Title
Directed specifications and assumption mining for monotone dynamical systems

Permalink
https://escholarship.org/uc/item/58g4p1dj

ISBN
9781450339551

Authors
Kim, ES
Arcak, M
Seshia, SA

Publication Date
2016-04-11

DOI
10.1145/2883817.2883833

Peer reviewed
ABSTRACT

Given a dynamical system and a specification, assumption mining is the problem of identifying the set of admissible disturbance signals and initial states that generate trajectories satisfying the specification. We first introduce the notion of a directed specification, which describes either upper or lower sets in a partially ordered signal space, and show that this notion encompasses an expressive temporal logic fragment. We next show that the order preserving nature of monotone dynamical systems makes them amenable to a systematic form of assumption mining that checks numerical simulations of system trajectories against directed specifications. The assumption set is then located with a multidimensional bisection method that converges to the boundary from above and below. Typical objectives in vehicular traffic control, such as avoiding or clearing congestion, are directed specifications. In an application to a freeway flow model with monotone dynamics, we identify the set of vehicular demand profiles that satisfy a specification that congestion be intermittent.

Keywords
Monotone Control Systems, Temporal Logic, Partially Ordered Sets, Assumption Mining

1. INTRODUCTION

Component-based design and analysis is a common paradigm for managing complexity in large networked systems. Each component is characterized by an input-output relationship, such as a finite input-output gain in control theory or an assume-guarantee contract in the formal methods literature, enabling higher order reasoning about global behavior.

To develop complex, yet robust, systems through interconnections of simpler components, identifying failure modes and determining the limits for safe system operation is of utmost importance. Towards this end, we formulate an assumption mining problem for dynamical systems where, given a deterministic system and a specification encoded as a set of acceptable state trajectories, we seek the largest set of initial states and exogenous disturbances for which the system satisfies the specification. Computing an exact representation of the assumption set for arbitrary specifications and dynamics is impossible. The next best option is to systematically and extensively test the system under a variety of environmental disturbances to construct an approximation of the assumption set.

In this paper, we define directed specifications and show that when paired with monotone dynamical systems, they are well suited to a systematic form of assumption mining. Directed specifications correspond to lower and upper sets in a signal space and thus favor signals with low or high values respectively. For instance, in a traffic network where the state represents number of vehicles, the specification “congestion will never be present” is a directed specification because it encourages state signals with low vehicle counts. As depicted in Fig. 1, lower and upper sets have a convenient geometry that makes it possible to use individual samples to extrapolate information about set membership. We provide a set of syntactic rules that provide a sufficient condition for a temporal logic specification to be directed. This condition is agnostic to timing semantics of the chosen temporal
logic specification language and encompasses linear temporal logic [17] and signal temporal logic [12] as long as predicates are over a partially ordered set.

Monotone dynamical systems exhibit order preserving dynamics and provide a clear functional relationship between the space of initial states and disturbance signals with the trajectories they generate. In particular, they preserve the aforementioned directed property, and the assumption set must be lower (upper) if the specification is lower (upper).

To construct a tight approximation of a directed set in Euclidean space, it suffices to converge to its boundary from below and above. We exploit the extrapolation property of directed sets highlighted in Fig. 1 and utilize a variant of multi-dimensional binary search for discrete time signals of finite length and spaces of finite dimension to converge to the boundary. Our solution uses simulation traces and harnesses a satisfiability modulo theories (SMT) solver [3] to systematically explore the space of initial state and disturbance signals until it provides a certificate that the assumption set is approximated to a desired precision.

To summarize, we make two primary contributions about mining assumptions for monotone control systems:

1. We define directed specifications and show that they encompass an expressive temporal logic fragment.
2. Given a directed specification and a monotone system, we characterize the set of admissible disturbance signals and show how a bisection algorithm exploits the ordering present in the problem.

Section 3 first describes directed specifications geometrically as a subset of the signal space and shows how to construct directed temporal logic specifications, Section 4 reviews monotone dynamical systems, and Section 5 explains how we can exploit both the specification and dynamics via a generalized bisection method.

Related Work

Assumption mining has previously been studied for the synthesis of discrete controllers that realize a temporal logic specification [1][4][16]. Our formulation of the assumption mining problem resembles the problem of computing weakest preconditions but we also compute admissible disturbance profiles [10].

Our work contains a number of parallels to prior work on requirement mining by Jin, Donze, Deshmukh, and Seshia [14] and robust controller synthesis by Topcu, Ozay, Liu, and Murray [19]. Both make use of partial orderings and allude to similar bisection search heuristics, but neither utilize properties of directed sets and their orderings are over different sets than those found in this paper. Jin et al. use monotonicity of a parametric signal temporal logic (pSTL) template to prune regions of the parameter space and find a tight overapproximation of the system’s possible state trajectories. Fundamentally, monotonicity with respect to pSTL parameters is about comparing two different specifications and checking if one implies the other. Directedness, in contrast, is a property that is intrinsic to a single specification.

