Parametric Identification of Temporal Properties

Eugene Asarin[1], Alexandre Donzé[2], Oded Maler[2], and Dejan Nickovic[3]

[1] LIAFA, Université Paris Diderot / CNRS, Paris, France
[2] Verimag, Université Joseph Fourier /CNRS, Gières, France
[3] IST Austria, Klosterneuburg, Austria

Abstract. Given a dense-time real-valued signal and a parameterized temporal logic formula with *both* magnitude and timing parameters, we compute the subset of the parameter space that renders the formula satisfied by the trace. We provide two preliminary implementations, one which follows the exact semantics and attempts to compute the validity domain by quantifier elimination in linear arithmetics and one which conducts adaptive search in the parameter space.

1 Introduction

Much of discrete verification is concerned with evaluating behaviors (traces) generated by a *system model* against *specifications* that classify behaviors as good or bad. A similar approach is used in other engineering domains, where the system model is described using some modeling and numerical simulation framework. Such models, which semantically correspond to continuous or hybrid systems, generate finite traces (trajectories, waveforms, signals). The simulation traces are then evaluated according to some *performance measures*, which are typically *quantitative* in nature. Such trace evaluation procedures are integrated in the development cycle of the system, where each time a specification violation is found or a behavior of a poor performance is observed, the systems is modified or fine-tuned to achieve its correctness or improve its performance.

The above description fits well the development of *engineered* systems constructed from components with known input-output behavior. Simulation and verification are required only because the outcome of the *interaction* between these components is hard to predict beyond a certain complexity. The specifications describe at a high-level the *intended* functionality that we want the system to achieve.

In this work we tackle the *inverse* problem, namely, given a trace or a set of traces, find a specification that it satisfies. The procedure used to resolve this problem consists in *learning* from examples (*system identification*, *inductive inference*, *parameter estimation*), and can be very useful in the context of *experimental science* such as Biology where one wants to come up with a succinct and human intelligible *description* of experimentally observed data. This approach can also help in the design of systems that admit physical parts whose properties are characterized experimentally, for example, analog components in digital circuits, and be integrated in a framework for compositional reasoning based on *assume-guarantee* principles.

As a specification formalism, we adopt *signal temporal logic* (STL) introduced in [17] to express and monitor temporal properties of *dense-time real-valued* signals. We introduce PSTL, a parametric extension of STL, where *threshold constants* in numerical

inequalities as well as *delay bounds* in temporal operators can be replaced by parameters. Then, we solve the following problem: *Given a PSTL formula, find the range of parameters that render the formula satisfied by a given set of traces*. This work extends the pioneering work of Fages and Rizk [10] who identify parameter ranges for numerical predicates on top of a the discrete-time temporal logic LTL [22]. Our use of a dense-time logic, where time is handled arithmetically, rather than as a sequence of "ticks", makes the whole framework more robust to changes in sampling rates or integration steps. More importantly, it allows us to use parameters in the temporal operators and compute trade-offs between timing and magnitude parameters.

The rest of the paper is organized as follows. In Sect. 2 we present PSTL and its semantics in terms of validity domains. In Sect. 3 we show that validity domains for PSTL formulae relative to (interpolated) piecewise-linear signals are semilinear and show that they can be computed, in principle, by quantifier elimination. In Sect. 4 we move to an approximate computation based on adaptive sampling of the parameter space using recently-developed techniques for approximating Pareto fronts. We demonstrate the viability of the approach by computing the validity domains on a non-trivial example of a stabilization property with 3 parameters relative to a signal with 1024 sampling points. We conclude with a discussion of past and future work.

2   Parametric Signal Temporal Logic

Parametric signal temporal logic (PSTL) is based on the logic STL introduced in [17, 21, 18] for specifying and monitoring properties of real-valued continuous time signals, in particular those produced by analog circuits [13]. In the rest of the paper, we assume a time domain $\mathbb{T} = [0, \infty)$ (or a finite prefix of it) and traces (signals) of the form $x : \mathbb{T} \to \mathbb{R}^n$. We use $x[t]$ to denote the value of x at time t and $x_i[t]$ for the value of its ith coordinate.

