in Proceedings CDC 07, Dec. 12-14, 2007, New Orleans, LA.

Partial order techniques for the analysis and synthesis of hybrid and embedded systems

Organizer: Domitilla Del Vecchio
EECS Systems Laboratory
University of Michigan, Ann Arbor

***Abstract*— The objective of this tutorial is to introduce in a tutorial fashion the employment of partial order techniques for analysis and synthesis problems in hybrid systems. While familiar to computer scientists, partial order notions may be less familiar to a control audience. The session will present fundamental notions in partial order theory including the definition of a partial order, properties of maps, and complete partial order (CPO) fix point theorems. The application of these notions to hybrid systems will be illustrated for the synthesis of safety controllers, for the design of state estimators, for robust verification, and for the analysis of the dynamics. Finally, problems in the context of intelligent transportation will be introduced and the application of partial order tools will be discussed.**

I. Controller Synthesis Using Lattice Theory

Jean-François Raskin
Computer Science Department
Université Libre de Bruxelles

A. Elements of lattice theory

Let S be a set. As usual, we note 2^S for the set of subsets of S, $\emptyset$ for the empty set, $S \times S$ for the set of pairs of elements of S, and we write $S_1 \subseteq S_2$ when the set S_1 is included in the set S_2.

A *partial-order* over S is a relation $\sqsubseteq \subseteq S \times S$ which is (i) *reflexive*, i.e. $\forall s \in S \cdot s \sqsubseteq s$; (ii) *transitive*, i.e. $\forall s_1, s_2, s_3 \in S \cdot s_1 \sqsubseteq s_2 \wedge s_2 \sqsubseteq s_3 \rightarrow s_1 \sqsubseteq s_3$; (iii) *antisymetric*, i.e. $\forall s_1, s_2 \in S \cdot s_1 \sqsubseteq s_2 \wedge s_2 \sqsubseteq s_1 \rightarrow s_1 = s_2$. We note $(S, \sqsubseteq)$, when S is partially ordered by $\sqsubseteq$. For example, let $S = \{s_1, s_2, s_3\}$, then $(2^S, \subseteq)$ is a partially ordered set.

Let $(S, \sqsubseteq)$ be a partially ordered set, let $s_1, s_2 \in S$, and let $T \subseteq S$. When $s_1 \sqsubseteq s_2$, we say that s_1 is *below* s_2 and s_2 is *above* s_1. We say that s_1 is *comparable* to s_2 whenever $s_1 \sqsubseteq s_2$ or $s_2 \sqsubseteq s_1$, there are *incomparable* otherwise. T is a *chain* if any two elements of T are comparable. T is a *antichain* is any two elements of T are incomparable. s (not necessarily in T) is a *lower-bound* for T if s is below any element of T, i.e. $\forall s' \in T \cdot s \sqsubseteq s'$. An element s (not necessarily in T) is a *upper-bound* for T if s is above any element of T, i.e. $\forall s' \in T \cdot s' \sqsubseteq s$. A lower bound for T is the *greatest lower bound* (glb) for T iff it is above any lower bound for T, it is denoted $\sqcap T$ if it exists. An upper bound for T is the *least upper bound* (lub) for T iff it is below any upper bound for T, it is denoted $\sqcup T$ if it exists. A element $s \in T$ is a *minimal* (respectively *maximal*) element of T if no other element of T is below (respectively above) s. $\mathsf{Min}(T)$ denotes the set of minimal elements of T and $\mathsf{Max}(T)$ denotes the set of maximal elements of T. If $\mathsf{Min}(T)$ is the singleton $\{s\}$ then s is called the *least* element of T. If $\mathsf{Max}(T)$ is the singleton $\{s\}$ then s is called the *greatest* element of T. We write $\mathsf{Max}_{\sqsubseteq}$ and $\mathsf{Min}_{\sqsubseteq}$ if we need to make clear the underlying order. A partially ordered set $(S, \sqsubseteq)$ is a *complete partial order*, cpo for short, iff every chain of S has a lub in S.

Example 1: Let us consider $S = \{s_1, s_2, s_3\}$. In the partially ordered set $(2^S, \subseteq)$, $\{s_1, s_3\}$ is below $\{s_1, s_2, s_3\}$ as $\{s_1, s_3\} \subseteq \{s_1, s_2, s_3\}$, $\mathsf{Max}(\{\emptyset, \{s_1\}, \{s_2, s_3\}\}) = \{\{s_1\}, \{s_2, s_3\}\}$, $\mathsf{Max}(2^S) = \{S\}$, $\mathsf{Min}(2^S) = \{\emptyset\}$, S is the greatest element of 2^S, and $\emptyset$ is the least element of 2^S. $\{\{s_1\}, \{s_1, s_3\}\}$ is a chain in 2^S and $\{\{s_1\}, \{s_2, s_3\}\}$ is an antichain.

Example 2: Let us consider $\mathcal{I}$ the set of intervals of reals included in $[0, 1]$. Let $\mathcal{I}^{\setminus 1}$ be the set of intervals of reals included in $[0, 1)$. $(I, \subseteq)$ and $(\mathcal{I}^{\setminus 1}, \subseteq)$ are partially ordered sets. It is easy to show that $(\mathcal{I}, \subseteq)$ is a complete partial order. But $(\mathcal{I}^{\setminus 1}, \subseteq)$ is not a complete partial order. Indeed, let us consider the set of intervals $\{I_0, I_1, \ldots, I_n, \ldots\}$ where I_i is the interval $[0, 1 - \frac{1}{i+2})$. Clearly, this set of intervals is a chain but as the interval $[0, 1]$ is not in $\mathcal{I}^{\setminus 1}$ there is no lub for this chain of intervals in $\mathcal{I}^{\setminus 1}$.

A complete partial order $(S, \sqsubseteq)$ is a *complete lattice* if every subset of S has a lub in S. As a direct consequence, every subset of S has also an glb in S. Indeed, the $\sqcap T = \sqcup\{s \mid s \text{ is a lower bound of } T\}$. So, in a complete lattice every subset of elements has a lub and a glb. The lub of $\emptyset$ is the least element of the set and the glb of the entire set is the greatest element of the set, those elements exist. We note $\langle S, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle$ for the complete lattice with carrier set S, partial order $\sqsubseteq$, least upper-bound operator $\sqcup$, greatest lower-bound operator $\sqcap$, least element $\bot$, greatest element $\top$.

Example 3: Let S be any set, $\langle 2^S, \subseteq, \cup, \cap, \emptyset, S \rangle$ is a complete lattice. This complete lattice is called the powerset lattice of S. The powerset lattice of $S = \{s_1, s_2, s_3\}$ is depicted in Fig. 1

A function $f : S \rightarrow S$ over a partially ordered set $(S, \sqsubseteq)$ is *monotone* iff $\forall s_1, s_2 \in S \cdot s_1 \sqsubseteq s_2 \rightarrow f(s_1) \sqsubseteq f(s_2)$.

Example 4: A transition system is a tuple (Q, q_{init}, Δ) where Q is the set of states, $q_{init} \in Q$ is the initial state, and $\Delta \subseteq Q \times Q$ is the transition relation. Fig. 1 (right) depicts a transition system whose nodes are the set of states

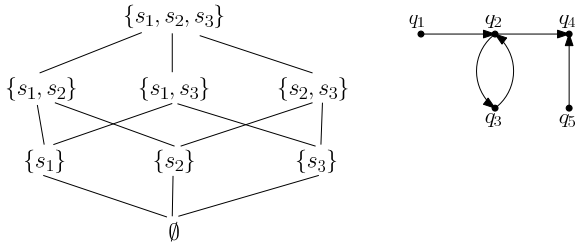

Fig. 1. The powerset lattice on the left and a transition system on the right.

$Q = \{q_1, q_2, q_3, q_4, q_5\}$, node q_1 is the initial state, and the transition relation is depicted by arrows, i.e. an arrow from a state q to a state q' denotes the fact that $(q, q') \in \Delta$. We say that q' is a *successor* of q when $(q, q') \in \Delta$. We say that a state q' is reachable from a state q if there exists a finite sequence of states $q_0 q_1 \dots q_n$ such that $q = q_0$, $q' = q_n$, and for all $0 \leq i < n$, $(q_i, q_{i+1}) \in \Delta$. The set of *reachable states* of (Q, q_{init}, Δ) is the set of states q that are reachable from q_{init}. Let us consider the partially ordered set $(2^Q, \subseteq)$, i.e. the set of sets of states of the transition system ordered by set inclusion, and let us define the function $\mathsf{post} : 2^Q \to 2^Q$ as $\mathsf{post}(T) = \{q' \mid \exists q \in T \cdot \Delta(q, q')\}$. $\mathsf{post}(T)$ is the set of states that are successors of a state in T in one step. This function is monotone over 2^Q.

For a function $f : S \to S$ over a partially ordered set $(S, \sqsubseteq)$, $s \in S$ is a fixed point iff $f(s) = s$. The set of fixed points of f, noted $\mathsf{fx}(f) = \{s \mid f(s) = s\}$. In every complete partial order, every monotone function f has a least fixed point, noted $\mathsf{lfp}(f)$ which is equal to $\sqcap \mathsf{fx}(f)$. On a complete lattice, every monotone function f has also a *greatest fixed point*, noted $\mathsf{gfp}(f)$ which is equal to $\sqcup \mathsf{fx}(f)$.

Example 5: Let us consider again the transition system depicted in Fig. 1 and the function post that we have defined in the previous example. The set of states $\{q_2, q_3, q_4\}$ is a fixed point for the function post. Now, let us consider the function $\mathsf{lpost} : 2^Q \to 2^Q$ which is defined, for any $T \subseteq S$, as: $\mathsf{lpost}(T) = \{q_1\} \cup T \cup \mathsf{post}(T)$, that is $\mathsf{lpost}(T)$ returns the initial states together with the states that are reachable in 0 or 1 step from a state in T. The set $\{q_1, q_2, q_3, q_4\}$ is a fixed point of the function lpost and furthermore it is the least one. Note that this set is exactly the set of reachable states in the transition system.

As the last example shows, least fixed points of monotone functions are interesting objects. We now defined some additional notions necessary to define iterative scheme to evaluate them. A function $f : S \to S$ over a complete partially ordered set $(S, \sqsubseteq)$ is *continuous* iff it is monotone and for all nonempty chains $T \subseteq S$, $f(\sqcup T) = \sqcup f(T)$. Clearly for finite sets, any monotone function is also continuous. Let $f : S \to S$, we define $f^0 = f$, and for any $i > 0$: $f^i = f \circ f^{i-1}$. The following theorem states that the least fixed point of a continuous function over a complete partial order corresponds to the limit of a simple iteration scheme due to Kleene:

Theorem 1: Let $(S, \sqsubseteq)$ be a cpo and $f : S \to S$ be a continuous function. Then $\mathsf{lfp}(f) = \sqcup\{f^i(\bot) \mid i \in \mathbb{N}\}$.

This theorem suggests an iteration scheme to compute the least fixed point of a function: iterate the function from the least element of the set until stabilization. This sequence is a chain which converges to the least fixed point. In any finite set, this scheme gives an algorithm which always terminates and computes the least fixed point of the function.

Example 6: Let us consider again the transition system depicted in Fig. 1, the function lpost, and apply theorem 1 to compute the least fixed point of lpost. First, remember that $\bot = \emptyset$ here, so $\mathsf{lpost}^0(\bot) = \mathsf{lpost}(\emptyset) = \{q_1\}$, $\mathsf{lpost}^1(\bot) = \mathsf{lpost}(\{q_1\}) = \{q_1, q_2\}$, $\mathsf{lpost}^2(\bot) = \mathsf{lpost}(\{q_1, q_2\}) = \{q_1, q_2, q_3, q_4\}$, $\mathsf{lpost}^3(\bot) = \mathsf{lpost}(\{q_1, q_2, q_3, q_4\}) = \{q_1, q_2, q_3, q_4\}$ which is equal to $\mathsf{lpost}^2(\bot)$, and the iteration have reached the least fixed point of lpost which is exactly the set of states reachable from q_1.

We can also formulate an iterative scheme to evaluate greatest fixed point that starts from the greatest element of the set.

Theorem 2: Let $(S, \sqsubseteq)$ be a complete lattice and $f : S \to S$ be a continuous function. Then $\mathsf{gfp}(f) = \sqcap\{f^i(\top) \mid i \in \mathbb{N}\}$.

