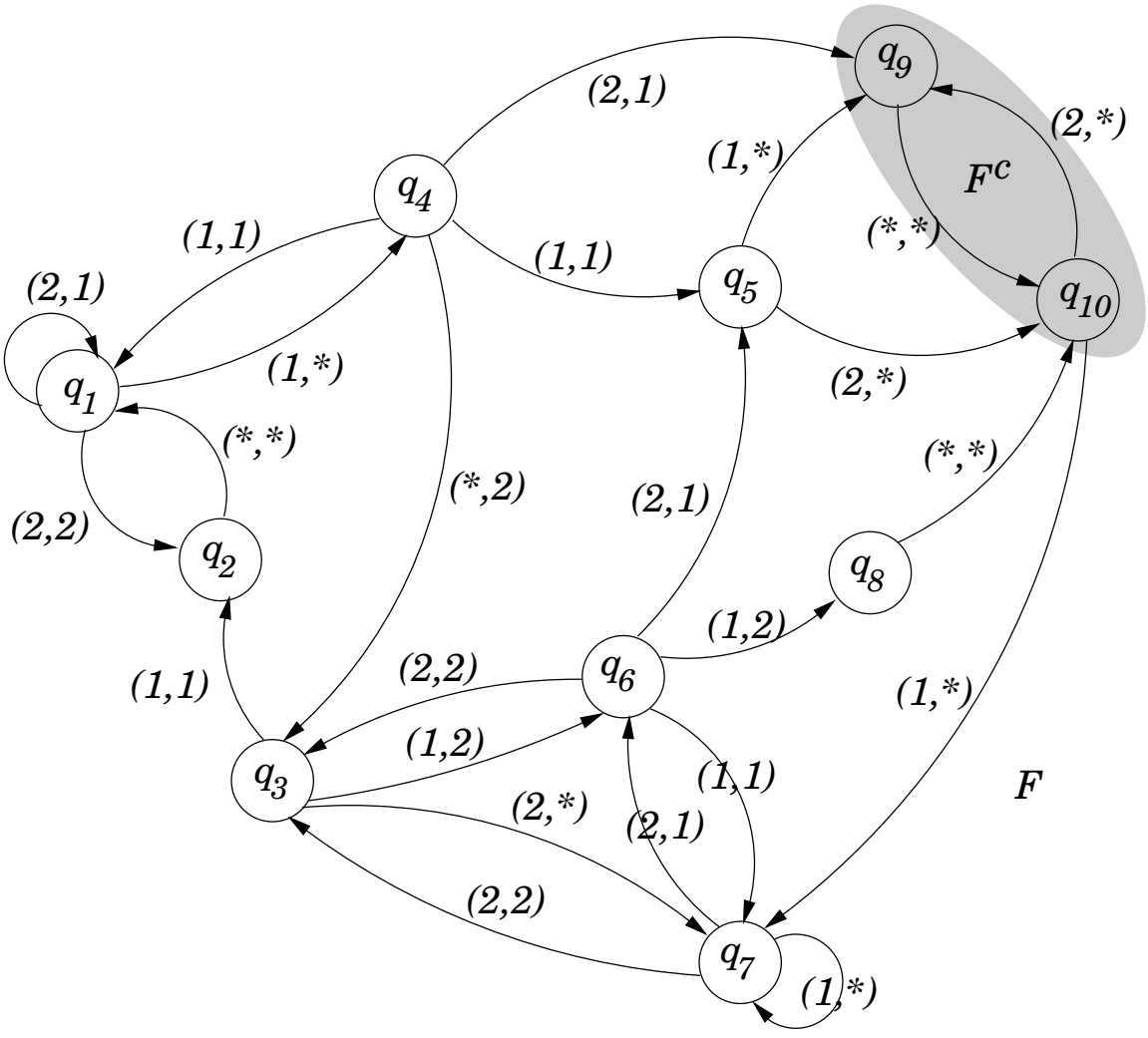

Figure 3: Example for controller synthesis for discrete systems.