In addition, in our work monotonicity is a dynamical system property and should not be confused with monotonicity with respect to specification parameters. Topcu et al. seek to synthesize a controller that is robust to the presence of an environmental adversary with varying levels of strength, where a stronger adversary has more available moves in the synthesis game. Our problem is formulated in a way that stronger environments are encoded directly in the partial ordering on disturbance signals, instead of as a larger set of available environment disturbances.

2. PRELIMINARIES

2.1 Notation and Terminology

For a given set \( P \) we let \( P^C \) and \( P \times Q \) respectively denote its complement (with respect to some universal set) and its Cartesian product with \( Q \). The empty set is \( \emptyset \). The symbol \( \Rightarrow \) represents the Boolean implication operator while the symbol \( \leftrightarrow \) represents a map between a domain and codomain. The image \( f(M) \) of a set \( M \subseteq P \) under function \( f : P \rightarrow Q \) is the set of points \( \{ f(x) : x \in M \} \) and the preimage \( f^{-1}(N) \) of the set \( N \subseteq Q \) is \( \{ x \in P : f(x) \in N \} \).

The sets \( \mathbb{R}_{\geq 0} \) and \( \mathbb{Z}_{\geq 0} \) are the sets of non-negative real numbers and integers with \( \mathbb{R}_{\geq 0}^n \) representing the non-negative orthant. A discrete time interval \( I = [a, b] \) is a contiguous subset of \( \mathbb{Z}_{\geq 0} \) where \( a, b \in \mathbb{Z}_{\geq 0} \cup \{ \infty \} \) and \( a \leq b \). The Boolean domain is denoted \( \mathbb{B} = \{ \bot, \top \} \) where \( \top \) is true and \( \bot \) is false.

The sets \( X \subseteq \mathbb{R}^n \), \( D \subseteq \mathbb{R}^m \) and \( Y \subseteq \mathbb{R}^p \) represent state, disturbance, and output spaces for appropriate positive integers \( n, m, p \) and let \( x[k], d[k], y[k] \) respectively denote variables in these spaces at time \( k \). When clear from context, the time index is dropped. For a set \( \mathcal{P} \) and an interval \( I \), the space of signals, \( \mathcal{P}[I] \), is given by a Cartesian product indexed by elements of \( I \):

\[
\mathcal{P}[I] = \prod_{k \in I} \mathcal{P}.
\]

For instance, a discrete-time real signal of length \( N \) can be thought of as a point in \( \mathbb{R}^N \). In this paper, the terms signal, trace, and trajectory are synonyms. For notational brevity, the interval \( I \) is typically omitted and only specified when necessary. The sets \( X[I] \), \( D[I] \), \( Y[I] \) are the set of state signals, disturbance signals, and output signals.

A specification \( \phi \) can be viewed as the subset of the signal space for which it is true. We signify that a signal \( x[I] \) satisfies a specification \( \phi \) by \( x[I] \models \phi \). One can switch between set theoretic and Boolean views of \( \phi \) by the definition \( \phi = \{ x[I] : x[I] \models \phi \} \) and the identity \( x[I] \models \phi \) if and only if \( x[I] \in \phi \) (using the set theoretic definition of \( \phi \)).

2.2 Assumption Mining

A deterministic dynamical system \( \Sigma : X \times D[I] \rightarrow X[I] \) is a map from an initial state and disturbance pair to a state trajectory. We formulate assumption mining as the problem of determining which exogenous disturbances and initial conditions are permitted for a system to satisfy a specification.

**Problem 1.** (Assumption Mining). Given a deterministic system \( \Sigma : X \times D[I] \rightarrow X[I] \) and a specification \( \phi \subseteq X[I] \),
what is the subset of initial states and disturbance signals, \( \Sigma^{-1}(\phi) \subseteq \mathcal{X} \times \mathcal{D}[\cdot] \), that ensures satisfaction of \( \phi \)?

By viewing a dynamical system \( \Sigma \) as a mapping \( \Sigma : \mathcal{X} \times \mathcal{D}[\cdot] \rightarrow \mathcal{X}[\cdot] \), the solution to the assumption mining problem is found by computing the pre-image \( \Sigma^{-1}(\phi) \) of specification \( \phi \). Computation of pre-images is typically intractable for arbitrary specifications \( \phi \) and non-linear dynamics for \( \Sigma \). We focus on monotone dynamical systems and directed specifications and show that this pair is particularly amenable to assumption mining.

### 2.3 Partially Ordered Sets and Signals

A partially ordered set \( \mathcal{P} \) has an associated binary relation \( \leq_p \) if all \( p_1, p_2, p_3 \in \mathcal{P} \) satisfy 1) \( p_1 \leq_p p_2 \) and \( p_2 \leq_p p_3 \) then \( p_1 \leq_p p_3 \), and 2) if \( p_1 \leq_p p_2 \) and \( p_2 \leq_p p_3 \) then \( p_1 \leq_p p_3 \). We define \( \geq_p \) so that \( p_1 \geq_p p_2 \) holds if and only if \( p_2 \leq_p p_1 \). If neither \( p_1 \leq_p p_2 \) nor \( p_1 \geq_p p_2 \) hold, we say that \( p_1 \) and \( p_2 \) are incomparable.