B. Controller synthesis and games on graphs

Computer programs are more and more frequently used as controllers in embedded systems. In safety critical applications like avionics, plant control, medical instruments, etc., a high degree of reliability is necessary. To design reliable systems, methods that are based on mathematics and logic are necessary. A large research effort has been put on defining adequate models and analysis methods for embedded systems. Several international conferences on mathematical methods to reason about the correctness of embedded systems are organized each year, see for example [1], [2]. Even if the verification of complex embedded systems is still a challenge, researchers are now trying to synthesize digital controllers instead of first constructing them and afterwards proving their correctness.

a) **Framework***:* Games playing is a powerful metaphor to think about the interaction between a controller and the environment to control. Typically, the controller must maintain the system into a set of good configurations no matter how the environment behaves. To develop algorithms for controller synthesis, we use a mathematical model known as *two-player game structures* [3]. A two-player game structure $(Q, q_0, \Gamma_1, \Gamma_2, \delta)$ consists of a state space Q, an initial state q_0, two sets of moves Γ_1 and Γ_2 and a transition function $\delta : Q \times \Gamma_1 \times \Gamma_2 \to Q$. Given a set of states $y \subseteq Q$ and an move $\gamma_1 \in \Gamma_1$ for Player 1, we note $\mathsf{post}_{\gamma_1}(y)$ the set $\{q' \mid \exists q \in y \cdot \exists \gamma_2 \in \Gamma_2 \cdot q' = \delta(q, \gamma_1, \gamma_2)\}$, that is the set of states that can be reached from y when Player 1 chooses move γ_1.

Games are played on two-player game structures as follows. The game starts in the initial state q_0. In that state, each player makes a choice of move, i.e. Player 1 chooses a move $\gamma_1 \in \Gamma_1$ and Player 2 chooses a move $\gamma_2 \in \Gamma_2$, then the game evolves to state $q_1 = \delta(q_0, \gamma_1, \gamma_2)$. From q_1, a new round is played and so on. As there is no end to

that game, the result, called a *play*, is an infinite sequence of states $\rho = q_0 q_1 \dots q_n \dots$. The winner is determined by a set of winning plays $W \subseteq Q^\omega$, i.e. Player 1 is winning if $\rho \in W$ and Player 2 is winning if $\rho \in Q^\omega \setminus W$. Players are playing according to *strategies*. A strategy for Player 1 is a function $\lambda_1 : Q^* \to \Gamma_1$ that maps an prefix of a play to a choice of move for Player 1, symmetrically a Player 2 strategy is a function $\lambda_2 : Q^* \to \Gamma_2$. We say that a play $\rho = q_0 q_1 \dots q_n \dots$ is played according to strategies λ_1 and λ_2 if for all $i \geq 0$, $q_{i+1} = \delta(q_i, \lambda_1(q_0 q_1 \dots q_i), \lambda_2(q_0 q_1 \dots q_i))$. We say that ρ is the outcome of strategies λ_1 and λ_2, and we note it $\mathsf{Outcome}(\lambda_1, \lambda_2)$. Finally, we say that a strategy for Player 1 λ_1 is winning for objective W if for all strategies λ_2 of Player 2, we have that $\mathsf{Outcome}(\lambda_1, \lambda_2) \in W$.

The *strategy synthesis problem* is defined as follows. Given a two-player game structure $(Q, q_0, \Gamma_1, \Gamma_2, \delta)$, given a winning objective $W \subseteq Q^\omega$ for Player 1, decide if there exists a strategy λ_1 for Player 1 such that for all strategies λ_2 of Player 2, $\mathsf{Outcome}(\lambda_1, \lambda_2) \in W$. If such strategy exists, construct an effective representation of it. When the state space Q is finite and when the objective $W \subseteq Q^\omega$ is omega-regular, there are algorithms to solve this problem [4], [5].

b) **Iterative algorithms***:* There are elegant lattice theoretic solutions to the strategy synthesis problem, we give a summary of the main ingredients here. To keep the exposition easy, we show here how to solve finite state safety games: games where the state space Q is finite and the winning objective is defined by a subset of states $\mathsf{Good} \subseteq Q$ that Player 1 want to stay in, i.e. $W = \{q_0 q_1 \dots q_n \dots \in Q^\omega \mid \forall i \geq 0 \cdot q_i \in \mathsf{Good}\}$. If Player 1 can win such a game, we say that Player 1 has a strategy λ_1 to ensure the safety objective Good. To check for the existence of a winning strategy for Player 1 in a safety game, we consider the complete lattice $(2^Q, \subseteq, \bigcup, \bigcap, \emptyset, Q)$ and a function $\mathsf{Cpre}_1 : 2^Q \to 2^Q$ called the *Player 1 controllable predecessor operator*. Intuitively, this function given a set of states $x \subseteq Q$ returns the set of states $y \subseteq Q$ from which Player 1 can force the next state of the game to be in x. According to that intuition, the function is defined as $\mathsf{Cpre}_1(x) = \{q \in Q \mid \exists \gamma_1 \in \Gamma_1 \cdot \forall \gamma_2 \in \Gamma_2 : \delta(q, \gamma_1, \gamma_2) \in x\}$ So, a state q is controllable for the set x if Player 1 has a choice of move such that, no matter what is the choice of move of Player 2, the next state by δ is in x. To solve safety games with the Cpre_1 operator, we make the following observation. A set of states $x \subseteq Q$ is a set of winning states for Player 1 if $x \subseteq \mathsf{Good}$ and Player 1 can force x from any state of x. Such a set is a fixed point for the function $f(x) = \mathsf{Good} \cap \mathsf{Cpre}_1(x)$. Clearly, we are interested by the largest such fixed point. This is formalized in the next theorem:

Theorem 3: Given a two-player game structure $(Q, q_0, \Gamma_1, \Gamma_2, \delta)$, a set of states $\mathsf{Good} \subseteq Q$, Player 1 has a strategy λ_1 to ensure the safety objective Good iff $q_0 \in \bigcup \{x \mid x = \mathsf{Good} \cap \mathsf{Cpre}_1(x)\}$.

To evaluate this greatest fixed point, we can use the theorem 2. If we observe the iterations computed during this evaluation, we can see that it computes the sets of states from which Player 1 can ensure to stay within Good for 0 steps (i.e. Good), 1 steps (i.e. $\mathsf{Good} \cap \mathsf{Cpre}_1(\mathsf{Good})$), etc.

C. Controller synthesis with imperfect information

The algorithms for controller synthesis that we have reviewed in Sect. I-B make a strong hypothesis: they consider that the controller has a *perfect information* about the state of the system. Unfortunately, this is usually an unreasonable hypothesis. Indeed, when the control strategy has to be implemented by a real hardware, the controller typically acquires information about the state of the system by reading values on sensors. Those sensors have finite precision, and so the information about the state in which the system lies is *imperfect*.

To model imperfect information, we fix a set of observations for the system states. The control problem that we want to solve is the *safety control problem with imperfect information*: "given a set of good states Good, a set of observations, is there an observation based strategy so that the system can be prevented from entering $Q \setminus \mathsf{Good}$?". We review here a fixed point based method to solve games of imperfect information, details can be found in [6], [7].

c) **Framework***:* A *two-player game structure with imperfect information* is a tuple $(Q, q_0, \gamma_1, \gamma_2, \delta, \mathcal{O})$ where Q, q_0, γ_1, γ_2, and δ are as for plain two-player game structures, and $\mathcal{O}$ is a partition $\{O_1, O_2, \dots, O_n\}$ of Q into n *observations*. Games with imperfect information are played as follows. The game starts in state q_0 and is played in rounds. At each round, Player 1 receives the observation $O \in \mathcal{O}$ that contains the current state q, i.e. O s.t. $q \in O$, while Player 2 has perfect information on the current state of the game (for a discussion about this asymmetry, the interested reader is referred to [6]). Each player makes a choice of move, i.e. $\gamma_1 \in \Gamma_1$ and $\gamma_2 \in \Gamma_2$, and then the game evolves to the state $\delta(q, \gamma_1, \gamma_2)$. From this state, a new round is started.

As Player 1 only knows the observations of the sequence of states traversed during the game, he has to apply so-called observation based strategies. The *observation sequence* associated to a sequence of states $q_0 q_1 \dots q_n$, noted $\mathcal{O}(q_0 q_1 \dots q_n)$, is equal to $O_{f(0)} O_{f(1)} \dots O_{f(n)}$ such that for all $0 \leq i \leq n$, $q_i \in O_{f(i)}$. A *observation based strategy* for Player 1 is a function $\lambda_1^o : \mathcal{O}^* \to \Gamma_1$ that maps any prefix of a sequence of observations to a choice of move for Player 1. A strategy for player 2 is as before, i.e. a function $\lambda_2 : Q^* \to \Gamma_2$. The winner in a game with imperfect information is determined by a set of winning sequences of observations $W^o \subseteq \mathcal{O}^\omega$, called an *observable objective*. We say that a play $\rho = q_0 q_1 \dots q_n \dots$ is played according to strategies λ_1^o and λ_2 if for all $i \geq 0$, $q_{i+1} = \delta(q_i, \lambda_1^o(\mathcal{O}(q_0 q_1 \dots q_i)), \lambda_2(q_0 q_1 \dots q_i))$, ρ is called the *outcome* of the strategies λ_1^o and λ_2, Player 1 is winning if $\mathcal{O}(\rho) \in W^o$

The *strategy synthesis problem with imperfect information* is defined as follows. Given a two-player game structure with observations $(Q, q_0, \Gamma_1, \Gamma_2, \delta, \mathcal{O})$, given an observable winning condition $W^o \subseteq \mathcal{O}^\omega$ for Player 1, decide if there exists an observation based strategy λ_1^o for Player 1 such that

for all strategies λ_2 of Player 2, $\mathcal{O}(\mathsf{Outcome}(\lambda_1^o, \lambda_2)) \in W^o$. If such strategy exists, construct an effective representation of it. When the state space Q is finite and when the objective $W^o \subseteq \mathcal{O}^\omega$ is omega-regular, there are algorithms to solve this problem [6]. We review here the solution for safety objective, i.e. we are given a set of good observation $\mathsf{Good}^o \subseteq \mathcal{O}$, and we want to check that Player 1 has an observation based strategy against any strategy of Player 2 to stay within states that have observations in Good. If Player 1 can win such a game, we say that Player 1 has an observation based strategy to ensure the safety objective Good.

d) **Iterative algorithm***:* Before presenting the algorithm, we give more intuition about games of imperfect information. To better understand how Player 1 can win such games, we characterize the evolution of the *knowledge* of Player 1 during a game. The knowledge of Player 1 is the set of states in which the game can be according to what Player 1 has observed so far. Clearly, the smaller the set is, the better the knowledge is . We formalize this notion as follows. Let $G = (Q, q_0, \Gamma_1, \Gamma_2, \delta, \mathcal{O})$, let $h = O_0\gamma_0 O_1\gamma_1 \dots O_n$ be a sequence of observations and moves for Player 1 in G, h is called an *history*. We associate to the history h a set of states, noted $\mathsf{K}(G, h)$, inductively as follows. *Base case.* If h is empty then $\mathsf{K}(G, h) = \{q_0\}$. *Inductive case.* If $h = O_0\gamma_0 \dots O_{n-1}\gamma_{n-1}O_n$ and $\mathcal{K} = \mathsf{K}(G, O_0\gamma_0 O_1\gamma_1 \dots O_{n-1})$ then $\mathsf{K}(G, h) = \mathsf{post}_{\gamma_{n-1}}(\mathcal{K}) \cap O_n$.

For games of perfect information, we have shown how to elegantly solve safety games by solving a fixed point equation constructed from the so-called controllable predecessor operator Cpre_1. Remember that the operator Cpre_1 operates over sets of states of the game structure and that $\mathsf{Cpre}_1(x)$ returns the set of states from which Player 1 can force the next state to be in x. In games of imperfect information, Player 1 does not know in which states the game is but only knows that the current state is in a set of possible states (the knowledge of Player 1), so we have to construct an operator that works on set of sets of states (sets of knowledges). Let $Y = \{y_1, ..., y_n\} \subset 2^Q$ be a set of sets of states (knowledges) from which we know that Player 1 can force the game to stay for n steps within Good. We are interested to compute the set of knowledges $Y' = \{y'_1, ..., y'_m\}$ from which Player 1 can force Y in one step. Note also that if the knowledge y is sufficient for Player 1 to win then clearly any knowledge $y' \subseteq y$ is also winning (Player 1 is clearly in a better situation). It is why we define below the operator for controllable predecessors not on sets of sets of states but on antichains of (maximal) sets of states.

We note $\mathcal{Y}(Q)$ for the set of antichains of knowledges over the set of states Q, i.e. antichains of sets of states. We order two antichains of knowledges as follows: let $Y = \{y_1, y_2, \dots, y_n\}$ and $Y' = \{y'_1, y'_2, \dots, y'_m\}$, $Y \preceq Y'$ iff $\forall y \in Y \cdot \exists y' \in Y' \cdot y \subseteq y'$. The partially ordered set $(\mathcal{Y}(Q), \preceq)$ forms a complete lattice with: (i) upper-bound operator $\bigvee$ defined as follows: let $Y, Y' \in \mathcal{Y}(Q)$ then $Y \bigvee Y' = \mathsf{Max}_\subseteq(Y \cup Y')$, where $\mathsf{Max}_\subseteq(Y \cup Y')$ returns the maximal elements of $Y \cup Y'$ for set inclusion; (ii) lower-bound operator $\bigwedge$ defined as follows: let $Y, Y' \in \mathcal{Y}(Q)$ then $Y \bigwedge Y' = \mathsf{Max}_\subseteq(\{y \cap y' \mid y \in Y \wedge y' \in Y'\})$; (iii) minimal element $\emptyset$; (iv) maximal element $\{Q\}$.