- This is easiest to demonstrate by the following example, illustrated in Figure 3. Consider the finite automaton shown in Figure 3 where $Q = \{q_1, \ldots, q_{10}\}$, $\Sigma_1 = \Sigma_2 = \{1, 2\}$, $F = \{q_1, q_2, \ldots q_8\}$, and the transition relation is as shown in the figure, where the dis-

crete inputs are listed in the order (σ_1, σ_2) which represents the "order of play"[1], and $*$ denotes a "wild-card" (either 1 or 2).

- Let us denote by W^* the largest set of states such that there exists a feedback controller that keeps the execution inside F. Clearly:

$$W^* \subseteq F = \{q_1, q_2, \ldots, q_8\} \tag{5}$$

- Next, look at states that can end up in F^c in one transition: (q_4, q_5, and q_8). Now, from q_4, if $\sigma_1 = 1$ then whatever σ_2 chooses to do the system will remain in F; however from q_5 and q_8, whatever σ_1 chooses to do, the system leaves F. Thus

$$W^* \subseteq \{q_1, q_2, q_3, q_4, q_6, q_7\} \tag{6}$$

- Now, we continue to iterate backwards. Consider states that can leave the above set in one transition: q_4 and q_6. Note that from q_4, whatever σ_1 chooses to do, if $\sigma_2 = 1$ then the system leaves the set, and from q_6, whatever σ_1 chooses to do, σ_2 can play a value to force the system out of the set. Thus

$$W^* \subseteq \{q_1, q_2, q_3, q_7\} \tag{7}$$

- From these remaining states, if σ_1 chooses according to the following feedback map:

$$g(q) = \begin{cases} \{1\} & \text{if } q = q_7 \\ \{2\} & \text{if } q \in \{q_1, q_3\} \\ \{1, 2\} & \text{if } q = q_2 \end{cases} \tag{8}$$

  then the system is guaranteed to remain in $\{q_1, q_2, q_3, q_7\}$ forever.

- Thus,

$$W^* = \{q_1, q_2, q_3, q_7\} \tag{9}$$

  and g is called the *least restrictive control scheme* that renders W^* invariant.

- More generally, we may encode the above as the following algorithm:

  We define the *winning states* W^* for the controller as the subset of F from which the system has a sequence of control actions $\sigma_1(\cdot)$ which can force the system to remain in F despite the actions of the environment $\sigma_2(\cdot)$. The set W^* can be calculated as the fixed point of the following iteration (where a negative index $i \in \mathbb{Z}_-$ is used to indicate that each step is a predecessor operation):

  Algorithm 1 (Maximal Controlled Invariant Set for Finite State Automata)

[1] As we shall see, this order gives the disturbance action the benefit of knowing what the control has just played, and thus represents a conservative solution.

initialization: $W^0 = F$, $W^{-1} = \emptyset$, $i = 0$.
while $W^i \neq W^{i-1}$ **do**
$W^{i-1} = W^i \cap \{q \in Q \mid \exists \sigma_1 \in \Sigma_1 \; \forall \sigma_2 \in \Sigma_2 \; R(q, \sigma_1, \sigma_2) \subseteq W^i \}$
$i = i - 1$
end while

The iteration terminates when $W^i = W^{i-1} \triangleq W^*$. At each step of the iteration, $W^{i-1} \subseteq W^i$. Since $|Q|$ is finite the iteration terminates in a finite number of steps. The set W^i contains those states for which the controller has a sequence of actions which will ensure that the system remains in F for at least i steps, for all possible sequences $\sigma_2(\cdot) \in \Sigma_2$.

This problem can be formulated as a game.