We abuse the same variables $\{x_1, \ldots, x_n\}$ to speak of the value of the signal in the logical formulae. In addition we use two types of parameters, *magnitude* parameters $\{p_1, \ldots, p_g\}$ and *timing* parameters $\{s_1, \ldots, s_h\}$, ranging over their respective domains $\mathcal{P}$ and $\mathcal{S}$, say hyper-rectangles in $\mathbb{R}^g$ and $\mathbb{R}^h$, respectively. We use p and s for the vectors of all parameters. A *numerical predicate* μ is an inequality of the form $f(x) < \theta$ or $f(x) > \theta$ where f is a function from $\mathbb{R}^n$ to $\mathbb{R}$ and θ is a threshold which is either a constant c or a magnitude parameter p_i. We use I to denote an interval of the form (a, b), $(a, b]$, $[a, b)$, $[a, b]$, (a, ∞) or $[a, \infty)$ where each of a, b can be either a non-negative constant or a timing parameter s_i. When both bounds are constants we require $0 \le a < b$. A PSTL formula is then defined by the grammar

$$\varphi := \mu | \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \, \mathcal{U}_I \varphi_2$$

The usual *always* and *eventually* operators are defined as: $\Diamond_I \varphi \triangleq \text{true} \, \mathcal{U}_I \varphi$ and $\Box_I \varphi \triangleq \neg\Diamond_I\neg\varphi$.

A parameter valuation $(u, v) \in \mathbb{R}^g \times \mathbb{R}^h$ transforms a PSTL formula φ into an STL formula $\varphi_{u,v}$ obtained by substituting the values (u, v) in the parameters (p, s). We use the notation $\theta_{u,v}$ to denote the threshold obtained from θ by such a substitution and $I_{u,v}$ for the similar operation on the interval I.

The *polarity* $\pi(p, \varphi)$ of a parameter p with respect to a formula φ is positive if it is easier to satisfy φ as we increase the value of p and is negative if it is harder. Intuitively, magnitude parameters satisfy

$$\pi(p, f(x) < p) = + \quad \pi(p, f(x) > p) = -$$

and timing parameters satisfy

$$\pi(s, \varphi \, \mathcal{U}_{[b,s]} \psi) = + \quad \pi(s, \varphi \, \mathcal{U}_{[s,b]} \psi) = -$$

We assume henceforth that each parameter has either a positive or negative polarity and without loss of generality, that it is positive.

The semantics of a PSTL formula φ with respect to a signal x is given, following [10], in terms of a *validity domain* $D(x, \varphi) \subseteq \mathcal{P} \times \mathcal{S}$ consisting of all tuples (u, v) such that x satisfies $\varphi_{u,v}$ in the usual sense of STL satisfaction. To compute it we will need at intermediate stages extended validity domains of the form $d(x, \varphi) \subseteq \mathbb{T} \times \mathcal{P} \times \mathcal{S}$ consisting of all tuples (t, u, v) such that $(x, t) \models \varphi_{u,v}$. Then $D(x, \varphi) = \{(u, v) : (0, u, v) \in d(x, \varphi)\}$ consists of all parameter values that yield satisfaction at time zero.

Definition 1 (Validity Domain). *The validity domain of a formula φ with respect to a signal x is defined inductively as follows.*

$$
\begin{aligned}
d(x, f(x) < \theta) &= \{(t, u, v) : f(x(t)) < \theta_{u,v}\} \\
d(x, \varphi \wedge \psi) &= d(x, \varphi) \cap d(x, \psi) \\
d(x, \neg\varphi) &= \overline{d(x, \varphi)} \\
d(x, \varphi \, \mathcal{U}_I \psi) &= \{(t, u, v) : \exists t' \in t \oplus I_{u,v} \text{ s.t. } (t', u, v) \in d(x, \psi) \wedge \\
&\qquad\qquad \forall t'' \in [t, t'](t'', u, v) \in d(x, \varphi)\}
\end{aligned}
$$

$$
d(x, f(x) < p) = \{(t, u, v) : \bigvee_{j=0}^{k-1} (t_j < t < t_{j+1}) \wedge (\alpha_j t + \beta_j < u)\}
$$

which is semilinear. For the inductive case, closure under Boolean operations is immediate. For the *until* operator observe that $d(x, \varphi \, \mathcal{U}_{[s_1, s_2]} \psi)$ can be written as

$$
\{(t, u, v) : \exists t' \, (t + v_1 \leq t' \leq t + v_2) \wedge (t', u, v) \in d(x, \psi) \wedge \\
\forall t''(t \leq t'' \leq t') \Rightarrow (t'', u, v) \in d(x, \varphi)\}
$$