Given a collection of partially ordered sets \( \mathcal{P}_i \), and relations \( \leq_{p_i} \) indexed by \( \mathcal{A} \), let \( \mathcal{P} = \prod_{i \in \mathcal{A}} \mathcal{P}_i \), and \( \pi_i(p) : \mathcal{P} \mapsto \mathcal{P}_i \) map \( p \in \mathcal{P} \) to its \( i \)-th component. For \( p_1, p_2 \in \mathcal{P} \), the product ordering relation \( p_1 \leq_p p_2 \) holds if and only if \( \pi_i(p_1) \leq_{p_i} \pi_i(p_2) \) for all \( i \in \mathcal{A} \). Time will frequently play the role of the index set as it does in (1).

In this paper, all sets \( \mathcal{X}, \mathcal{D}, \) and \( \mathcal{Y} \) are equipped with partial orders \( \leq_{\mathcal{X}}, \leq_{\mathcal{D}}, \) and \( \leq_{\mathcal{Y}} \). We also introduce an induced signal partial ordering \( \leq_{[\cdot]} \) over \( \mathcal{P} \) and interval \( I = [a, b] \) such that for signals \( p_1[\cdot], p_2[\cdot] \) the ordering \( p_1[\cdot] \leq_{[\cdot]} p_2[\cdot] \) signifies that \( p_1[k] \leq_p p_2[k] \) for all \( k \in [a, b] \).

A function between partially ordered sets \( f : \mathcal{P} \mapsto \mathcal{Q} \) is a monotone function if \( p_1 \leq_p p_2 \) implies \( f(p_1) \leq_q f(p_2) \) for all \( p_1, p_2 \in \mathcal{P} \). The composition of monotone functions is also a monotone function [7].

### 3. DIRECTED SPECIFICATIONS

#### 3.1 Lower and Upper Sets

**Definition 1.** (Lower Set) Given a partially ordered set \( \mathcal{P} \) with relation \( \leq_p \), a subset \( \mathcal{L} \subseteq \mathcal{P} \) is a lower set if for all \( p, q \in \mathcal{P} : \)

\[
p \in \mathcal{L} \text{ and } q \leq_p p \implies q \in \mathcal{L}.
\]

(2)

**Property 1.** If \( \mathcal{L} \subseteq \mathcal{P} \) is a lower set, then \( \mathcal{L}^c \) is an upper set.

**Property 2.** The collection of all lower sets of \( \mathcal{P} \) is closed under arbitrary unions and intersections.

**Property 3.** Let the collection of lower sets \( \mathcal{L}_i \subseteq \mathcal{P}_i \) be indexed by a set \( \mathcal{A} \) and \( \mathcal{P} = \prod_{i \in \mathcal{A}} \mathcal{P}_i \). Their Cartesian product \( \prod_{i \in \mathcal{A}} \mathcal{L}_i \) is a lower set with the product ordering \( \leq_p \).

**Property 4.** Sets \( \mathcal{P} \) and \( \emptyset \) are both upper and lower sets.

We now define a set of specifications \( \phi \) that are satisfied on lower/upper sets in the signal space.

**Definition 2.** (Lower/Upper Specifications). A lower specification \( \phi \) on signals \( \mathcal{X}[\cdot] \) satisfies

\[
(\langle x_1[\cdot], x_2[\cdot] \rangle \leq_{\mathcal{X}[\cdot]} \mathcal{X}[\cdot]) \wedge (x_2[\cdot] \models \phi) \implies (x_1[\cdot] \models \phi).
\]

(3)

Likewise, an upper specification \( \phi \) on signals \( \mathcal{X}[\cdot] \) satisfies

\[
(\langle x_1[\cdot], x_2[\cdot] \rangle \leq_{\mathcal{X}[\cdot]} \mathcal{X}[\cdot]) \wedge (x_1[\cdot] \models \phi) \implies (x_2[\cdot] \models \phi).
\]

(4)

A directed specification is one that is either a lower or upper specification.

Common examples of lower specifications in the control theory literature are the set of non-negative signals with upper...
bounds on some norm, as shown in Fig. 3, and the set of non-negative signals that converge to zero. However, only considering specifications of interest to be over sublevel sets with respect to some norm is too restrictive and does not permit non-convex specifications such as “intermittent spikes in freeway occupancy are permitted as long as they do not last longer than 30 minutes”. Lower sets do not need to be convex, open, or closed, and allow specifications like the freeway specification above. The following section covers a set of rules that provides a sufficient condition for a temporal logic specification to be directed.