Given an antichain of knowledges $Y \in \mathcal{Y}(Q)$, we define the controllable predecessor operator for the game of imperfect information $G = (Q, q_0, \Gamma_1, \Gamma_2, \delta, \mathcal{O})$ as $\mathsf{Cpre}_1^o(Y) = \mathsf{Max}_\subseteq(\{y' \subseteq Q \mid \exists \gamma_1 \in \Gamma_1 \cdot \forall O \in \mathcal{O} \cdot \exists y \in Y : \mathsf{post}_{\gamma_1}(y') \cap O \subseteq y\})$. Intuitively, given a set of knowledges Y, a knowledge $y \subseteq Q$ is controllable for Player 1 if there is a choice of move γ_1 for Player 1 that ensures the following: no matter how Player 2 plays, Player 1 knows, given the next observation, in which set of Y the next state of the game is.

This operator is used similarly as in the games of perfect information of previous section and so we have the following theorem.

Theorem 4 (in [6], [7]): Given a two-player game structure with imperfect information $(Q, q_0, \gamma_1, \gamma_2, \delta, \mathcal{O})$, a set of observations $\mathsf{Good}^o \subseteq \mathcal{O}$, Player 1 has an observation based strategies to ensure the safety objective Good^o, iff $\{\{q_0\}\} \preceq \bigvee\{Y \mid Y = \{\mathsf{Good}^o\} \bigwedge \mathsf{Cpre}_1^o(Y)\}$

II. State Estimation on a Partial Order

Domitilla Del Vecchio
EECS Systems Laboratory
University of Michigan, Ann Arbor

A. Introduction

Observability and observer design has been studied by several researchers in the hybrid systems community [8]–[17]. Due to the interaction of continuous dynamics and logic and to a potentially large number of discrete states, computational issues arise when trying to estimate the state of a hybrid system. Bemporad et al. [8] propose a deadbeat observer that requires large amounts of computation. Balluchi et al. [9] combine a *location* observer with a Luenberger observer. However, if the number of locations is large, such an approach is impracticable. Alessandri et al. propose Luenberger-like observers for hybrid systems, but the system location is known [11], [12]. In the discrete event literature, observability has been defined in [18], for example, which derives a test for current state observability. Oishi et al. [19] derive a test for immediate observability. Özveren et al. [20] and Caines [21], [22] propose discrete event observers based on the construction of the current-location observation tree that, as explored also in [23], is impractical when the number of locations is large, which is our case. In [24], an approach to state estimation is developed, which relies on a partial order structure that allows to represent sets by appropriate lower and upper bounds and thus it avoids enumeration of the locations. This approach that uses partial orders was extended to the joint estimation of continuous and discrete variables in [17] and to the presence of noise and process uncertainty in [25]. Conditions for the existence of a state estimator on a partial order are investigated in [26]. In this tutorial, we will review the basic idea behind this approach to state estimation.

B. Basic idea of a state estimator on a partial order

The idea is the one of finding an alternative way to enumeration in order to *represent* the sets of interest. Let us

consider a simple discrete state system given by $s(k+1) = f(s(k))$ for $s(k) \in \mathcal{U}$ ($\mathcal{U}$ finite) and with output measurement $y(k) = g(s(k))$ for $y(k) \in \mathcal{Y}$. Let $O_y(k) = \{s \in \mathcal{U} \mid g(s) = y(k)\}$ be the set of all possible discrete states that could give rise to the measured output. This set is also called an output set. An enumeration approach (see [21], for example) computes at each step the set of all possible current states compatible with the measurements. Let the set $\hat{s}(k) \subseteq \mathcal{U}$ represent the set of all possible states at step k that are compatible with all of the output measurements up to step k and with the system update map f. The update laws of the estimator are thus given by $\hat{s}(k+1) = f(\hat{s}(k)) \cap O_y(k+1)$. These updates are given by a prediction step given by mapping $\hat{s}(k)$ forward by means of f and by a correction step obtained by intersecting $f(\hat{s}(k))$ with a new output set $O_y(k+1)$. This approach requires (i) to compute f a number of times equal to the size of $\hat{s}(k)$; (ii) to compute the intersection with a new set, which will require a number of computations proportional to the size of $\hat{s}(k)$.

Now, assume that $s(k) \in \mathbb{N}$ and that $\hat{s}(k) = [3, 3000]$. To represent such a set it is not needed to *list* all of its elements, but it is enough to know its lower and upper bounds, that is, 3 and 3000, respectively. Assume now that $f(s(k)) = s(k)+5$. To compute $f(\hat{s}(k))$ it is not necessary to compute f on all of the elements of $\hat{s}(k)$, but it is enough to compute it on the lower and on the upper bounds of $\hat{s}(k)$. Therefore, $f(\hat{s}(k)) = [8, 3005]$. At this point, if $O_y(k+1) = [1, 200]$, it is clear that $f(\hat{s}(k)) \cap O_y(k+1) = [\sup\{8, 1\}, \inf\{3005, 200\}]$. This procedure relies on an ordering relation between elements and on order preserving properties of the map f. This reasoning can be generalized to arbitrary partial orders as it will be explained in the next section. Basically, the mapping forward of a set and set intersection can be performed by only acting on the lower and upper bounds of the sets of interest. Let $\mathcal{U} \subseteq \chi$, in which the elements of χ are related by a partial order relation "$\leq$". Let $\tilde{f}$ be the new function that is equal to f on $\mathcal{U}$ and arbitrarily chosen on the elements of χ that are not in $\mathcal{U}$. Then, if $\tilde{f}$ preserves the ordering of elements, the estimator becomes the one of Fig. 2. This approach was first proposed in [24], in which only the discrete variables were estimated, while the continuous variables were available for measurement. In [17], this approach was extended to the case in which the continuous variables also need to be estimated in systems with a cascade structure.

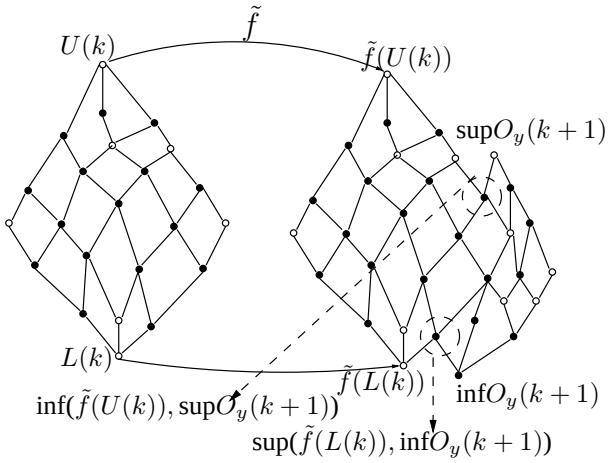

Fig. 2. Hasse diagrams of an estimator on a partial order. Any two elements that are connected by an edge are related by the order relation. The lower one is smaller than the upper one. Any two elements that are not connected by an edge are not related by the order relation. The open circles represent elements that are in χ but not in $\mathcal{U}$ while the closed circles represent the elements that are in $\mathcal{U}$. The set $\hat{s}(k)$ will be contained in the "interval" $[L(k), U(k)]$ and $f(\hat{s}(k))$ will be contained in $[\tilde{f}(L(k)), \tilde{f}(U(k))]$. The intersection with the new output set $f(\hat{s}(k)) \cap O_y(k+1)$ can be approximated by computing the lower and upper bound of such an intersection, namely, $\sup\{\tilde{f}(L(k)), \inf(O_y(k+1))\}$ and $\inf\{\tilde{f}(U(k)), \sup(O_y(k+1))\}$, respectively.

C. Properties of maps on partial orders

This section is meant to complement the section about basic partial order notions in I-A. Let $(\chi, \leq)$ be a partial order. For all $x, w \in \chi$, $\sup\{x, w\}$ is the smallest element that is larger than both x and w. In a similar way, $\inf\{x, w\}$ is the largest element that is smaller than both x and w. We define the *join* "$\curlyvee$" and the *meet* "$\curlywedge$" of two elements x and w in χ as $x \curlyvee w := \sup\{x, w\}$ and $x \curlywedge w := \inf\{x, w\}$. Also, if $S \subseteq \chi$, we have $\bigvee S := \sup\ S$, and $\bigwedge S := \inf\ S$. Let $(\chi, \leq)$ be a partial order. If $x \curlywedge w \in \chi$ and $x \curlyvee w \in \chi$ for any $x, w \in \chi$, then $(\chi, \leq)$ is a *lattice*. Let $(\chi, \leq)$ be a lattice and let $S \subseteq \chi$ be a non-empty subset of χ. Then, $(S, \leq)$ is a *sublattice* of χ if $a, b \in S$ implies that $a \curlyvee b \in S$ and $a \curlywedge b \in S$. If any sublattice of χ contains its least and greatest elements, then $(\chi, \leq)$ is called *complete*. Any finite lattice is complete, but infinite lattices may not be complete, and hence the significance of the notion of a complete partial order [27]. Given a complete lattice $(\chi, \leq)$, we will be concerned with a special kind of a sublattice called an *interval sublattice* defined as follows. Any interval sublattice of $(\chi, \leq)$ is given by $[L, U] = \{w \in \chi | L \leq w \leq U\}$ for $L, U \in \chi$. That is, this special sublattice can be represented by two elements only. For example, the interval sublattices of $(\mathbb{R}, \leq)$ are just the familiar closed intervals on the real line.

Let $(P, \leq)$ and $(Q, \leq)$ be partially ordered sets. A map $f : P \to Q$ is (i) an *order preserving map* if $x \leq w \implies f(x) \leq f(w)$; (ii) an *order embedding* if $x \leq w \iff f(x) \leq f(w)$; (iii) an *order isomorphism* if it is order embedding and it maps P *onto* Q. Every order isomorphism faithfully mirrors the structure of P onto Q. The strongest properties that we can obtain for a state estimator on a partial order are obtained when the extended update map $\tilde{f}$ is an order isomorphism. This is explained in the next section.

D. State Estimator on a Partial Order

We model a hybrid system as a deterministic transition system with both continuous and discrete variables. A *deterministic transition system* is the tuple $\Sigma = (S, \mathcal{Y}, F, g)$, where S is a set of states with $s \in S$; $\mathcal{Y}$ is a set of outputs with $y \in \mathcal{Y}$; $F : S \to S$ is the state transition function; $g : S \to \mathcal{Y}$ is the output function. An execution of Σ is any sequence $\sigma = \{s(k)\}_{k \in \mathbb{N}}$ such that $s(0) \in S$ and

$s(k+1) = F(s(k))$ for all $k \in \mathbb{N}$. An output sequence of Σ is denoted $y = \{y(k)\}_{k\in\mathbb{N}}$, with $y(k) = g(\sigma(k))$, for $\sigma \in \mathcal{E}(\Sigma)$. For system Σ, we say that two executions σ_1 and σ_2 are *distinguishable* if there exists a k such that $g(\sigma_1(k)) \neq g(\sigma_2(k))$. The notion of distinguishability is used to define the one of observability. A deterministic transition system $\Sigma = (S, \mathcal{Y}, F, g)$ is said to be *observable* if any two different executions $\sigma_1, \sigma_2 \in \mathcal{E}(\Sigma)$ are distinguishable. This means that the information of the output is enough to differentiate between different executions. As a consequence, the initial states can be distinguished by looking at the output sequence.

We now restrict our attention to transition system models describing hybrid systems. In this case, we specify a set of continuous and a set of discrete variables. In particular, for a system $\Sigma = (S, \mathcal{Y}, F, g)$ we suppose that (i) $S = \mathcal{U} \times \mathcal{Z}$ with $\mathcal{U}$ a finite set and $\mathcal{Z}$ a finite dimensional space; (ii) $F = (f, h)$, where $f : \mathcal{U} \times \mathcal{Z} \to \mathcal{U}$ and $h : \mathcal{U} \times \mathcal{Z} \to \mathcal{Z}$; (iii) $y = g(\alpha, z) := z$, where $\alpha \in \mathcal{U}$, $z \in \mathcal{Z}$, $y \in \mathcal{Y}$, and $\mathcal{Y} = \mathcal{Z}$. The set $\mathcal{U}$ is a set of logic states and $\mathcal{Z}$ is a set of measured states or physical states, such as position, velocity, temperature, etc. In the sequel, we will denote with abuse of notation this class of deterministic transition systems by $\Sigma = (\mathcal{U}, \mathcal{Z}, f, h)$, in which we associate to the tuple $(\mathcal{U}, \mathcal{Z}, f, h)$, the equations:

$$\begin{aligned} \alpha(k+1) &= f(\alpha(k), z(k)) \\ z(k+1) &= h(\alpha(k), z(k)) \\ y(k) &= z(k), \end{aligned} \tag{1}$$

in which $\alpha \in \mathcal{U}$ and $z \in \mathcal{Z}$. An execution of the system Σ in equations (1) is a sequence $\sigma = \{\alpha(k), z(k)\}_{k\in\mathbb{N}}$. The output sequence is $\{y(k)\}_{k\in\mathbb{N}} = \{z(k)\}_{k\in\mathbb{N}}$. Given an execution σ of the system Σ, we denote the α and z sequences corresponding to such an execution by $\{\sigma(k)(\alpha)\}_{k\in\mathbb{N}}$ and $\{\sigma(k)(z)\}_{k\in\mathbb{N}}$, respectively.