Note that in the terminology of machine learning and inductive inference, our whole setting is that of learning from *positive* examples: we observe traces that occur but nobody gives us impossible traces. Hence it is natural to look for the *minimal* elements of the validity domain that yield the tightest (strongest) formulae satisfied by the traces.

3 Computing Validity Domains

In this section, we present a procedure for exact computation of validity domains for a given trace and PSTL formula, and illustrate it with a simple example. Finally, we present experimental results that indicate how this exact technique scales both with respect to the size of the input traces and the size of the PSTL formula.

3.1 Semilinear Validity Domains

To start with, observe that the semantics of STL formulae is defined in terms of *dense-time* real-valued signals, but in reality the signals that one can observe, either experimentally or via numerical simulators, are *sampled* signals consisting of sequences of time stamped values of the form

$$(t_0, x[t_0]), (t_1, x[t_1]), \ldots, (t_k, x[t_k]). \tag{1}$$

for an increasing sequence of time stamps with $t_0 = 0$. We interpret these sampled signals as continuous-time signals using linear interpolation as in [18]. In each interval of the form $[t_j, t_{j+1}]$ we consider the value of $x[t]$ to be

$$x[t] = x[t_j] + \frac{x[t_{j+1}] - x[t_j]}{t_{j+1} - t_j} \cdot t = \beta_j + \alpha_j t.$$

Definition 2 (Semilinear Validity Domains). *A subset of the parameter space is semilinear if it can be written as a Boolean combination of linear inequalities on the corresponding variables.*

Proposition 1. *For every PSTL formula φ and piecewise-linear signal x, the validity domain $D(x, \varphi)$ is semilinear.*

Proof. We first prove that $d(x, \varphi)$ is semilinear for every φ by a simple induction on the structure of the formula. For the base case of a predicate $f(x) < p$ we first construct from x a derived sampled signal $y = (t_0, y[t_0]), (t_1, y[t_1]), \ldots$ with $y[t_j] = f(x[t_j])$ that by interpolation is extended to the real time axis to obtain $y[t] = \alpha_j t + \beta_j$ whenever $t \in [t_j, t_{j+1}]$. Then the validity domain can be written as

$$d(x, f(x) < p) = \{(t, u, v) : \bigvee_{j=0}^{k-1} (t_j < t < t_{j+1}) \wedge (\alpha_j t + \beta_j < u)\}$$

which is semilinear. For the inductive case, closure under Boolean operations is immediate. For the *until* operator observe that $d(x, \varphi \, \mathcal{U}_{[s_1, s_2]} \psi)$ can be written as

$$\{(t, u, v) : \exists t' \, (t + v_1 \le t' \le t + v_2) \wedge (t', u, v) \in d(x, \psi) \wedge$$
$$\forall t''(t \le t'' \le t') \Rightarrow (t'', u, v) \in d(x, \varphi)\}$$

and since semilinear sets are closed under universal and existential projection (quantifier elimination) and $d(x, \varphi)$ and $d(x, \psi)$ are semilinear by the inductive hypothesis, the result follows. Finally, transforming d to D by projecting on $t = 0$ also preserves semilinearity. ∎

Note that a function f appearing in a predicate need not be necessarily linear. The result also holds when each f is linear and parameters are allowed as coefficients. In the discrete time logic used in [10], the restriction of parameters to threshold will lead to rectangular validity domains. The extension of Proposition 1 to validity domains associated with *several* signals is trivial: $D(\{x, x'\}, \varphi) = D(x, \varphi) \cap D(x', \varphi)$.