3.2 Constructing Directed Specifications

Temporal logics are a logical formalism for expressing specifications as sets of satisfying signals [17]. We restrict our interest to signals over a partially ordered set \( X \). A predicate \( \mu : X \rightarrow \mathbb{B} \) assigns a truth value to elements in \( X \). As an example, consider a discrete-time variant on signal temporal logic (STL) [12] where specifications can be constructed with the grammar

\[
\phi := \top \lor \neg \phi_1 \land \phi_2 | \phi_1 \text{U} \phi_2
\]

(5)

where \( I \) is an interval, \( \neg \) is Boolean negation, and \( \land \) is a Boolean AND. Specification \( \phi_1 \text{U} \phi_2 \) is true if there exists a time \( k \in I \) such that \( \phi_2 \) is true at time \( k \) and \( \phi_1 \) is true until \( k \). From the above grammar, one can derive additional temporal operators \( \diamond_I \phi := \bigcup_{j \in I} \phi \) for “\( \phi \) is eventually true in \( I \)” and \( \Box_I \phi := \neg (\diamond_I \neg \phi) \) for “\( \phi \) is always true in \( I \)”. Formally:

\[
(x[],k) \models \mu \quad \text{iff} \quad x[k] \models \mu \\
(x[],k) \models \neg \phi \quad \text{iff} \quad (x[],k) \not\models \phi \\
(x[],k) \models \phi_1 \land \phi_2 \quad \text{iff} \quad (x[],k) \models \phi_1 \land (x[],k) \models \phi_2 \\
(x[],k) \models \phi_1 \text{U} \phi_2 \quad \text{iff} \quad \exists p \in [k + a, k + b] \text{ such that } (x[],p) \models \phi_2 \land \forall q \in [k + a, p - 1], (x[],q) \models \phi_1
\]

When temporal operators omit the interval \( I \), it is assumed that \( I = [0, \infty) \) and \( (x[],0) \models \phi \) is a shorthand for \( (x[],0) \models \phi \).

In this section, we adopt a geometric view of temporal logic formulas with predicates on \( \mathbb{R} \) by viewing them as subsets of a Euclidean signal space, just as level sets of \( l_1, l_2 \) and \( l_\infty \) norms are visualized as high dimensional diamonds, balls and boxes. Fig. 4 depicts a specification consisting of an eventually operator and shows that it can be thought of as a union over different sets in the signal space. The mixed-integer constraints appearing in model predictive control with temporal logic constraints effectively encode unions of polyhedra in a signal space [20] [18].

3.3 Order Preserving Operations

We outline a fragment of the temporal logic over a partially ordered set by restricting the grammar (5) in such a way that all generated specifications are directed.

**Theorem 1.** Let \( \mu^l \) and \( \mu^u \) be restricted to predicates that are true on lower and upper sets of \( X \) respectively. Let \( \phi^d \) be constructed with the grammar

\[
\phi^d := \phi^l | \phi^u \\
\phi^l := \top | \neg \mu^l \lor \phi_1^l \land \phi_2^l | \phi_1^l \text{U} \phi_2^l \\
\phi^u := \top | \neg \phi^l \lor \phi_1^u \land \phi_2^u | \phi_1^u \text{U} \phi_2^u
\]

Any specification \( \phi^d \) respecting the grammar above is a directed specification of \( X[\cdot] \) satisfying (3) or (4).

**Proof.** We adopt a syntax directed approach to proving that a specification is directed and only prove the following statements about lower specifications \( \phi^l \). The dual statements for upper specifications are easily derived.

- Formulas \( \phi = \top \) and \( \phi = \bot \) are both lower and upper specifications.
  **Proof:** Follows from Property 4.

- If predicate \( \mu^l \) is true on a lower set in \( X \), then \( \phi = \mu^l \) is a lower specification in \( X[\cdot] \).
  **Proof Sketch:** Follows from the definition of lower set, Property 3, and the identity:

\[
\{x[\cdot] : x[0] \in \mu^l \} \equiv \{x[\cdot] : x[\cdot] \in \left( \mu^l \times \prod_{i \in [1,\infty]} X \right) \}
\]

- If \( \phi^l \) is a lower specification, then \( \neg \phi^l \) is an upper specification, as shown using DeMorgan’s law:

\[
\begin{align*}
& (x[\cdot] [\leq x[\cdot]] x[\cdot] \land x[\cdot] [\leq x[\cdot]] \Rightarrow \phi^l) \Rightarrow (x[\cdot] [\leq x[\cdot]] \neg \phi^l) \\
& \equiv \neg (x[\cdot] [\leq x[\cdot]] x[\cdot] \land x[\cdot] [\leq x[\cdot]] \Rightarrow \phi^l) \lor (x[\cdot] [\leq x[\cdot]] \neg \phi^l) \\
& \equiv x[\cdot] \not\leq x[\cdot] x[\cdot] \lor x[\cdot] \not\leq x[\cdot] \Rightarrow \neg \phi^l \lor x[\cdot] \not\leq x[\cdot] \Rightarrow \neg \phi^l \\
& \equiv (x[\cdot] [\leq x[\cdot]] x[\cdot] \land x[\cdot] \not\leq x[\cdot] \Rightarrow \neg \phi^l) \lor x[\cdot] \not\leq x[\cdot] \Rightarrow \neg \phi^l.
\end{align*}
\]

- If \( \phi^l_1 \) and \( \phi^l_2 \) are both lower specifications, then \( \phi^l_1 \land \phi^l_2 \) also is a lower specification.
  **Proof Sketch:** Consider two signals \( x_1[\cdot], x_2[\cdot] \) where \( x_1[\cdot] \not\leq x_1[\cdot] x_2[\cdot] \). If \( x_2[\cdot] \models \phi^l_1 \land \phi^l_2 \), then the inequality \( x_1[\cdot] \not\leq x_1[\cdot] x_2[\cdot] \) and the definition of lower specification guarantee that \( x_1[\cdot] \models \phi^l_1 \land x_1[\cdot] \models \phi^l_2 \).