From the measurement of the output sequence, which in our case coincides with the evolution of the continuous variables, we want to construct a discrete state estimator: a system $\hat{\Sigma}$ that takes as input the values of the measurable variables and asymptotically tracks the value of the variable α. An alternative to simply maintaining a list of all possible values for α is proposed in [24]. Specifically, (1) we immerse the set of discrete states $\mathcal{U}$ in a larger set χ whose elements can be related by an order relation $\leq$ and (2) we extend the functions f and h to functions $\tilde{f}$ and $\tilde{h}$ that are defined on $\chi \times \mathcal{Z}$ (as opposed to being defined on $\mathcal{U} \times \mathcal{Z}$). At this point, if (a) $\tilde{h}$ is such that the set of discrete states corresponding to a pair of consecutive output measurements is an *interval* and (b) $\tilde{f}$ is an *order isomorphism* between such an interval and its image through $\tilde{f}$, we obtain a state estimator that updates only the lower and the upper bound of the set of all possible current discrete states. More formally, given a partial order $(\chi, \leq)$ with $\mathcal{U} \subseteq \chi$, we can define an extended system $\tilde{\Sigma}$ of Σ as the tuple $\tilde{\Sigma} = (\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$, in which $\tilde{f}|_{\mathcal{U}\times\mathcal{Z}} = f$ and $\tilde{h}|_{\mathcal{U}\times\mathcal{Z}} = h$. For an output sequence $\{y(k)\}_{k\in\mathbb{N}}$, the output set of $\tilde{\Sigma}$ corresponding to two consecutive outputs is given by $O_y(k) := \{w \in \chi \mid y(k+1) = \tilde{h}(w, y(k))\}$. The following theorem gives the formula for the state estimator on a partial order. Its proof can be found in [24].

Theorem 1 (in [24]): Let the deterministic transition system $\Sigma = (\mathcal{U}, \mathcal{Z}, f, h)$ be observable. Let $(\chi, \leq)$ be a lattice, such that the pair $\tilde{\Sigma}$ satisfied the two following properties

(a) Any output set $O_y(k)$ is an interval, that is, $O_y(k) = [\bigwedge O_y(k), \bigvee O_y(k)]$;
(b) $\tilde{f} : (O_y(k), z) \to [\tilde{f}(\bigwedge O_y(k), z), \tilde{f}(\bigvee O_y(k), z)]$ is an order isomorphism for all z.

Then, the following update laws

$$\begin{aligned} L(k+1) &= \tilde{f}(L(k) \curlyvee \textstyle\bigwedge O_y(k), y(k)) \\ U(k+1) &= \tilde{f}(U(k) \curlywedge \textstyle\bigvee O_y(k), y(k)) \end{aligned} \tag{2}$$

with $L(0) = \bigwedge \chi$, $U(0) = \bigvee \chi$ are such that

(i) $L(k) \leq \alpha(k) \leq U(k)$ (correctness);
(ii) $|[L(k+1), U(k+1)]| \leq |[L(k), U(k)]|$ (non-increasing error);
(iii) There exists $k_0 > 0$ such that for any $k \geq k_0$ we have $[L(k), U(k)] \cap \mathcal{U} = \alpha(k)$ (convergence).

Update laws (2) were pictorially shown in Fig. 2. We have used the notation $|X|$ to denote the cardinality of a finite set X.

Example: The robotic example described in detail in [24] and [16] involves two teams of robots competing against each other in a defensive maneuver of a robotic capture-the-flag game [28]. The state estimator on a partial order was implemented by one of the teams to efficiently estimate in real-time the current internal state describing the evolution of the strategy of the opposing team. In [29], this approach was extended to perform dynamic feedback (state estimation plus control) in order device a better team strategy.

III. Temporal Logics over Lattices and Applications

Georgios E. Fainekos and George Pappas
GRASP Laboratory
University of Pennsylvania

A. Temporal Logics

The need to utter statements whose truth value depends on the current, future and past time has led to the development of the temporal logics which are a branch of modal logics. They were first introduced in the middle of the previous century by philosophers who wanted to argue about the passage of time. In the context of computer science, temporal logics were first (very insightfully) employed by Pnueli [30] in the seventies for the analysis of distributed systems. With this formalism, we can express temporal properties like *always*, *until*, *before* and *sometimes*. For example, temporal logics can formally capture statements like "*Whenever the voltage drops below -10, then it should also eventually raise above 10*". Both the signals σ^1 and σ^2 in the plot of Fig. 3 satisfy the aforementioned specification.

The propositional logic's semantics is interpreted using the Boolean algebra over the lattice $\mathfrak{L}_2 = (\{0, 1\}, \sqcap, \sqcup)$ (see right side diagram of Fig. 3). The set of well formed

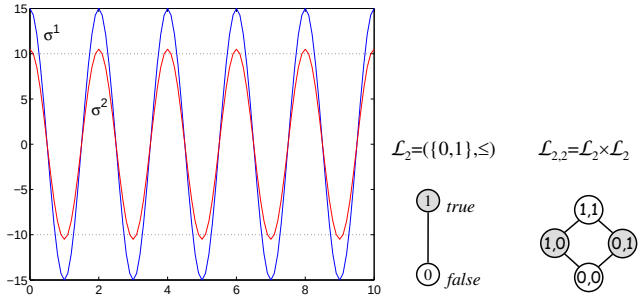

Fig. 3. Two signals σ^1 and σ^2 on the left. The Hasse diagrams of the lattices $\mathfrak{L}_2$ and $\mathfrak{L}_{2,2}$ on the right.

formulas Φ of propositional logic are build upon a set AP of atomic propositions and some binary and unary operators like *conjunction* ($\wedge$), *disjunction* ($\vee$), *negation* ($\neg$), *implication* ($\rightarrow$), and *equivalence* ($\leftrightarrow$). Temporal logics are obtained from the standard propositional logic by adding temporal operators such as *eventually* ($\Diamond$), *always* ($\Box$), *next* ($\bigcirc$), *until* ($\mathcal{U}$) and *release* ($\mathcal{R}$) in combination with the path quantifiers *for all* paths (A) and *for some* path (E).

The intuition behind these operators is as follows. $\Diamond\phi$ means that at some point in the future the formula ϕ will hold, whereas $\Box\phi$ means that ϕ should hold at every moment in the future. $\bigcirc\phi$ states that ϕ will hold at the next time moment (note that we regard time as a discrete sequence and not as continuous quantity). $\phi_1\,\mathcal{U}\phi_2$ means that ϕ_1 should be true at every time in the future until ϕ_2 becomes true. Let's consider again the voltage example mentioned above. Let the set of atomic propositions be $AP = \{a_1, a_2\}$ such that a_1 labels the set $\mathbb{R}_{\leq -10}$ and a_2 labels the set $\mathbb{R}_{\geq 10}$, then formally we can write "$A\Box(a_1 \rightarrow \Diamond a_2)$".

CTL* is a temporal logic whose syntax contains both *state* formulas ϕ_s and *path* formulas ϕ_p. Let AP be the set of atomic propositions, then the set Φ_{CTL^*} of well formed CTL* formulas is generated according to the following grammar:

$$\begin{aligned} \phi_s &::= a \mid \neg\phi_s \mid \phi_s \vee \phi_s \mid E\phi_p \mid A\phi_p \\ \phi_p &::= \phi_s \mid \neg\phi_p \mid \phi_p \vee \phi_p \mid \bigcirc \phi_p \mid \phi_p\,\mathcal{U}\phi_p \end{aligned}$$

where $a \in AP$. If ϕ_s above is replaced by $\phi_s ::= A\phi_p$, then we generate the set of all well formed LTL formulas Φ_{LTL}. Whereas, if ϕ_p is replaced by $\phi_p ::= \bigcirc\phi_p \mid \phi_p\,\mathcal{U}\phi_p$, then we generate the set of all well formed CTL formulas Φ_{CTL}. The rest of the operators can be derived from the above basic operators (see [31] for details).

B. Multi-Valued CTL Model Checking

The symbolic model checking methods [32] have been the flagship of the formal verification techniques. Here, we give a brief presentation of the extension of the classic two-valued symbolic model checking [32] to the many-valued case over quasi-Boolean algebras [33]. We proceed to define multi-valued sets, relations and Kripke structures which are going to be the foundations of the multi-valued model checking.

In the classical notion of sets, the membership of an object to a set is determined by the set's *membership* or *characteristic* function. Assume that we have a set of objects S and that we want to define a subset T of S such that every object in T satisfies a property H. Let the membership function of T be $h : S \rightarrow \{0, 1\}$ with definition: $h(s) = 1$ $(true)$ if s satisfies property H and $h(s) = 0$ $(false)$ otherwise. Then the collection of objects of S that constitute the subset T is denoted by $\{s \in S \mid h(s)\}$, which implies that for all the objects $t \in T$ it is the case that $h(t) = 1$. The multi-valued sets, denoted by mv-sets from now on, are a straightforward extension of the classical sets. The characteristic function of an mv-set takes values over a lattice instead of the classical two valued Boolean set. Intuitively, when the characteristic function is multi valued it expresses the degree that an object belongs to the mv-set.

Definition 1: Let $\mathfrak{L} = (L, \sqcap, \sqcup)$ be a lattice and S a set of objects, then a *multi-valued set*, denoted by $\mathbf{S}$, is a total function $\mathbf{S} : S \rightarrow L$.

As mv-sets are functions, $\mathbf{S}(x)$ actually denotes the degree of membership of x in $\mathbf{S}$. Next, we will define the operations of union $(\cup_{\mathfrak{L}})$, intersection $(\cap_{\mathfrak{L}})$, set inclusion $(\subseteq_{\mathfrak{L}})$ and equality $(=_{\mathfrak{L}})$ for the multi valued case using the lattice join and meet operations.

Definition 2: Let $\mathfrak{L} = (L, \sqcap, \sqcup)$ be a lattice, then:

$$\begin{aligned} (\mathbf{S} \cap_{\mathfrak{L}} \mathbf{S}')(x) &:= \mathbf{S}(x) \sqcap \mathbf{S}'(x) && \textit{(mv-intersection)} \\ (\mathbf{S} \cup_{\mathfrak{L}} \mathbf{S}')(x) &:= \mathbf{S}(x) \sqcup \mathbf{S}'(x) && \textit{(mv-union)} \\ \mathbf{S} \subseteq_{\mathfrak{L}} \mathbf{S}' &:= (\forall x).(\mathbf{S}(x) \sqsubseteq \mathbf{S}'(x)) && \textit{(mv set inclusion)} \\ \mathbf{S} =_{\mathfrak{L}} \mathbf{S}' &:= (\forall x).(\mathbf{S}(x) = \mathbf{S}'(x)) && \textit{(mv-equality)} \end{aligned}$$

Definition 3: Let $\mathfrak{B} = (B, \sqcap, \sqcup, \sim, \bot, \top)$ be an algebra, then the multi-valued set will be the total function $\mathbf{S} : S \rightarrow B$. Now, we can define the mv-set complement operation using the algebra's complementation $\sim$ operation and, also, derive the De Morgan laws:

$$\begin{aligned} \overline{\mathbf{S}}(x) &:= \sim (\mathbf{S}(x)) && \textit{(mv-complementation)} \\ \overline{\mathbf{S} \cap_{\mathfrak{B}} \mathbf{S}'} &= \overline{\mathbf{S}} \cup_{\mathfrak{B}} \overline{\mathbf{S}'} && \textit{(De-Morgan)} \\ \overline{\mathbf{S} \cup_{\mathfrak{B}} \mathbf{S}'} &= \overline{\mathbf{S}} \cap_{\mathfrak{B}} \overline{\mathbf{S}'} \\ \mathbf{S} \subseteq_{\mathfrak{B}} \mathbf{S}' &= \overline{\mathbf{S}'} \subseteq_{\mathfrak{B}} \overline{\mathbf{S}} && \textit{(antimonotonicity)} \end{aligned}$$

Note that all the above definitions actually follow the definitions for the algebraization of the classical two-valued logic. Hence, in the special case where the algebra is over the lattice $\mathfrak{L}_2$ we get the classical two valued set theory (see Theorem 2 in [33]). Now that we have established the notion of mv-sets, we proceed to define multi-valued relations (or mv-relations). Defining mv-relations is important as they are necessary for defining Kripke structures with multi-valued transition relations.

Definition 4: A *multi-valued relation* $\mathbf{R}$ on sets S and T over a lattice $\mathfrak{L}$ is a function $\mathbf{R} : S \times T \rightarrow L$.

The extension of the classical notion of Kripke structures to the multi-valued ones (mv-Kripke structures) is straightforward.