In order to characterize this iteration mathematically, we associate a *value function* $J(q, i)$ to each state at each iteration, representing the future reward or cost to be incurred by the system given that its current state is q and iteration i:

$$J(q,i) : Q \times \mathbb{Z}_- \to \{0,1\} \text{ such that } J(q,i) = \begin{cases} 1 & q \in W^i \\ 0 & q \in (W^i)^c \end{cases} \tag{10}$$

Therefore, $W^i = \{q \in Q \mid J(q,i) = 1\}$. Since the most logical action of the controller is to keep the system inside F in the face of unknown and therefore possibly hostile actions of the environment:

$$\max_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} \min_{q' \in R(q,\sigma_1,\sigma_2)} J(q', i) = \begin{cases} 1 & \text{if } \exists \sigma_1 \in \Sigma_1 \; \forall \sigma_2 \in \Sigma_2, R(q, \sigma_1, \sigma_2) \subseteq W^i \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

The "$\min_{q' \in R(q,\sigma_1,\sigma_2)}$" in the above compensates for the nondeterminism in R; the order of operations $\max_{\sigma_1} \min_{\sigma_2}$ means that the controller *plays first*, trying to maximize the minimum value of $J(\cdot)$. This representation gives the environment the advantage, since it has "prior" knowledge of the controller's action when making its own choice. Therefore, in general,

$$\max_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} \min_{q' \in R(q,\sigma_1,\sigma_2)} J(\cdot) \leq \min_{\sigma_2 \in \Sigma_2} \max_{\sigma_1 \in \Sigma_1} \min_{q' \in R(q,\sigma_1,\sigma_2)} J(\cdot) \tag{12}$$

with equality occurring when the action (σ_1, σ_2) is a *saddle solution*, or a *no regret* solution for each player. Here, we do not need to assume the existence of a saddle solution, rather we always give advantage to the environment, the player doing its worst to drive the system out of F, in order to ensure a conservative solution. Strictly speaking this is a *Stackelberg solution* of the game with the controller as leader.

The iteration process in Algorithm 1 may be summarized by the difference equation:

$$J(q, i-1) - J(q,i) = \min\{0, \max_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} [\min_{q' \in R(q,\sigma_1,\sigma_2)} J(q', i) - J(q,i)]\} \tag{13}$$

We refer to equation (13) as a "discrete Hamilton-Jacobi equation." The first "min" in the equation ensures that states outside W^i that can be forced by the controller to transition into W^i are prevented from appearing in W^{i-1}. This means that once a state has associated to it a value of zero, the value stays at zero for all subsequent iterations: enforcing the requirement that "once a state becomes unsafe, it remains unsafe".

Proposition 4 (Winning States W^*) *For finite sets Q and Σ, a fixed point $J^*(q)$ of (13) is reached in a finite number of steps. The set of winning states for the controller is $W^* = \{q \in Q \mid J^*(q) = 1\}$. W^* is the largest controlled invariant subset of F.*

Additional References

The problem of synthesizing control laws $g : Q \to 2^{\Sigma_1}$ in the presence of uncertain actions $\sigma_2(\cdot) \in \Sigma_2^\omega$ for the finite automaton described by (4) was first posed by Church in 1962 [5], who was studying problems in digital circuit design, and was solved by Büchi and Landweber [6] and Rabin [7] in the late 1960's and early 1970's using a version of the von Neumann-Morgenstern discrete game [8]. More recently, Ramadge and Wonham [9] added new insight into the structure of the control law. A temporal logic for modeling such games is introduced in [10]. The "discrete Hamilton-Jacobi equation" to model such a process was introduced in [2].

References

[1] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, 1999.

[2] C. Tomlin, J. Lygeros, and S. Sastry. Synthesizing controllers for nonlinear hybrid systems. In T. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control*, number 1386 in LNCS, pages 360–373. Springer Verlag, New York, 1998.

[3] C. J. Tomlin, J. Lygeros, and S. Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970, July 2000.

[4] C. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management: A case study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, April 1998.

[5] A. Church. Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35. 1962.

[6] J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state operators. In *Proceedings of the American Mathematical Society*, pages 295–311, 1969.

[7] M. O. Rabin. Automata on infinite objects and Church's problem. In *Regional Conference Series in Mathematics*, 1972.

[8] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1947.

[9] P. J. G. Ramadge and W. M. Wonham. The control of discrete event dynamical systems. *Proceedings of the IEEE*, Vol.77(1):81–98, 1989.

[10] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 100–109. IEEE Computer Society Press, 1997.