3.2 Example

Let us illustrate the computation of validity domains on the formula $\varphi = \Diamond_{[0,s_2]}\Box_{[0,s_1]}(x < p)$ and some of its variants and subformulae relative to the signal x of Fig. 3.2-(a). The formula admits two temporal parameters s_1 and s_2 and a magnitude parameter p. The validity domain $V_1 = d(x, x < p)$, depicted in Fig. 3.2-(b), is

$$
\begin{aligned}
V_1 = &(t \geq 0 \wedge t < 2 \wedge 2p > 4t) & \vee \\
&(t \geq 2 \wedge t < 4 \wedge 2p + 4t > 16) & \vee \\
&(t \geq 4 \wedge t < 5 \wedge p > 2t - 8) & \vee \\
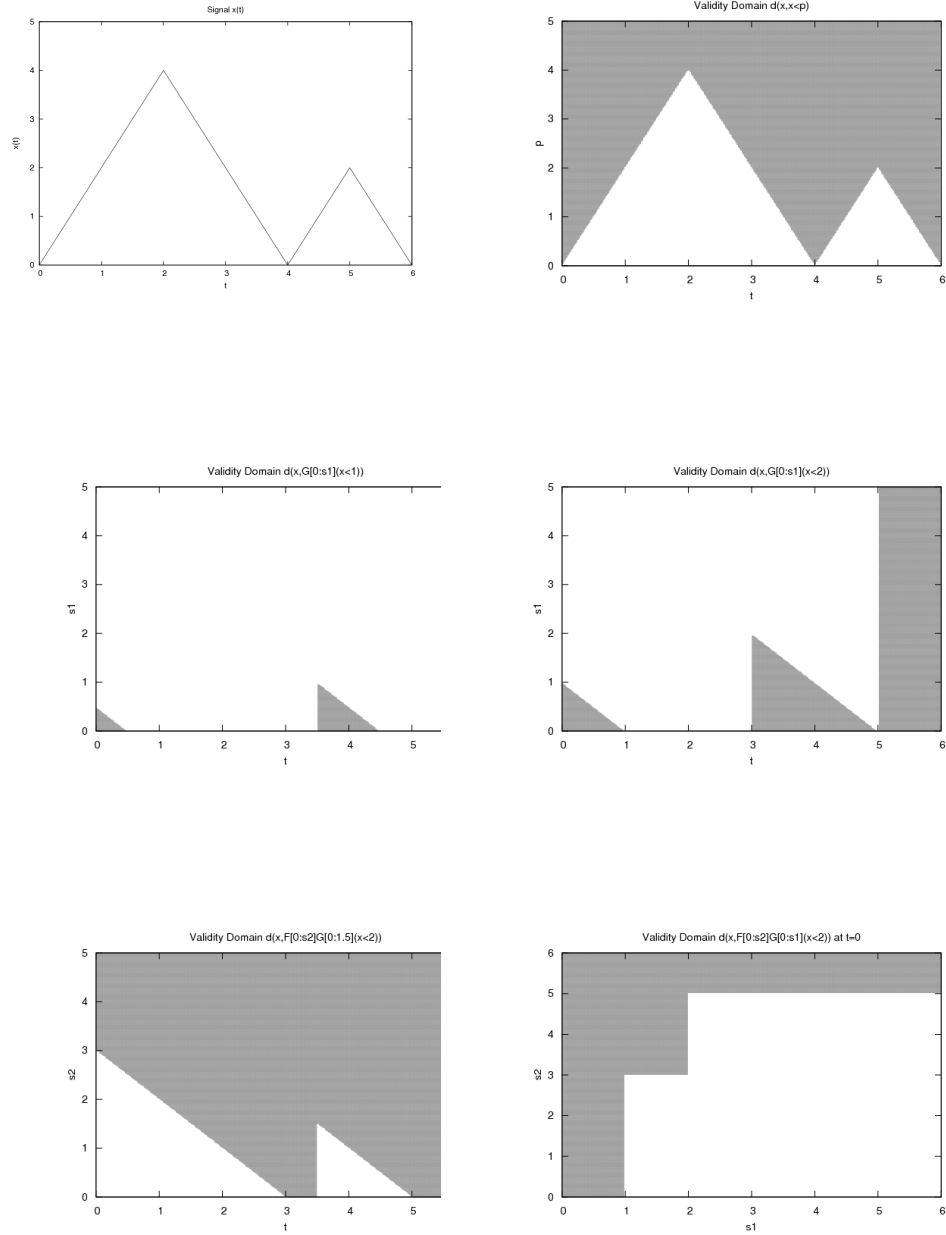&(t \geq 5 \wedge t < 6 \wedge p + 2t > 12)
\end{aligned}
$$