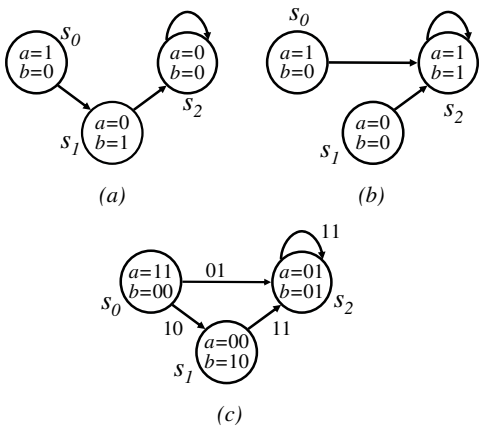

Fig. 4. Two independent experts model a real system as the Kripke structures $\mathcal{K}_1$ and $\mathcal{K}_2$. In this example, the two models are combined by simply merging the states with the same name. Also assume that the initial state of $\mathcal{M}$ is s_0. (a) The structure $\mathcal{K}_1$. (b) The structure $\mathcal{K}_2$. (c) The merged multi-valued model $\mathcal{M}$.

Definition 5: A *multi-valued Kripke structure* (mv-Kripke structure) is a tuple $\mathcal{M} = (S, S_0, \mathbf{R}, AP, \mathcal{O}, \mathfrak{B})$ where:

- S is a finite set of states
- S_0 is the set of initial states
- $\mathbf{R} : S \times S \rightarrow B$ is an *mv-transition relation*
- AP is a finite set of atomic propositions
- $\mathcal{O} : S \times AP \rightarrow B$ is a total labelling function that maps a pair $(s, a) \in S \times AP$ to some $b \in B$
- $\mathfrak{B}$ is an algebra $(L, \sqcap, \sqcup, \sim, \bot, \top)$

Also, we need to revisit the notion of predecessor states. We need to extend the definition in order to handle multi-valued sets of states. The extension seems to be straightforward, if one follows the standard definition of existential and universal quantification as introduced for the First Order predicate calculi of non-classical logics. Thus, we replace the existential quantifiers with join operations over the objects of the set and the universal quantifiers with meet operations. Note that the definition of the mv-predecessor membership function reduces to the classical one when the algebra is $\mathfrak{B}_2$ (for a proof see Theorem 3 in [33]).

Definition 6: Let $\mathbf{R} : S_1 \times S_2 \rightarrow B$ be an mv-relation defined over an algebra $(B, \sqcap, \sqcup, \sim, \bot, \top)$ and let $\mathbf{Q} : S_2 \rightarrow B$ be an mv-set, then for all $s_1 \in S_1$ we define $pre^{\mathbf{R}}_{\exists}(\mathbf{Q}) : S_1 \rightarrow B$ and $pre^{\mathbf{R}}_{\forall}(\mathbf{Q}) : S_1 \rightarrow B$ as:

$$\begin{aligned} pre^{\mathbf{R}}_{\exists}(\mathbf{Q})(s_1) &:= \sqcup_{s_2 \in S_2}(\mathbf{R}(s_1, s_2) \sqcap \mathbf{Q}(s_2)) \\ pre^{\mathbf{R}}_{\forall}(\mathbf{Q})(s_1) &:= \sqcap_{s_2 \in S_2}(\overline{\mathbf{R}}(s_1, s_2) \sqcup \mathbf{Q}(s_2)) \end{aligned}$$

Lemma 1 (in [33]): The predecessor mv-sets $pre^{\mathbf{R}}_{\exists}$ and $pre^{\mathbf{R}}_{\forall}$ are order preserving.

Example 1: In order to make clear the notion of multi-valued sets and the usage of the predecessor functions, we present an example from [33]. Assume that we have two different (classical) Kripke structures $\mathcal{K}_1$ and $\mathcal{K}_2$ that model a real system. The differences in the two structures can be attributed to the attempts of two independent experts to model the same system. We can merge the two models to get a multi-valued model $\mathcal{M}$ (Fig. 4). One can think of the resulting structure $\mathcal{M}$ as a model where we can verify properties that hold despite the possible local inconsistencies in the initial models $\mathcal{K}_1$ and $\mathcal{K}_2$. In the model $\mathcal{M}$, both the atomic propositions and the transition relation take truth values over the lattice $\mathfrak{L}_{2,2}$ (right side diagrams of Fig. 3). It can be easily proven that the lattice $\mathfrak{L}_{2,2}$ extended with a negation operator $\sim$ such that $\sim (1,1) = (0,0)$, $\sim (0,0) = (1,1)$, $\sim (1,0) = (0,1)$ and $\sim (0,1) = (1,0)$ is a Boolean algebra (as a product of two Boolean algebras). In Fig. 5, we present some multi-valued sets. Each diagram represents an mv-set that specifies the degree that each state of $\mathcal{M}$ belongs to it. Similarly in Fig. 6, we present the mv-transition relation. Finally, Fig. 7 presents the results of the computations for the predecessor mv-sets using Definition 6.

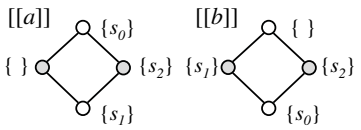

Fig. 5. The multi-valued sets $[\![a]\!]$ and $[\![b]\!]$. Here, the notation $[\![a]\!]$ with $a \in AP$ represents the multi-valued set that indicates the degree that a state s of $\mathcal{M}$ satisfies the property a.

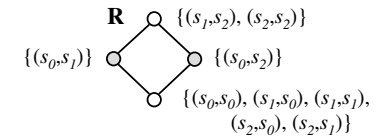

Fig. 6. The mv-transition relation $\mathbf{R}$ of $\mathcal{M}$.

Next, we define the multi-valued CTL semantics over mv-Kripke structures. In the following, we use the notation $[\![\cdot]\!]_{\mathcal{M}}$ to denote the membership functions.

Definition 7: Let the mv-membership function $[\![\phi]\!] : S \rightarrow B$ denote the degree that a state $s \in S$ satisfies a specification ϕ, then the multi-valued semantics of the core operators of mv-CTL with respect to an mv-Kripke structure $\mathcal{M} = (S, S_0, \mathbf{R}, AP, \mathcal{O}, \mathfrak{B})$ is defined as follows:

$$\begin{aligned} [\![a]\!]_{\mathcal{M}}(s) &:= \mathcal{O}(s, a) \text{ for } a \in AP \\ [\![\neg\phi]\!]_{\mathcal{M}} &:= \overline{[\![\phi]\!]_{\mathcal{M}}} \\ [\![\phi \vee \psi]\!]_{\mathcal{M}} &:= [\![\phi]\!]_{\mathcal{M}} \cup_{\mathfrak{B}} [\![\psi]\!]_{\mathcal{M}} \\ [\![E \bigcirc \phi]\!]_{\mathcal{M}} &:= pre^{\mathbf{R}}_{\exists}([\![\phi]\!]_{\mathcal{M}}) \\ [\![E\Box\phi]\!]_{\mathcal{M}} &:= \nu\mathbf{Z}.[\![\phi]\!]_{\mathcal{M}} \cap_{\mathfrak{B}} [\![EX\mathbf{Z}]\!]_{\mathcal{M}} \\ [\![E\phi\,\mathcal{U}\psi]\!]_{\mathcal{M}} &:= \mu\mathbf{Z}.[\![\psi]\!]_{\mathcal{M}} \cup_{\mathfrak{B}} ([\![\phi]\!]_{\mathcal{M}} \cap_{\mathfrak{B}} [\![E \bigcirc \mathbf{Z}]\!]_{\mathcal{M}}) \end{aligned}$$

where $[\![EX\mathbf{Z}]\!] = pre^{\mathbf{R}}_{\exists}(\mathbf{Z})$.

Finally, note that mv-CTL reduces to classical CTL when the algebra is on $\mathfrak{L}_2$ (see Theorem 4 in [33]). Now, we state a series of important theorems.

Theorem 2 (in [33]): The mv-CTL temporal operators EX and AX are order preserving.

The finiteness of the multi-valued Kripke structures and

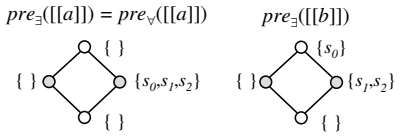

Fig. 7. Some multi-valued predecessor sets. As an example, consider the following computation: $pre_{\exists}^{\mathbf{R}}([\![a]\!])(s_1) = (\mathbf{R}(s_1,s_0) \sqcap [\![a]\!](s_0)) \sqcup (\mathbf{R}(s_1,s_1) \sqcap [\![a]\!](s_1)) \sqcup (\mathbf{R}(s_1,s_2) \sqcap [\![a]\!](s_2)) = ((0,0) \sqcap (1,1)) \sqcup ((0,0) \sqcap (0,0)) \sqcup ((1,1) \sqcap (0,1)) = (0,0) \sqcup (0,0) \sqcup (0,1) = (0,1)$. Also, note that $[\![E \bigcirc b]\!](s_0) = pre_{\exists}^{\mathbf{R}}([\![b]\!])(s_0) = (1,1)$.

the monotonicity of the predecessor functions as well as the monotonicity of the join and meet operations guarantee the termination of the fixpoint algorithm. Thus, the next result is immediate.

Theorem 3 (in [33]): The mv-CTL model checking problem is decidable.

Example 2: Consider the case where we want to verify that the property $E \bigcirc b$ is true despite the differences in the two models. As Fig. 7 indicates, the property holds on both $\mathcal{K}_1$ and $\mathcal{K}_2$ when we start from the initial state s_0. Also, assume that we want to compute the value of the specification $E\Box a$. Hence, we use the fixpoint function $[\![E\Box a]\!]_{\mathcal{M}} = \nu\mathbf{Z}.[\![a]\!]_{\mathcal{M}} \cap_{\mathfrak{B}} pre_{\exists}^{\mathbf{R}}(\mathbf{Z})$. The fixpoint algorithm is initialized by setting $(\forall s \in S).(\mathbf{Z}_0(s) = (1,1))$. After 2 iterations, the algorithm converges to $[\![E\Box a]\!]_{\mathcal{M}}(s_0) = (0,1)$ and we can conclude that the property holds only in the model that the second expert has provided ($\mathcal{K}_2$). The sequence of computations can be found in Fig. 14 in [31].

C. Robustness of LTL Formulas for Signals & Bounded Time Verification of Systems

Dynamical systems (linear, non-linear or hybrid systems) either model physical processes or the interaction between some software and/or hardware system and the continuous physical world. Fundamentally, no formal model exists that can capture accurately the behaviour of such systems. Moreover, these type of systems have a certain degree of sensitivity with respect to initial conditions or to system parameters. This has one major implication. Deciding the Boolean truth value of a temporal logic specification with respect to a system's trajectory - in some of the cases - does not allow us to draw any conclusions about the real system. A small perturbation of the trajectory or the parameters of the system can lead to a different truth value for the formula.

For example, consider again the signals σ^1 and σ^2 in Fig. 3. Both of them satisfy the same specification "*if the value of the signal drops below -10, then it should eventually raise above 10*". Nevertheless, a visual inspection of Fig. 3 indicates that there exists a qualitative difference between σ^1 and σ^2. The later "barely" satisfies the specification, i.e., a small change in the offset voltage may render the property unsatisfiable on σ^2 while σ^1 might not be affected. In order to differentiate between such trajectories of a system, the concept of robustness estimate was introduced in [34]. Informally, the robustness estimate is a bound on the perturbation that a trajectory can tolerate without changing the truth value of a specification expressed in the Linear Temporal Logic.

Let $\sigma : \mathbb{T} \to X$ be a signal (you can think of it as the output trajectory of a dynamical system), where (X,d) is a metric space and $\mathbb{T}$ is a time domain (usually $\mathbb{T} = \mathbb{N}$ or $\mathbb{T} = \mathbb{R}_{\geq 0}$). We propose multi-valued semantics for LTL where the valuation function on the atomic propositions takes values over the infinite complete lattice $\mathfrak{R} = (\mathbb{R}^{\infty}, \sqcap, \sqcup)$, where $\mathbb{R}^{\infty} = \mathbb{R} \cup \{\pm\infty\}$, according to the metric d operating on the state space X of the signal σ. For this purpose, we let the valuation function to be the signed distance from the current value of the signal to the set $\mathcal{O}(a)$ labeled by the atomic proposition a. Intuitively, this distance represents how robustly is the point $\sigma(t)$ within a set $\mathcal{O}(a)$. If this metric is zero, then even the smallest perturbation of the point can drive it inside or outside the set $\mathcal{O}(a)$, dramatically affecting membership.

Definition 8: Let $x \in X$ be a point, $S \subseteq X$ be a set and d be a metric on X. Then, the *Distance* from x to S is

$$\mathbf{dist}_d(x,S) := \inf\{d(x,y) \mid y \in S\}$$

and the *Signed Distance* from x to S is

$$\mathbf{Dist}_d(x,S) := \begin{cases} -\mathbf{dist}_d(x,S) & \text{if } x \notin S \\ \mathbf{dist}_d(x, X \backslash S) & \text{if } x \in S \end{cases}$$

Note that $\mathfrak{R}$ is actually a totally ordered set (the closure of the reals with the usual ordering relation). Recall that $\sqcup\mathbb{R}^{\infty} = +\infty$ and $\sqcap\mathbb{R}^{\infty} = -\infty$ and that any subset of $\mathbb{R}^{\infty}$ has a supremum and infimum. Finally, because $\mathfrak{R}$ is a totally ordered set, it is also distributive, i.e., for all $a,b,c \in \overline{\mathbb{R}}$ it is $a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c)$ (and its order dual).