- If \( \phi^l_1 \) and \( \phi^l_2 \) are lower specifications, then \( \phi^l_1 \text{U} \phi^l_2 \) also is a lower specification.
  **Proof Sketch:** Consider two signals \( x_1[\cdot], x_2[\cdot] \) where \( x_1[\cdot] \not\leq x_1[\cdot] x_2[\cdot] \) and \( x_2[\cdot] \models \phi^l_1 \text{U} \phi^l_2 \). At some time \( p \in [a,b] \), signal \( x_2[\cdot,p] \models \phi^l_2 \) and the definition
of lower specification guarantees that \((x_1[,\, p)] \models \phi_2'\) \((x_1[\cdot]) \text{ may satisfy } \phi_2'\text{ earlier than time } p\). A similar argument can be made about \(x_1[\cdot] \models \phi_1'\) for all time in \([a, p - 1]\).

Although this proof is on discrete-time specifications, the above properties also apply to specifications with continuous-time semantics.

It is straightforward to prove the above properties about temporal logic operators in a set theoretic context using Properties 1-4 of lower/upper sets. The above are sufficient conditions to determine satisfaction of properties (3) and (4), and allow derivation of similar statements for \(\square\), \(\Diamond\), \(\bigcirc\phi\), \(\phi_1 \lor \phi_2\), and \(\phi_1 \Rightarrow \phi_2\).

- \(\square\phi\) and \(\Diamond\phi\) are lower specifications if \(\phi\) is a lower specification. Fig. 5 demonstrates how order preserving operators are used to derive \(\square\phi\)'s directed property.
- \(\phi_1 \lor \phi_2\) is a lower specification if \(\phi_1\) and \(\phi_2\) are lower specifications.
- \(\phi_1 \Rightarrow \phi_2\) is a lower specification if \(\phi_1\) is an upper specification and \(\phi_2\) is a lower specification.

Note that the proof for Theorem 1 only assumes that sets have a partial ordering and no makes no restrictions that predicates be over discrete or continuous sets such as \(B^n\) or \(\mathbb{R}^n\), respectively.

Curiously, it is possible to generate lower specifications that combine elements of temporal logics and norms that are not expressible in either alone. Consider

\[
\Box_{[0,100]} [0,7] \left( \sum_{i=0}^{4} x[i] \leq 3 \right)
\]

which encodes that a running average is periodically below a constant, is a specification that cannot be written in signal temporal logic, yet is still a lower specification.

4. MONOTONE SYSTEM DYNAMICS

Let the discrete time system \(\Sigma\) have an associated update equation \(F_\Sigma: X \times D[\cdot] \to X\) such that

\[
x[k + 1] = F_\Sigma(x[k], d[k])
\]

for all \(k \geq 0\).

**Definition 3 (Monotone Systems).** A system \((6)\) is monotone with respect to ordering \(\leq_X \text{ and } \leq_D\) if

\[
x_1 \leq_X x_2 \text{ and } d_1 \leq_D d_2 \Rightarrow F_\Sigma(x_1, d_1) \leq_X F_\Sigma(x_2, d_2).
\]

The system has a monotone output if the output function \(b(\cdot): X[\cdot] \to Y[\cdot]\) is monotone.

To extend the definition of monotone system from a single-step update equation \(F_\Sigma\) to a definition about the signals generated by \(\Sigma\), consider \(d_1[\cdot] \leq_D d_2[\cdot]\) on the interval \([a, b]\) and \(x_1[a] \leq_X x_2[a]\). If the system is monotone, it follows from iterating Definition 3 via (6) that:

\[
x_1[\cdot] \leq_X x_2[\cdot]
\]

for state signals \(x_1[\cdot], x_2[\cdot]\) on the interval \([a, b + 1]\).

Thus monotonicity of \(F_\Sigma(\cdot, \cdot)\) in (7) implies the monotonicity of \(\Sigma: X \times D[\cdot] \to X[\cdot]\).

**Example 1.** Let \(a \geq 0, x \in \mathbb{R}_{\geq 0}, d \in \mathbb{R}\). The system

\[
x[k + 1] = \max(0, ax[k] + d[k])
\]

is monotone.

Consider the assumption mining problem as formalized in Problem 1, but with the additional information that the system \(\Sigma\) is monotone and the specification \(\phi\) is a lower set. The following property of lower sets and monotone functions lets us deduce that the assumption set is also a lower set as depicted in Fig. 6.