The validity domain $V_2 = d(x, \Box_{[0,s_1]}(x < p))$, which by definition is the set $\{(t, p, s_1) \mid \forall t' \in [t, t + s_2] \cap [0, 6), (t', p, s_1) \in d(x, x < p)\}$, is obtained by eliminating the universal quantifier, yielding a validity domain expressed by:

$$
\begin{aligned}
V_2 = &(p + 2s_1 + 2t < 12 \vee p + 2t > 12 \vee p > 0 \vee p \leq 0) \wedge \\
&(p + 2s_1 + 2t < 8 \vee p + 2t > 8 \vee p + 4 \leq 0 \vee p > 4) \wedge \\
&(s_1 + t \geq 6 \vee (p - 2s_1 - 2t > 0 \wedge s_1 + t < 2) \vee \\
&(p + 2s_1 + 2t > 8 \wedge s_1 + t \geq 2 \wedge s_1 + t < 4) \vee \\
&(p - 2s_1 - 2t + 8 > 0 \wedge s_1 + t \geq 4 \wedge s_1 + t < 5) \vee \\
&(p + 2s_1 + 2t > 12 \wedge s_1 + t \geq 5)) \wedge (p \geq 2 \vee s_1 + t < 5 \vee t \geq 5) \wedge \\
&(p > 0 \vee s_1 + t < 4 \vee t \geq 4) \wedge (p \geq 4 \vee s_1 + t < 2 \vee t_1 \geq 2) \wedge \\
&(p > 0 \vee s_1 + t < 6 \vee t \geq 6)
\end{aligned}
$$

Figures 3.2-(c,d) depict the projections of V_2 on $p = 1$ and $p = 2$, respectively. Finally the validity domain of the top-level formula, $V_3 = d(x, \Diamond_{[0,s_2]}\Box_{[0,s_1]}(x < p))$, which is the set $\{(t, p, s_1, s_2) \mid \exists t' \in [t, t + s_2] \cap [0, 6) \text{ s.t. } (t', p, s_1, s_2) \in V_2\}$, is obtained by eliminating the existential quantifier. The projection of V_3 on $t = 0$ and $p = 2$ yields the domain expressed by the following quantifier-free formula:

$$
\begin{aligned}
V_3 = &(s_1 + s_2 \geq 5 \wedge 0 \leq s_1 < 2 \wedge s_2 \geq 0) \vee \\
&(s_1 + s_2 > 5 \wedge s_1 \geq 0 \wedge s_2 > 5) \vee \\
&(s_1 + s_2 \geq 4 \wedge s_1 + s_2 < 5 \wedge s_1 \geq 0 \wedge s_2 > 3) \vee \\
&(s_1 + s_2 > 3 \wedge s_1 + s_2 < 4 \wedge s_1 \geq 0 \wedge s_2 > 3) \vee \\
&(s_1 \geq 0 \wedge s_2 \geq 6) \vee (s_1 < 1 \wedge s_1 \geq 0 \wedge s_2 \geq 0) \vee \\
&(s_1 + s_2 < 1 \wedge s_1 \geq 0 \wedge s_2 \geq 0)
\end{aligned}
$$

The projections of V_3 on $(s_1 = 1.5 \wedge p = 2)$ and on $(t = 0 \wedge p = 2)$ are shown Figures 3.2-(e) and 3.2-(f), respectively.