Definition 9: For an LTL formula ϕ and a trajectory $\sigma : \mathbb{T} \to X$, the robust semantics of ϕ with respect to σ is

$$\begin{aligned} [\![a]\!](\sigma) &:= \mathbf{Dist}_d(\sigma(0), \mathcal{O}(a)) \\ [\![\neg\psi]\!](\sigma) &:= -[\![\psi]\!](\sigma) \\ [\![\phi_1 \vee \phi_2]\!](\sigma) &:= [\![\phi_1]\!](\sigma) \sqcup [\![\phi_2]\!](\sigma) \\ [\![\bigcirc\psi]\!](\sigma) &:= [\![\psi]\!](\sigma|_1) \\ [\![\phi_1 \,\mathcal{U}\, \phi_2]\!](\sigma) &:= \sqcup_{i\geq 0}\big([\![\phi_2]\!](\sigma|_i) \sqcap \sqcap_{0\leq j<i} [\![\phi_1]\!](\sigma|_j)\big) \end{aligned}$$

where $a \in AP$, $\sigma|_i(j) = \sigma(i+j)$ and $\mathcal{O} : AP \to 2^X$ maps each atomic proposition to a subset of the state space X.

The following theorem can be proven by straightforward induction on the structure of formula ϕ.

Theorem 4 (in [34]): Given $\phi \in \Phi_{LTL}$ and $\sigma : \mathbb{T} \to X$, if $[\![\phi]\!](\sigma) = \varepsilon > 0$, then for all signals $\sigma' : \mathbb{T} \to X$ such that $\sup_{i\geq 0} d(\sigma(i), \sigma'(i)) < \varepsilon$, we have $[\![\phi]\!](\sigma', t) > 0$.

Informally, the previous theorem says that if a signal satisfies an LTL specification ϕ with robustness estimate $\varepsilon > 0$, then any other signal that remains ε-close to σ will also satisfy the same LTL specification. This implies that if we can find a way to determine the neighborhood of signals that remain ε-close to σ, then we can automatically guarantee that all the signals in such a neighborhood satisfy the same specification ϕ. One way to determine neighborhoods of signals with similar behaviour is by employing approximate metrics [35]. Based on this simple observation, in [36] we presented

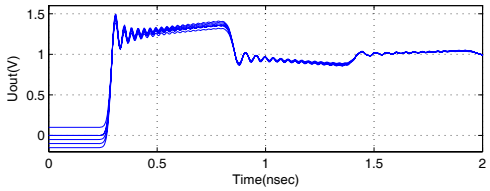

Fig. 8. The simulation trajectories of Example 3.

an algorithm for the bounded time temporal logic verification of dynamical systems. First, the algorithm samples points in the set of initial conditions of the system using guidance from the bisimulation function. Starting from this set of points, we simulate the system for a bounded horizon. For each of these trajectories we compute its robustness estimate. If the robustness estimate bounds the distance computed by the bisimulation function then we are done, otherwise we repeat the procedure. The novelty in our framework is that the number of simulations, which are required for the verification of the system, decreases inversely to the robustness of the system with respect to the temporal property. The following example from [37] indicates how the framework in [36] can be utilized in the context of bounded time verification of autonomous systems. Note that actually the algorithm in [36] can handle more complicated temporal specifications, namely Metric Temporal Logic specifications, which can reason also about the metric properties of time and not only about properties of the state space.

Example 3: The goal of this example is to check whether the transient behavior of a long transmission line is acceptable in terms of overshoot. The dynamics of the system are given by the linear dynamical system $\dot{x}(t) = Ax(t)+bU_{in}(t)$ and $U_{out}(t) = cx(t)$ where $x(t) \in \mathbb{R}^N$ is the state vector containing the voltage of the capacitors and the current of the inductors and $U_{in}(t) \in \mathbb{R}$ is the voltage at the sending end. The output of the system is the voltage $U_{out}(t) \in \mathbb{R}$ at the receiving end. Here, A, b and c are matrices of appropriate dimensions. Initially, $U_{in}(0) \in [-0.2, 0.2]$ and the system is at its steady state $x(0) = -A^{-1}bU_{in}(0)$. Then, at time $t = 0$ the input is set to the value $U_{in}(t) = 1$. We use an 81st order RLC model of the transmission line (i.e., $N = 81$). We would like to check that the amplitude of the voltage always remains bounded by θ Volts (overshoot) where $\theta \geq 0$ is a design parameter. The specification is expressed as the LTL property $\phi = \Box a$, where the predicate a is mapped to the set $\mathcal{O}(a) = [-\theta, \theta]$. When we apply the framework in [36] to this example we get the following results : (i) for $\theta = 1.4$ the property ϕ is determined to be *false* with only one simulation and (ii) for $\theta = 1.6$ the property ϕ is determined to be *true* after 5 simulations (see Fig. 8).

IV. Application of Partial Orders to Timed Concurrent Systems

Edward E. Lee
Electrical Engineering and Computer Science
UC Berkeley

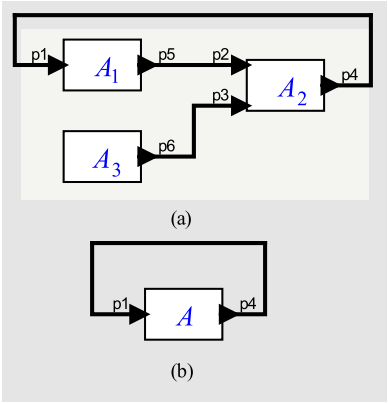

Fig. 9. A composition of three actors and an abstraction.

A. Introduction

This tutorial focuses on system models consisting of interactive concurrent components that communicate via signals. The signals are functions that map time into values. Discrete-event, continuous-time, and hybrid systems fall in this category. This tutorial shows how signals can be viewed as elements of a partially ordered set under a prefix order, and systems can be interpreted as compositions of functions over this partially ordered set. Properties of these functions, such as monotonicity, continuity, and causality, can be used to answer questions about systems, such as liveness and existence and uniqueness of solutions. For example, Zeno behavior in hybrid systems can be studied as a liveness problem. More importantly, we can give a rigorous semantics to discrete-event, continuous-time, and hybrid systems, and we can show that synchronous/reactive systems are a simple special case where time is simply a totally ordered discrete set.

B. Structure of Models

Models will be given as hierarchical networks of actors like those shown in Fig. 9. These models can represent physical components with continuous or discrete dynamics or software components interacting by sending each other time-stamped messages. In either case, we call the components "actors" and we call the communications between actors "signals." In Fig. 9(a), a program is given as a network of three actors, A_1, A_2, and A_3. The boxes represent actors, and the triangles on the boxes represent ports. The ports pointing into the boxes are input ports, which receive signals, and the ports pointing out of the boxes are output ports, which send signals. The interconnections ("wires") between actors represent interaction pathways, and carry signals. A signal is the entire (possibly infinite) history of an interaction between actors along one of these pathways. The actors themselves will be represented as functions whose domain and codomain are sets of signals.

Models are hierarchical. A set of interconnected actors can be abstracted, as illustrated in Fig. 9(b), where actor A is an abstraction of A_1, A_2, and A_3 and their interconnections.

C. Mathematical Interpretation

We model time by totally ordered set T. This could specifically be a subset of the real time, or more interestingly, super-dense time (SDT) [38], where $T = \mathbb{R}_+ \times \mathbb{N}$ with lexical ordering,

$$(r_1, n_1) \leq (r_2, n_2) \iff r_1 < r_2, \text{ or } r_1 = r_2 \text{ and } n_1 \leq n_2 .$$

SDT provides a better model than the real numbers for hybrid systems [39].

A *signal* is a partial function defined on a down set of T, where a subset T' of T is a down set if for all $t' \in T'$ and $t \in T$, $t \leq t'$ implies $t \in T'$. Down sets are also called initial segments in the literature [40].

Let $\mathcal{S}$ denote the set of all signals with tag set T. That is, this is the set of partial functions with domain T and codomain V that are defined on a down set of T. $\mathcal{S}$ is a partially ordered set (poset) under the **prefix order**, defined next. For any $s_1, s_2 \in \mathcal{S}$, s_1 is a prefix of s_2, denoted by $s_1 \sqsubseteq s_2$, if and only if s_2 is defined everywhere that s_1 is defined and $s_1(t) = s_2(t)$, $\forall t$ where s_1 is defined. The prefix order on signals is a natural generalization of the prefix order on strings or sequences, and the extension order on partial functions [41].

A **complete partial order** (CPO) $(P, \leq)$ is a poset where P has least element $\bot_P \in P$, and where every directed subset of P has a least upper bound. A subset $D \subseteq P$ is directed if for all $d_1, d_2 \in D$, $\{d_1, d_2\}$ has an upper bound in D.

A signal set with the prefix order $(\mathcal{S}, \sqsubseteq)$ is a CPO [42]. The least element of $\mathcal{S}$ is $s_\bot : \emptyset \to V$, an **empty signal** (it has no events). If a signal is defined for all tags in T, then it is a maximal element of $\mathcal{S}$, and is called a **total signal**.

Actors will be modeled as functions from input signals to output signals. For example, in Fig. 9(b), actor A is given by a function $F_A : \mathcal{S} \to \mathcal{S}$, where the function will be defined in terms of the corresponding functions for actors A_1, A_2, and A_3. Because of the feedback structure in Fig. 9(b), the behavior of this model will be given as a fixed point of the function F_A.

This tutorial will study the properties of such fixed points that emerge from the mathematical model, reviewing and leveraging classical results on partial orders [43]. In particular, monotonicity and continuity of functions turns out to be central. Let $(D, \sqsubseteq)$ and $(E, \sqsubseteq)$ be CPOs. A function $G : D \to E$ is **monotonic** if it is order-preserving, $\forall d_1, d_2 \in D,\ d_1 \sqsubseteq d_2 \implies G(d_1) \sqsubseteq G(d_2)$. The same function is (Scott) **continuous** if for all directed sets $D' \subseteq D$, $G(D')$ is a directed set and $G(\bigvee D') = \bigvee G(D')$. Here, $G(D')$ is defined in the natural way as $\{G(d) \mid d \in D'\}$, and $\vee X$ denotes the least upper bound of the set X.

It is easy to show that every continuous function is monotonic. A classic fixed point theorem [43] states that if $G : D \to D$ for CPO D is continuous, then it has a least fixed point, and that least fixed point is

$$\bigvee \{G^n(\bot_D) \mid n \in \mathbb{N}\}, \tag{3}$$

where $\bot_D$ is the least element of D and $\mathbb{N}$ is the natural numbers.

These results can be immediately applied to closed actor systems (those that have no external input signals) like those in Fig. 9. If each component actor A_1, A_2, and A_3 is given by continuous function, then the composite actor function F_A is continuous. Thus, it has a least fixed point, and that fixed point is given by (3). Following [44], we can define the semantics of the feedback system to be the single unique behavior that is the least fixed point. As we will see, however, this result applies much more broadly than to the process networks of [44].

It is intuitive for actors to be monotonic in the prefix order. Consider an actor A with a single input port, a single output port, and actor function F_A. Consider two possible input signals s_1 and s_2, where $s_1 \sqsubseteq s_2$. That is, s_2 extends (or equals) s_1. If F_A is monotonic, then $F_A(s_1) \sqsubseteq F_A(s_2)$. That is, $F_A(s_2)$ extends (or equals) $F_A(s_1)$. Intuitively, extending the input does not result in changes to "previously produced" outputs (the portion of the output that results from the unextended input). Thus, it is natural for actor functions to be monotonic. Intuitively, if an actor function is also continuous, then this means that the actor does not wait forever before producing output. This behavior is also intuitive and natural. Thus, we conclude that constraining functional actors to be continuous is not onerous.

Open systems have external input signals, the sources of which are not included in the model. We have a bit more work to do to build a suitable theory for such systems, but again, classic results can be applied almost immediately. As before, let $(D, \sqsubseteq)$ and $(E, \sqsubseteq)$ be CPOs, but now we consider a function of the form

$$G : D \times E \to E. \tag{4}$$

For a given $d \in D$, let $G(d) : E \to E$ be the function such that

$$\forall\ e \in E, \quad (G(d))(e) = G(d, e).$$

If G is continuous, then for all $d \in D$, $G(d)$ is continuous (lemma 8.10 in [45]). Hence, $G(d)$ has a unique least fixed point, and that fixed point is

$$\bigvee \{(G(d))^n(\bot_E) \mid n \in \mathbb{N}\},$$

where $\bot_E$ is the least element of E.

These results lead to a reasonably compositional formalism, where the properties of compositions of actors follow from the properties of the component actors. In this tutorial, we will emphasize intuition about the mathematical structure of these models. We will give brief consideration to their topological properties. And we will show concretely how they apply to familiar systems such as continuous-time control systems, discrete-event systems, and hybrid systems.