**Property 5.** If \(f: P \to Q\) is a monotone function, the preimage \(f^{-1}(M)\) of a lower(upper) set, \(M \subseteq Q\), is itself a lower(upper) set.
Because \( \Sigma \) is monotone and \( \preceq \) is a partial order, we have that \( \Sigma^{1}(\phi) \) is a lower set. Thus, the assumption set will also be directed with the same polarity.

\[
\Sigma^{-1}(\phi) = \{ x \in \mathcal{P} | f(x) \in \mathcal{M} \} = \{ x \in \mathcal{P} | f(x) \preceq x \}
\]

We assume that a single simulation of \( \Sigma \) with different initial states and disturbance signals induces a trajectory that either satisfies \( \phi \) or \( \neg \phi \). To evaluate a trace's satisfaction of a specification, we would use the Breach or S-TaLiRo toolboxes [11][2]. An initial state and disturbance pair \( (x_0, d[\cdot]) \) is used to underapproximate \( \Sigma^{-1}(\phi) \) if \( \Sigma(x_0, d[\cdot]) \models \phi \) and is used to underapproximate \( \Sigma^{-1}(\neg \phi) \) if \( \Sigma(x_0, d[\cdot]) \models \neg \phi \) (see lines 8-13 of Algorithm 1 and green/red points in Fig. 7a).

Observe that the boundary between a lower and upper set can be under-approximated arbitrarily well by a union of principal sets, which under a coordinate-wise ordering are rectangular. If the assumption space is of finite dimension and is bounded, then the set can be approximated arbitrarily well (in a way that will be made precise) with a finite number of simulations. So, for the purpose of assumption mining, we impose a practical restriction that disturbance and state signals be of finite length because simulations are of finite length. This implicitly limits the set of specifications that can be mined to those whose satisfaction can be decided for signals of that length. For instance, given a signal \( x[\cdot] \) over \( F = [0, N] \), the specifications \( \Sigma_{[0,2N]}(x[\cdot]) \leq 1 \) and \( \phi_{[0,2N]}(x[\cdot]) \leq 1 \) are disallowed.

A natural question arises of how to intelligently generate disturbance inputs to simulate. This problem becomes more difficult as the signal length increases because the lower set resides in progressively larger dimension spaces. Ideally, one would like to represent the lower set as accurately as possible with a small number of sample points in the assumption space. We provide a modified algorithm that uses the Z3 satisfiability modulo theories (SMT) solver as an oracle to determine the location of the next query [8]. Legriel, Le Guernic, Cotton, and Maler take a similar approach when tackling the problem of estimating a Pareto front of a multi-criteria optimization problem by making queries to a satisfiability solver [15].

If we query a signal in the white region of Fig. 7a, then we are guaranteed to refine our approximation of the ordered set. A rectangle is encoded as an intersection of half planes, and its complement is a union over the opposite half planes. The white region is equivalent to the intersection of the complements of each rectangle. Thus, finding a point in that region can be posed as a conjunctive normal form-like satisfiability query, which we pose to an SMT solver.

The query may return a point that does not contribute to improving the boundary approximation such as the black dot in Fig. 7b. To explore the signal space effectively, we first "bloat" each rectangle along each side by a constant \( \epsilon \) immediately before sending the query to the SMT solver. The bloating factor is represented by the grey region in Fig. 7b, and this technique serves a similar purpose to tabu search in [9]. A satisfying query will return a point in the white region, whereas unsatisfiability signifies that the white region does not exist and the entire space \( X \times D[\cdot] \) is covered.

Unsatisfiability of the queried formula indicates that the rectangle bloating was too aggressive and covered the entire space; it serves as a convenient certificate for the quality of our approximation of the ordered set. Unsatisfiability with an \( \epsilon \) bloating indicates that every point on the boundary must lie within \( \epsilon \) coordinate-wise of a point in either the set of upper or lower points as in Fig. 7c. If a query is unsatisfiable, then we decrease the size of \( \epsilon \) by multiplying by a learning rate \( \alpha \) until our algorithm reaches a given desired precision \( \epsilon_{\text{final}} \) (see lines 2-7 of Algorithm 1) or a timeout.

### 5.2 Remarks on Tractability

If the assumption space is of finite dimension and is bounded, then a finite number of queries are needed for our algorithm to approximate the set to an arbitrary degree of precision \( \epsilon_{\text{final}} > 0 \). As a worst case scenario, our algorithm has an upper bound of \( \frac{\epsilon_{\text{final}}}{\epsilon} \) queries, where \( \epsilon \) is the dimension of the assumption set and \( d = \sup(||x - y||_{\infty}, x, y \in X \times D[\cdot]) \) is the diameter of the assumption space. If the assumption space and assumption set are not bounded, it may still be possible for the latter to have a finite \( \epsilon \)-approximation for all \( \epsilon > 0 \), but establishing this requires prior knowledge about the assumption set, the very object we are trying to identify.

As highlighted in our freeway example in Section 6 below, the aforementioned upper bound is rather conservative and the number of queries is typically orders of magnitude less. The variable bloating \( \epsilon \) effectively amounts to a way to vary the grid granularity. Picking a learning rate \( \alpha \) requires a delicate tradeoff between exploring the space earlier. A learning rate that is too conservative will result in slower convergence but better coverage of the space, whereas too aggressive a learning rate will rapidly increase the number of points in the grid.
The assumption mining algorithm consists of generalizing information from simulations (Fig. 7a) and systematic exploration of the assumption space (Fig. 7b). After the algorithm terminates, we have a certificate for the approximation’s quality (Fig. 7c).