Fig. 1. (a) Signal x; (b) $d(x, x < p)$; (c) $d(x, \Box_{[0,s_1]}(x < 1))$; (d) $d(x, \Box_{[0,s_1]}(x < 2))$; (e) $d(x, \Diamond_{[0,s_2]}\Box_{[0,1.5]}(x < 2))$; and (f) $D(x, \Diamond_{[0,s_2]}\Box_{[0,s_1]}(x < 2))$
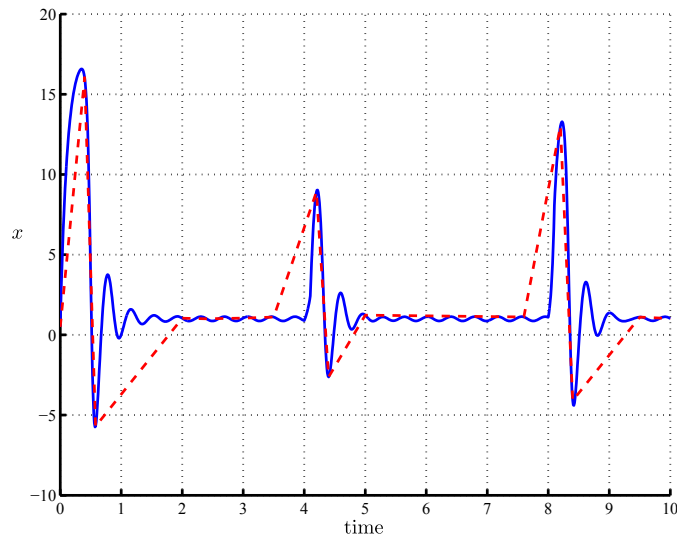
3.3 Experimental Results

We have implemented the above semantics using the linear quantifier elimination procedure of the tool Redlog [14]. It should be noted that our implementation consists of a straightforward invocation of the elimination procedure with no attempt to tailor and tune the procedure to the specificity of our problem (see further discussion on Sect. 5). As a benchmark we use the following very typical stabilization (disturbance rejection) property:

$$\varphi_{st} : \Box((x \geq p) \rightarrow \Diamond_{[0,s_2]}\Box_{[0,s_1]}(x < p)). \tag{2}$$

The property speaks of a controlled signal which is required in normal conditions to stay below a threshold p. If due to some disturbance the signal is driven above p, than the control system should stabilize it with s_1 time, that is, drive it again below p, and moreover, stay below p for at least s_2 time. Characterizing the parameters (delays and amplitudes) of such a behavior is relevant for many systems ranging from heart pacemakers to cooling systems in nuclear power plants.

We find validity domains for this formula relative to the signal x_{st} of Fig. 2 represented by $k = 1024$ sampling points. Since the complexity of the validity domain and quantifier elimination depends on k we apply our procedure to various under-samplings of x_{st}, see Fig. 2. Of course, below some sampling resolution, the signal loses its characteristics and the results become less meaningful.

Fig. 2. The signal x_{st} against which the stability property is tested: 1024 sampling points (continuous blue) and 16 sampling points (dashed red).

Table 3.3 shows some statistics on computation time and description size of the validity domain for the formula φ_{st} and its subformulae

$$\varphi_1 : \Box_{[0,s_1]}(x < p)$$
$$\varphi_2 : \Diamond_{[0,s_2]}\Box_{[0,s_1]}(x < p)$$
$$\varphi_3 : (x \geq p) \rightarrow \Diamond_{[0,s_2]}\Box_{[0,s_1]}(x < p)$$

against various sampled versions of x_{st}. The size of the solution corresponds to the number of linear inequalities used for its representation (no redundancy elimination applied at this point) and the symbol $*$ denotes a time-out after 10 minutes.

formula	φ_1		φ_2		φ_3		φ_{st}	
k	time(s)	size	time(s)	size	time(s)	size	time(s)	size
8	0.02	38	0.11	197	0.17	207	3	4219
16	0.10	66	0.81	855	0.74	375	83.79	37709
32	0.26	86	19.07	6553	18.27	2885	*	*
64	4.16	144	341.95	23103	308.93	10258	*	*
128	68.29	895	*	*	*	*	*	*
256	386.72	3098	*	*	*	*	*	*

Table 1. Computation time and description size for the stabilization formula φ_{st} and its subformulae for different sampling of signal x_{st}.