D. Background

This tutorial builds on domain theory [27], developed for the denotational semantics of programming languages [45], [46]. But unlike many semantics efforts that focus on system

state and transformation of that state (temporal logics and automata theory), we focus on concurrent interactions, and do not even assume that there is a well-defined notion of system state. In particular, we develop a timed version of the fixed-point semantics for process networks as introduced by Kahn [44]. Our version uses the tagged-signal model [47].

An alternative approach to the semantics of timed systems that is distinctly denotational builds on metric spaces. In 1988, Reed and Roscoe [48], [49] gave a semantic framework for concurrent systems, specifically timed CSP, based on complete metric spaces, saying that they "seem a natural method by which to induce a hierarchy on the various models" and they "appear more appropriate for modelling continuous concepts such as real time." The basic approach is to define a metric (actually, typically an ultrametric) on the set of traces of signals communicated between actors. The actors are then modeled as contraction maps, and the Banach fixed point theorem yields a fixed-point semantics. This approach has been pursued by others for both timed systems [50], [51] and more conventional concurrent programs [52]–[54]. In this tutorial, we will briefly review these results.

Acnknowledgements. Lee thanks Adam Cataldo, Xiaojun Liu, and Eleftherios Matsikoudis for their contributions to this work, and acknowledges the support from the National Science Foundation (NSF awards #CCR-0225610 and #0647591).

V. Application to Intelligent Transportation: Problems, Challenges, and Opportunities

Domitilla Del Vecchio(1), Derek Caveney(2), and Lorenzo Caminiti(2)
(1)EECS Systems Laboratory
University of Michigan, Ann Arbor
(2)Toyota Technical Center (TTC) of Ann Arbor

A. Introduction

Intelligent Transportation Systems (ITS) are quickly moving away from single-vehicle, autonomous sensor-based, applications to multiple-vehicle cooperative applications. This is a result of emerging technologies in global positioning and inter-vehicular wireless communications. These technologies have ignited the current growth of academic and industrial research into advanced vehicular applications for safety, mobility, and commercial benefit. Although this tutorial will focus on safety applications, strong business and environmental cases can be made for communication-enabled mobility and commercial applications. Forthcoming applications will depend on the ability of vehicles to quickly predict and then communicate their future positions, velocities, and accelerations to neighboring vehicles and infrastructure. In such a way, each vehicle can build dynamic maps of the surrounding driving environment. Conversely, current production systems including Adaptive Cruise Control (ACC), Lane Keeping Assist (LKA), Pre-Collision System (PCS), and Intuitive Parking Assist (IPA) rely solely on a myriad of autonomous sensors like radars, lidars, cameras, and ultrasonic transmitters to detect other vehicles and objects. In reality, ideal future systems will incorporate a mixture of autonomous sensors and communication between vehicles.

B. Societal Motivation

Safety applications and related technologies also continue to be examined by industry/government consortiums, such as the Crash Avoidance Metrics Partnership (CAMP[1]) in North America, the Car2Car Communications Consortium[2] in Europe, and the Advanced Safety Vehicle (ASV) Project[3] in Japan. In 2006, 42,462 motor vehicle related fatalities were registered by the National Highway Traffic Safety Administration (NHTSA) Fatality Analysis Reporting System (FARS) [55]. Fortunately, this was a 2.0% decrease over the number of fatalities (43,510) reported in 2005. However, despite the number advances in vehicular safety (ex. anti-lock brakes, airbags), the number of fatalities has remained over 40,000/year for the past 15 years. One hypothesis for this constant rate is that with the increased ability of vehicles to protect occupants from severe accidents and injuries, drivers are subsequently willing to undertake more risky driving behaviors. Vehicle-to-Vehicle (V2V), Infrastructure-to-Vehicle (I2V), and vehicle-to-infrastructure (V2I) wireless communications aim to significantly reduce these numbers, by providing information to the driver and the vehicle that is unavailable through driver perception and autonomous sensors alone.

C. Enabling Technologies

In North America, the Global Positioning System (GPS) and Differential GPS (DGPS) provide an estimate of the current location of the vehicle in an earth coordinate frame. Galileo and GLONASS are being assembled for comparable systems in Europe and Russia, respectively. DGPS relies on correction signals originating from land-based (e.g., NDGPS) or space-based (e.g., WAAS) beacons, and can attain roughly 1m accuracy in open-sky conditions. Moreover, improved positioning accuracy is possible with Real-Time Kinematic (RTK) methods that utilize the carrier-phase measurements of the GPS, Galileo, or GLONASS signals. RTK methods have shown decimeter level positioning accuracy is possible. Furthermore, two additional benefits of GPS (or similar) are that in a cooperative environment, the satellites can provide a common clock and a common earth coordinate frame for safety applications running distributively on multiple vehicles. Conversely, all the above mentioned positioning techniques are susceptible to signal loss under overpasses, within developed urban areas, and amongst heavy foliage. Fortunately, substantial academic and industrial research is now focused on GPS, Inertial Measurement Unit (IMU), and other on-board vehicle sensor integration to compensate for GPS signal outages [56], [57]. Leading academic groups

[1]The CAMP consortium comprises the following vehicle companies: DaimlerChysler, Ford, GM, Honda, and Toyota. It works in partnership with the USDOT and NHTSA.

[2]See `http://www.car-2-car.org`.

[3]The ASV is a partnership by 14 automobile, truck, and motorcycle manufacturers, sponsored by Japanese Ministry of Land, Infrastructure and Transport (See `http://www.mlit.go.jp`).

are the Position, Location, and Navigation Research Group (PLAN) at the University of Calgary[4] and Stanford's Center for Position, Navigation, and Time[5]. Dedicated Short Range Communications (DSRC) is the leading wireless technology worldwide under consideration for cooperative vehicular safety applications [58]. In the United States, the Federal Communications Commission (FCC) has allocated a 75MHz band in the 5.9GHz range for DSRC. Various layers of the DSRC stack are now under different stages of standardization (IEEE 802.11p, IEEE 1609, and SAE J2735).

Particular challenges facing wireless communication implementation involve message security, efficient message composition, reliable message dissemination, and cross-channel interference. At the application level, wireless communication-based systems must be robust to other wireless communication characteristics including packet dropping, uncertain latencies, and bandwidth limitations. While positioning and communication technologies are emerging, the amount of information available within the vehicle itself is also growing. Many modern vehicles are equipped with some form of vehicle dynamics stability system, which incorporate wheel speed sensors, yaw rate sensors, and accelerometers. This information is usually available over a vehicle communication bus (e.g., CAN, FlexRay). This expanding amount of vehicle information available from GPS positioning measurements and on-board sensor data can be used as an initial condition for a prediction routine of the host vehicle's future trajectory as well as directly shared with neighboring vehicles.

D. Collision Detection and Avoidance and the Use of Partial Order Techniques

In terms of cooperative safety systems, the majority of applications focus on collision detection and avoidance. Despite the various challenges of wireless communication technologies, inter-vehicular communication can still provide larger neighborhood maps than are possible with radars and cameras alone. These larger maps provide more time to detect and resolve conflicts in desired vehicle trajectories. The road geometry provides a structured behavior which vehicles are expected to follow. Furthermore, GPS provides imperfect information on the position of the vehicles.

The larger time horizon made possible by wireless communication enable the employment of safety control techniques based on the computation of the backward reachable set [59] or uncontrollable predecessor [60] of an unsafe set. The employment of GPS requires that a state estimator is constructed in order to cope with missing or imperfect data regarding the position of the vehicles. The road geometry and the rules of the road force the vehicles to move in one direction only in their lane. This basically implies that vehicles motion enjoys order preserving properties with respect to the order established by the lane direction of motion. The presence of traffic constraints, such as those that prevent a vehicle to stop inside an intersection, and the need to account for discrete decisions made by a driver can be modeled employing hybrid automata [61].

[4]See `http://plan.geomatics.ucalgary.ca`.
[5]See `http://scpnt.stanford.edu/`.

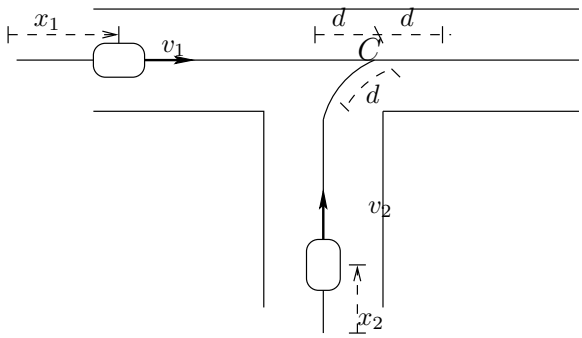

Fig. 10. Two vehicles in the proximity of a traffic "T" intersection. The unsafe set is defined as the set of all vehicle 1/vehicle 2 configurations in which the vehicles are both closer than some distance d from the intersection C of their paths. This corresponds to a rectangle in the x_1, x_2 plane.

Taking all of the above facts into account, the problem of collision detection and avoidance at traffic intersections has been formulated in [62] as an observer-based control problem for order preserving hybrid automata. In [62], the set of vehicle speed and position configurations that lead to a collision independently of driver's actions is considered for the case of two vehicles converging to an intersection. Such a set is called the escape set. By virtue of the properties of the vehicle dynamics, an approximation of the escape set can be efficiently computed by iteratively computing only suitable lower and upper bounds. A state estimator is constructed, which also keeps track of suitable upper and lower bounds in the vehicle configuration space. The basic idea of such an estimator is the one explained in Section II. The estimator gives the set of all possible current vehicle configuration values as an interval. The control algorithm guarantees that such a set never intersects the escape set. From a practical point of view, if entrance into this set is predicted, then a warning signal may be issued to the driver. Alternately, suggested control actions may be given to the driver or automatically implemented to avoid entering the escape set. These collision detection and resolution techniques that incorporate partial order approaches and consider imperfect state information could be particularly useful in highway merging and urban intersection scenarios. An illustrative example is next provided.

Example. Let us consider two vehicles converging to a traffic intersection (Fig. 10). The vehicle's physical motion can be modeled by considering its longitudinal dynamics along its geometric path (determined by the geometry of the lanes) following a similar modeling framework as performed in [59]. Let then x_1 and x_2 denote the position of the two vehicles along their path with respect to some fixed reference point. Let v_1 and v_2 be the velocities of the two vehicles along their lanes. For simplicity, we assume that each vehicle dynamics along its lane can be modeled by a simple second

order system, which in discrete time takes the form:

$$x_i' = x_i + v_i(\Delta T),\ v_i' = v_i + u_i(\Delta T), i \in \{1, 2\}, \quad (5)$$

in which ΔT is the time interval. The controller u_i can directly affect the acceleration by acting on the throttle pedal or on the brake. The system also has the following constraints. When a vehicle is inside the intersection, it cannot stop as it has to free the intersection as soon as possible, while it can stop before entering the intersection. In addition, a vehicle cannot move backwards in its lane. These constraints can be modeled, for a suitable x_i^A, by requiring that for $x_i \leq x_i^A$ then $v_i \geq 0$, while for $x_i > x_i^A$ we must have $v_i \geq v_m$ with $v_m > 0$. Let $u_m < 0 < u_M$. To enforce this requirement, we assume that u_i satisfy

$$\text{when } x_i \leq x_i^A,\ u_i \in \begin{cases} [0, u_M], & \text{if } v_i \leq 0 \\ [u_m, u_M], & \text{if } v_i > 0, \end{cases}$$

$$\text{when } x_i > x_i^A,\ u_i \in \begin{cases} [0, u_M], & \text{if } v_i \leq v_m \\ [u_m, u_M], & \text{if } v_i > v_m. \end{cases}$$

Thus, each vehicle can be described by a hybrid automaton with two modes: $q_i = q_{1,i}$ if ($x_i \leq x_i^A$ and $v_i \leq 0$) or ($x_i > x_i^A$ and $v_i \leq v_m$); $q_i = q_{2,i}$ if ($x_i \leq x_i^A$ and $v_i > 0$) or ($x_i > x_i^A$ and $v_i > v_m$). In each one of these modes, the update map is given by equations (5), in which the allowed inputs for each vehicle in each mode are given by $\iota(q_{1,i}) = [0, u_M]$ for mode $q_{1,i}$ and by $\iota(q_{2,i}) = [u_m, u_M]$ for mode $q_{2,i}$. The hybrid automaton admits only autonomous mode transitions, that is, transitions induced by the configuration of the continuous variable. The variables v_i are assumed to be directly measurable, while the variables x_i are obtained with a measurement error Δ. By exploiting the fact that the update map 5 preserves the order with respect to component-wise ordering in $\mathbb{R}^4$, an approximation of the escape set can be efficiently computed as explained in [62]. For this example, Fig. 11 shows an approximation of the escape set in the x_1, x_2 plane for the current values of v_1, v_2.

E. Open Problems

The largest open problem with existing collision detection and avoidance approaches is their scalability. Many techniques exist for pairwise collision resolution but these often don't scale well in environments where many vehicles co-exist. This is particularly true in the automotive domain where time scales are smaller, spacing distances are tighter, and vehicle densities are higher than aerospace systems. Again, the difficulty of the automotive domain emphasizes the need for computationally efficient algorithms that can be executed quickly on inexpensive processing platforms.