Figure 7: The assumption mining algorithm consists of generalizing information from simulations (Fig. 7a) and systematic exploration of the assumption space (Fig. 7b). After the algorithm terminates, we have a certificate for the approximation’s quality (Fig. 7c).

Algorithm 1 ε-approximation of assumption $\Sigma^{-1}(\phi)$

**Input:** $\Sigma, \phi, \alpha \in (0, 1), \epsilon, \epsilon_{\text{final}}$

**Output:** $\text{lowerPts}, \text{upperPts}$

1. $\text{lowerPts} = \emptyset$, $\text{upperPts} = \emptyset$
2. while $\epsilon \geq \epsilon_{\text{final}}$
3. $(x_0, d_0) = \text{orderedQuery}(\text{lowerPts}, \text{upperPts}, \epsilon)$
4. if $(x_0, d_0)$ is NaN then
5. $\epsilon = \alpha \cdot \epsilon$
6. continue
7. end if
8. $x_0 = \Sigma(x_0, d_0)$
9. if $(x_0) = \phi$ then
10. $\text{lowerPts}.\text{append}\,(x_0, d_0)$
11. else
12. $\text{upperPts}.\text{append}\,(x_0, d_0)$
13. end if
14. end while
15. return $(\text{lowerPts}, \text{upperPts})$

Algorithm 2 Ordered Query

**Input:** $\text{lowerPts}, \text{upperPts}, \epsilon$

**Output:** $x, d_1$

1. $\text{smtConstraints} = \emptyset$
2. for all $(x_i, d_i) \in \text{lowerPts}$ do
3. rect = getLowerRect$(x_i, d_i)$
4. bloatedRect = bloatRect(rect, $\epsilon$)
5. $\text{smtConstraints}.\text{append}\,(\text{bloatedRect})$
6. end for
7. for all $(x_i, d_i) \in \text{upperPts}$ do
8. rect = getUpperRect$(x_i, d_i)$
9. bloatedRect = bloatRect(rect, $\epsilon$)
10. $\text{smtConstraints}.\text{append}\,(\text{bloatedRect})$
11. end for
12. satisfied = smt.solve($\text{smtConstraints}$)
13. if satisfied then
14. return $(x, d_1) = \text{smt.model}()$
15. else
16. return $(x, d_1) = (\text{NaN}, \text{NaN})$
17. end if

6. **EXAMPLES**

6.1 **Monotone Integrator Example**

Our first example illustrates the use of our mining algorithm for a simple system and short disturbance signals. Let $\Sigma$ be a discrete time system with $x \in \mathbb{R}_{\geq 0}, d \in \mathbb{R}$ and update equations

$$x[k + 1] = \max(0, ax[k] + d[k])$$

(9)

where $a = .4$. We let the initial state $x[0] = 0$. This system is monotone with respect to the standard ordering on $\mathbb{R}$. The specification $\phi = \Box_{[0, 1]}(\Diamond_{[0, 2]}(x(t) \leq 4.0))$ is a lower specification according to the rules outlined in Section 3.3 and it is either satisfied or violated with a state signal on the interval $[0, 3]$ and an input signal on the interval $[0, 2]$. Fig. 8 shows the approximate assumption set for $\phi$ with an absolute precision $\epsilon = .25$ with a total of 430 points.

6.2 **Freeway Example**

Our second example is of a freeway traffic network, where we seek to determine the limits of the assumption mining...
algorithm by introducing a large state space and long disturbance signals. Consider the network depicted in Fig. 9, which has a main stretch of three links \( x_0, x_2, x_4 \) and two on-ramps \( x_1, x_3 \). The dynamics are taken from the cell transmission model (CTM) [6][13], a macroscopic fluid-like model of freeway dynamics. Individual vehicles are not a component of this model. Each discrete time instant represents a five minute period in each of the five links. We overload notation and refer to links and their occupancy values using the same variable. The state update equations arise from conservation of mass:

\[
\begin{align*}
x_0[k + 1] &= \min(100, x_0[k] - f_0^\text{out}[k] + 30) \\
x_1[k + 1] &= \min(100, x_1[k] - f_1^\text{out}[k] + d_1[k]) \\
x_2[k + 1] &= \min(100, x_2[k] - f_2^\text{out}[k] + \sum_{i=0}^{5} f_i^\text{out}[k]) \\
x_3[k + 1] &= \min(100, x_3[k] - f_3^\text{out}[k] + d_3[k]) \\
x_4[k + 1] &= \min(100, x_4[k] - f_4^\text{out}[k] + \sum_{i=2,3} f_i^\text{out}[k])
\end{align*}
\]

where \( f_i^\text{out}[k] \) represents the flow exiting link \( x_i \) at time \( k \). The disturbances \( d_1[k], d_3[k] \) represent the number of vehicles that would like to enter the network via on-ramps \( x_1, x_3 \) and lie within a range [0, 30]. We assume link \( x_0 \) experiences a constant disturbance of 30 vehicles from the exogenous upstream link. The disturbance space is \( D = \prod_{i=1,3} \{0, 30\} \subset \mathbb{R}_{\geq 0}^2 \). The minimization terms above prevent the occupancy from exceeding the maximum capacity of the freeway segment.