Note that in the worst case, the Fourier-Motzkin quantifier elimination procedure may square the number of constraints which gives a description size of k^{2^m} where m is the number of nested simple ($\Box$ or $\Diamond$) temporal operators, not counting the normalization of the formula after each iteration.
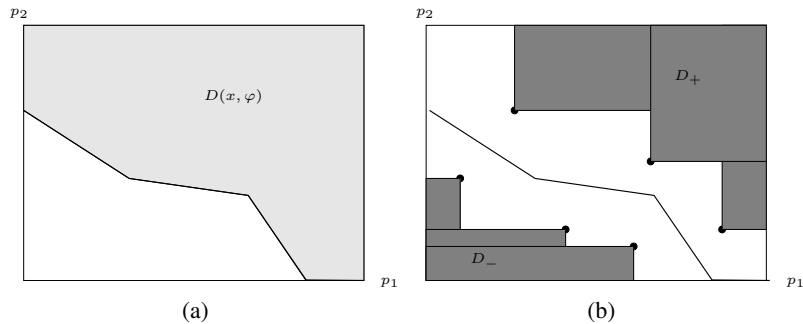
4 Approximating Validity Domains

The limitations of the exact method motivate us to apply an alternative approximation technique based on intelligent search in the parameter space. For every point (u, v) in the parameter space we can pose a *query* concerning its membership in $D(x, \varphi)$ by constructing the STL formula $\varphi_{u,v}$ and checking whether $x \models \varphi_{u,v}$. This approach to parameter space exploration has been implemented in a tool [5] and applied to embedded [7] and biological [6] case studies. To conduct this exploration efficiently we will take advantage of an additional property of our validity domains due to the use of a fixed polarity for each parameter.

Definition 3 (Monotonic Validity Domains). *A subset $V \subseteq \mathcal{P} \times \mathcal{S}$ is monotonic if for every i, whenever a parameter valuation $(v_1, \ldots, v_i, \ldots, v_{g+h})$ is in V so is any $(v_1, \ldots, v_i', \ldots, v_{g+h}) \in \mathcal{P} \times \mathcal{S}$ satisfying $v_i' > v_i$ (when $\pi(p_i, \varphi) = +$) or $v_i' < v_i$ (when $\pi(p_i, \varphi) = -$).*
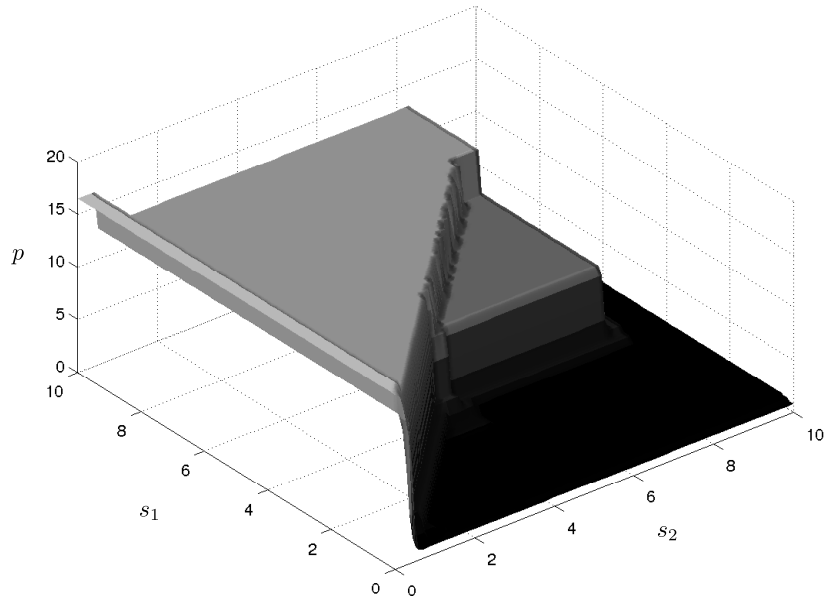
To facilitate the discussion we apply a coordinate transformation to the parameter space and replace every negative polarity parameter p by its complement $-p$ and thus deal with validity domains which are *upward closed* relative to the parameter space, namely $v \in V$ implies $v' \in V$ for every $v' > v$. The set of minimal parameter values that render the formula satisfied is the boundary between the validity domain and its complement relative to the parameter space. Such sets are known in the context of multi-criteria optimization [9] as *Pareto surfaces* or *Pareto fronts*, see Fig. 3-(a). An ϵ-approximation of the surface is a set of points $S \subseteq V$ such that that each point on the surface admits an ϵ-close point in S. In other words, the set S consists of a representative sample of the optimal trade-offs available in the problem. In the following we describe briefly the exploration technique developed in [15] for efficient approximation of Pareto fronts, which constitutes a multi-dimensional generalization of binary search.