Secondly, cooperative systems for the automotive domain must have algorithms that are robust to the lossy behavior of the wireless communication channel. Often, control engineers assume perfect communication models in their system design and validation. Although no system can ever be guaranteed to be 100% reliable, all designs should be accompanied by some quantification of their sensitivity to various communication deficiencies.

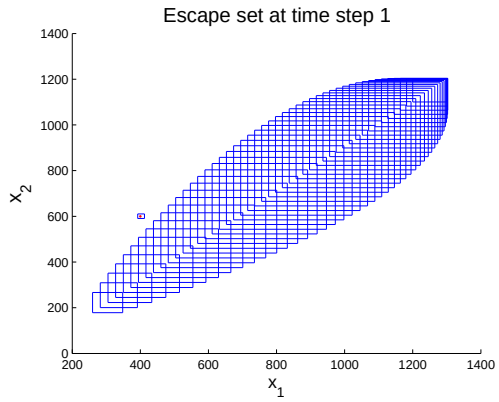

Fig. 11. The approximation of the escape set at the initial time in the x_1, x_2 plane for the initial values of velocities v_1, v_2. The dot represents the position of the vehicles in the x_1, x_2 plane and the rectangle surrounding it is given by the state estimator. (Taken from [62])

Finally, all cooperative systems that employ warnings or automated behavior struggle with driver acceptance and expectation. Systems that maintain a level of driver engagement at all times in the driving cycle should provide more positive driver reactions to these systems' behaviors.

REFERENCES

[1] J. P. Hespanha and A. Tiwari, Eds., *Hybrid Systems: Computation and Control, 9th International Workshop, HSCC 2006, Santa Barbara, CA, USA, March 29-31, 2006, Proceedings*, ser. Lecture Notes in Computer Science, vol. 3927. Springer, 2006.

[2] E. Asarin and P. Bouyer, Eds., *Formal Modeling and Analysis of Timed Systems, 4th International Conference, FORMATS 2006, Paris, France, September 25-27, 2006, Proceedings*, ser. Lecture Notes in Computer Science, vol. 4202. Springer, 2006.

[3] R. Alur, T. A. Henzinger, and O. Kupferman, "Alternating-time temporal logic." *J. ACM*, vol. 49, no. 5, pp. 672–713, 2002.

[4] L. de Alfaro, T. A. Henzinger, and R. Majumdar, "From verification to control: Dynamic programs for omega-regular objectives." in *LICS*, 2001, pp. 279–290.

[5] L. de Alfaro and T. A. Henzinger, "Concurrent omega-regular games," in *LICS*, 2000, pp. 141–154.

[6] K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin, "Algorithms for omega-regular games with imperfect information," in *CSL06*, ser. Lecture Notes in Computer Science, vol. 4207. Springer, 2006, pp. 287–302.

[7] M. D. Wulf, L. Doyen, and J.-F. Raskin, "A lattice theory for solving games of imperfect information." in *HSCC 2006*, ser. Lecture Notes in Computer Science, vol. 3927. Springer, 2006, pp. 153–168.

[8] A. Bemporad, G. Ferrari-Trecate, and M. Morari, "Observability and controllability of piecewise affine and hybrid systems," *IEEE Transactions on Automatic Control*, vol. 45, pp. 1864–1876, 1999.

[9] A. Balluchi, L. Benvenuti, M. D. D. Benedetto, and A. Sangiovanni-Vincentelli, "Design of observers for hybrid systems," in *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science vol. 2289, C. J. Tomlin and M. R. Greensreet, Eds. Springer Verlag, 2002, pp. 76–89.

[10] ——, "Observability of hybrid systems," in *Conf. on Decision and Control*, 2003, pp. 1159–1164.

[11] A. Alessandri and P. Coletta, "Design of Luenberger observer for a class of hybrid linear systems," in *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, vol. 2034, M. D. Di Benedetto, A. Sangiovanni-Vincentelli, Eds. Springer Verlag, 2001, pp. 7–18.

[12] ——, "Design of observer for switched discrete-time linear systems," in *American Control Conference*, 2003, pp. 2785–2790.

[13] R. Vidal, A. Chiuso, and S. Soatto, "Observability and identifiability of jump linear systems," in *Conf. on Decision and Control*, 2002, pp. 3614 – 3619.

[14] R. Vidal, A. Chiuso, S. Soatto, and S. Sastry, "Observability of linear hybrid systems," in *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, vol. 2623, O. Maler and A. Pnueli, Eds. Springer Verlag, 2003, pp. 526–539.

[15] E. D. Santis, M. D. D. Benedetto, and G. Pola, "On observability and detectability of continuous-time linear switching systems," in *Conf. on Decision and Control*, 2003, pp. 5777–5782.

[16] D. Del Vecchio and R. M. Murray, "Discrete state estimators for a class of hybrid systems on a lattice," in *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, vol. 2993, R. Alur and G. Pappas, Eds. Springer Verlag, Philadelphia, 2004, pp. 311–325.

[17] ——, "Existence of cascade discrete-continuous state estimators on a partial order," in *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, vol. 3414, M. Morari, L. Thiele, and F. Rossi, Eds. Springer Verlag, Zurich, 2005, pp. 226–241.

[18] P. J. Ramadge, "Observability of discrete event systems," in *Conf. on Decision and Control*, 1986, pp. 1108–1112.

[19] M. Oishi, I. Hwang, and C. Tomlin, "Immediate observability of discrete event systems with application to user-interface design," in *Conf. on Decision and Control*, 2003, pp. 2665 – 2672.

[20] C. M. Özveren and A. S. Willsky, "Observability of discrete event dynamic systems," *IEEE Transactions on Automatic Control*, vol. 35, no. 7, pp. 797–806, 1990.

[21] P. E. Caines, R. Greiner, and S. Wang, "Classical and logic-based dynamic observers for finite automata," *IMA J. of Mathematical Control and Information*, pp. 45–80, 1991.

[22] P. E. Caines and S. Wang, "COCOLOG: A conditional observer and controller logic for finite machines," *SIAM J. on Control and Optimization*, pp. 1687–1715, 1995.

[23] D. Del Vecchio and E. Klavins, "Observation of guarded command programs," in *Conf. on Decision and Control*, 2003, pp. 3353–3359.

[24] D. Del Vecchio, R. M. Murray, and E. Klavins, "Discrete state estimators for systems on a lattice," *Automatica*, vol. 42, no. 2, pp. 271–285, 2006.

[25] D. Del Vecchio and R. M. Murray, "Discrete state estimation for nondeterministic hybrid systems on a partial order," in *Conf. on Decision and Control*, Paradise Island, Bahamas, 2004, pp. 3215–3220.

[26] ——, "Existence of discrete state estimators for hybrid systems on a lattice," in *Conf. on Decision and Control*, Paradise Island, Bahamas, 2004, pp. 1–6.

[27] S. Abramsky and A. Jung, "Domain theory," in *Handbook of logic in computer science (vol. 3): semantic structures*. Oxford, UK: Oxford University Press, 1995, pp. 1–168.

[28] R. D'Andrea, R. M. Murray, J. A. Adams, A. T. Hayes, M. Campbell, and A. Chaudry, "The RoboFlag Game," in *American Control Conference*, 2003, pp. 661–666.

[29] D. Del Vecchio, "A partial order approach to discrete dynamic feedback in a class of hybrid systems," in *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, vol. 4416, A. Bemporad, A. Bicchi, and G. Buttazzo, Eds. Springer-Verlag, 2007, pp. 159–173.

[30] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th IEEE Symposium Foundations of Computer Science*, 1977, pp. 46–57.

[31] G. E. Fainekos, "An introduction to multi-valued model checking," Dept. of CIS, Univ. of Pennsylvania, Tech. Rep. MS-CIS-05-16, September 2005.

[32] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. Cambridge, Massachusetts: MIT Press, 1999.

[33] M. Chechik, B. Devereux, S. Easterbrook, and A. Gurfinkel, "Multi-valued symbolic model-checking," *ACM Transactions on Software Engineering and Methodology*, vol. 12, no. 4, pp. 1–38, Oct. 2004.

[34] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications," in *Proceedings of FATES/RV*, ser. LNCS, vol. 4262. Springer, 2006, pp. 178–192.

[35] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE Trans. Auto. Cont.*, vol. 52, no. 5, pp. 782–798, 2007.

[36] G. E. Fainekos, A. Girard, and G. J. Pappas, "Temporal logic verification using simulation," in *Proceedings of FORMATS*, ser. LNCS, vol. 4202. Springer, 2006, pp. 171–186.

[37] Z. Han, "Formal verification of hybrid systems using model order reduction and decomposition," Ph.D. dissertation, Dept. of ECE, Carnegie Mellon University, 2005.

[38] Z. Manna and A. Pnueli, "Verifying hybrid systems," *Hybrid Systems*, pp. 4–35, 1992.

[39] E. A. Lee and H. Zheng, "Operational semantics of hybrid systems," in *Hybrid Systems: Computation and Control (HSCC)*, M. Morari and L. Thiele, Eds., vol. LNCS 3414. Zurich, Switzerland: Springer-Verlag, 2005, pp. pp. 25–53.

[40] Y. Gurevich, "Evolving algebras 1993: Lipari Guide," in *Specification and Validation Methods*, E. Börger, Ed., 1994, pp. 9–37.

[41] P. Taylor, *Practical Foundations of Mathematics*. Cambridge University Press, 1999.

[42] X. Liu, "Semantic foundation of the tagged signal model," EECS Department, University of California," PhD Thesis, December 20 2005.

[43] B. A. Davey and H. A. Priestly, *Introduction to Lattices and Order*. Cambridge University Press, 1990.

[44] G. Kahn, "The semantics of a simple language for parallel programming," in *Proc. of the IFIP Congress 74*. North-Holland Publishing Co., 1974.

[45] G. Winskel, *The Formal Semantics of Programming Languages*. Cambridge, MA, USA: MIT Press, 1993.

[46] J. E. Stoy, *Denotational Semantics*. Cambridge, MA: MIT Press, 1977.

[47] E. A. Lee and A. Sangiovanni-Vincentelli, "A framework for comparing models of computation," *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, vol. 17, no. 12, pp. 1217–1229, 1998.

[48] G. M. Reed and A. W. Roscoe, "Metric spaces as models for real-time concurrency," in *3rd Workshop on Mathematical Foundations of Programming Language Semantics*, London, UK, 1988, pp. 331–343.

[49] ——, "A timed model for communicating sequential processes," *Theoretical Computer Science*, vol. 58, pp. 249–261, 1988.

[50] R. K. Yates, "Networks of real-time processes," in *Proc. of the 4th Int. Conf. on Concurrency Theory (CONCUR)*, E. Best, Ed., vol. LNCS 715. Springer-Verlag, 1993.

[51] E. A. Lee, "Modeling concurrent real-time processes using discrete events," *Annals of Software Engineering*, vol. 7, pp. 25–45, 1999.

[52] A. Arnold and M. Nivat, "Metric interpretations of infinite trees and semantics of non deterministic recursive programs," *Fundamenta Informaticae*, vol. 11, no. 2, pp. 181–205, 1980.

[53] C. Baier and M. E. Majster-Cederbaum, "Denotational semantics in the cpo and metric approach," *Theoretical Computer Science*, vol. 135, no. 2, pp. 171–220, 1994.

[54] J. W. de Bakker and E. P. de Vink, "Denotational models for programming languages: Applications of Banachs fixed point theorem," *Topology and its Applications*, vol. 85, pp. 35–52, 1998.

[55] "2006 traffic safety annual assessment - a preview," in *DOT HS 810 791, National Center for Statistics and Analysis (NCSA), of NHTSA.*, http://www-nrd.nhtsa.dot.gov/Pubs/810791.pdf, July 2007.

[56] S. Godha and M. Cannon, "Integration of DGPS with a low cost MEMS-based inertial measurement unit (IMU) for land vehicle navigation application," in *Proceedings of ION GNSS*, Long Beach, CA, September 2005.

[57] J. Gao, M. Petovello, and M. Cannon, "Development of precise GPS/INS/wheel speed sensor/yaw rate sensor integrated vehicular positioning system," in *Proceedings of ION National Technical Meeting*, Monterey, CA, January 2006.

[58] A. Carter, "The status of vehicle-to-vehicle communication as a means of improvement crash prevention performance," in *NHTSA Tech. Rep. 05-0264*, http://www-nrd.nhtsa.dot.gov/pdf/nrd-01/esv/esv19/05-0264-W.pdf, 2005.

[59] C. J. Tomlin, J. Lygeros, and S. Sastry, "A game theoretic approach to controller design for hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 949–970, 2000.

[60] T. A. Henzinger and P. W. Kopke, "Discrete-time control for rectangular hybrid automata," *Theoretical Computer Science*, vol. 221, pp. 369–392, 1999.

[61] T. A. Henzinger, "The theory of hybrid automata," in *Proceedings of the 11th Annual Symposium on Logic in Computer Science*. IEEE press, 1996, pp. 278–292.

[62] D. Del Vecchio, "Observer-based control for block-triangular hybrid automata," in *Conf. on Decision and Control*, 2007, p. (To Appear).