The flows into and out of a link are determined by supply and demand. A link’s demand is the rate at which it would like to send vehicles to downstream links. The demand \( \Phi_i(x_i[k]) \) that link \( x_i \) exhibits is a non-decreasing function

\[
\Phi_i(x_i[k]) = \min(c_i, \alpha_i x_i[k])
\]

where \( c_i \) is a saturation rate and \( \alpha_i \in [0, 1] \) is the fraction of current vehicles that will leave \( x_i \). The primary links have saturation rates \( c_0 = c_2 = c_4 = 40 \) and on-ramps have saturation rates \( c_1 = c_3 = 30 \). Link \( x_i \) also exhibits a supply

\[
\Psi(x_i[k]) = 100 - x_i[k],
\]

which is the rate of incoming vehicles that it can accept from upstream. A link’s supply is partitioned among upstream links, with links \( x_2, x_4 \) allocating 80% of their supply to an upstream highway link and 20% to on-ramps. The flow out of a link \( x_i \) is the minimum between supply available to it and \( x_i \)’s demand:

\[
\begin{align*}
f_0^\text{out}[k] &= \min (8(100 - x_2[k]), 40, 5x_0[k]) \\
f_1^\text{out}[k] &= \min (2(100 - x_2[k]), 30, x_1[k]) \\
f_2^\text{out}[k] &= \min (8(100 - x_1[k]), 40, 5x_2[k]) \\
f_3^\text{out}[k] &= \min (2(100 - x_4[k]), 30, x_3[k]) \\
f_4^\text{out}[k] &= \min (40, x_4[k])
\end{align*}
\]

The exogenous link that is immediately downstream from \( x_4 \) can always accept up to 40 vehicles.

Congestion occurs when demand exceeds supply and the left term in the minimization is active; one can verify that congestion occurs on an upper set of the state space because the left term remains active after an increase in any \( x_i[k] \). Let the predicate \( \phi(x) \) be true if congestion is present. Our specification, \( \phi = \land_{i \in [0, T]} \phi_{i+1}((\neg \phi_i(x_i[k]))) \), requires that congestion be intermittent. Note that \( \phi \)’s satisfaction value can be determined for inputs over \( I = [0, T + 1] \) because the generated state trajectories are over \( I = [0, T] \). Because a maximum rate of 40 vehicles can exit the network but a net rate of 90 vehicles can enter via links \( x_0, x_1, x_3 \) at any time step, \( \phi \) can easily be violated with a determined adversary. The network in Fig. 9 was shown to exhibit monotone dynamics in [5].

We consider disturbance signals \( I = [0, T - 1] \) with signal
lengths $T \in \{2, 3, \ldots, 11\}$. The input is two dimensional and the state space has five dimensions, so the dimension of the assumption space is $2T + 5$ and the boundary of the assumption set can be as complex as a $2T + 4$ dimensional object. Fig. 11 shows how increasing the dimension of the problem presents a tradeoff between granularity and sample points. The precision $\epsilon$ is normalized along each axis; for example, $\epsilon = .1$ represents that demand signals $d_k[k]$, $d_3[k] \in [0, 30]$ have a granularity of a rate of three vehicles at each time $k$ and that $x_t[0] \in [0, 100]$ has granularity of ten vehicles. We opted to test our method for higher dimensions and less granularity (i.e. higher $\epsilon_{\text{final}}$) because obtaining a fine approximation of the assumption set doesn’t make sense when traffic model parameters are imprecise and the model does not incorporate higher-order dynamics. The learning rate was $\alpha = .95$ and the initial bloating factor was $\epsilon = .6$. Our algorithm required anywhere from 3 to 12 orders of magnitude less samples than the worst case scenario number of samples $(\frac{1}{2})^{2T+5}$.

Runtimes for $T = \{2, \ldots, 7\}$ are plotted in Fig 12. Larger dimensions did not achieve the desired precision by a mining algorithm timeout of 10 minutes. Experiments were run on a standard on a laptop with 8 GB memory and 2.4 GHz Intel Core i7 processor. As expected, the main factors that influenced the miner’s total run time was dimension and

![Figure 11: For assumption spaces of varying dimension, precision and number of samples are used as two stopping criteria, with the other criterion plotted after termination. Lower values are better for both precision and samples. The samples plateau in the lower graph appears because of a time out.](image1.png)

![Figure 12: For a fixed $\epsilon_{\text{final}} = .25$, the total solver runtime and average runtime for each SMT query. The increase in total solver time is primarily due to the fact that more points were required to cover the space with an $\epsilon$ bloating. The total solver time does not include overhead for declaring constraints and was a variable proportion of total mining time.](image2.png)

7. CONCLUSION

We have introduced directed specifications and shown how their structure can be exploited in the assumption mining problem for monotone dynamical systems. Future work will focus on seeking out other connections between classes of dynamical systems and temporal logic fragments. Viewing temporal logic specifications as geometric objects in a signal space could enable a common language to explore connections between objectives written as temporal logic specifications and traditional control theoretic properties on signals and dynamical systems.

8. REFERENCES