Fig. 3. (a) An upward-closed validity domain in 2 dimensions and its lower boundary (thick line); (b) state of knowledge after 3 positive and 3 negative queries in the parameter space.

Figure 3-(b) depicts our state of knowledge after performing 3 positive and 3 negative queries in the parameter space. Since the set is upward closed, we know that the upward closure of the positive points (the set D_+) is included in $D(x, \varphi)$ while the downward closure of the negative points (the set D_-) is included in the the complement of $D(x, \varphi)$. The frontier that we look for is situated between these two sets, and the *distance* between their boundaries gives and upper bound the quality of the approximation (ϵ) provided by the set of positive points. Orienting subsequent queries to points in the parameter space that reduce this distance provides for focusing the queries on the boundary, see more details in [15]. Exponentiality in the dimension of the parameter space cannot be, of course, avoided but the time for each query is linear in k. We have implemented a search based approach to the example, and Fig. 4 depicts the surface obtained for φ_{st} and the 1024-points version of x_{st}.

Fig. 4. Approximate boundary of the validity domain $D(x_{st}, \varphi_{st})$ for the stabilization formula φ_{st} with parameters s_1, s_2 and p, for signal x_{st} of Fig. 2 with $k = 1024$.

5 Discussion

We have shown how to synthesize magnitude and timing parameters in a quantitative temporal logic formula so that it fits observed data. The only similar work we are aware of is that of [10] that we extend by making the temporal dimension quantitative and hence parameterizable. This line of work should not be confused with other types of "temporal queries", e.g. [4] where a parametric temporal formula contains a "placeholder" that needs to be replaced by a proposition resulting in a formula that satisfies a given model. In the context of real-time model checking, the decision problems for parametric timed automata and parametric extension of a real-time temporal logic MITL were studied in [12, 3].

We consider the following extensions of this work in order to enlarge its scope both in terms of problem size and richer settings. We are investigating specialized ways to organize the quantifier elimination process so as to proceed along the time axis, in the same manner as qualitative [18] and quantitative [8] satisfaction is computed. A particular difficulty here is that validity domains do not decompose naturally into time segments, that is, a disjunction where each disjuncts admits a distinct term of the form $a < t < b$, but rather segments of the from $a < t + s < b$ for a temporal parameters s. Another technical problem to solve is the efficient derivation of the semilinear formula characterizing the minimal facets of a non-convex validity domain. To this end we intend to employ the novel quantifier elimination techniques of [19, 20].

Although the restriction to parameters of fixed polarity is justified in many cases and simplifies life, one can imagine situations where it should be dropped, for example in a predicate of the form $p + a < x < p + b$ where the value of x is constrained to be in an interval of a fixed size but a parameterized displacement. Likewise we may have parameterized temporal intervals of the form $[s + a, s + b]$. In such situations, semilinearity is preserved but not monotonicity. Other relaxation of fixed polarity may be required in the context of parameters in nonlinear functions. In the absence of monotonicity, finding the minimal set of parameters is not the only natural choice. In fact, one my argue on the contrary, that it is safer to pick parameters which are deep inside the validity domain as they provide for more *robust* [23, 11, 8] satisfaction. Since tightness and robustness are conflicting goals perhaps the best solution would be to provide trade-offs (Pareto points) between the two.

The work presented in this paper was fully parametric in the sense that the template formula φ is given and only parameters were sought. A more ambitious goal would be to combine it with a search in the space of formula templates. While such a solution will bring us closer to the science fiction scenario of automatic derivation of theories from experiments, it is clear that it is very easy to face a combinatorial explosion if the search space is not restricted to some small class of property templates. For for example one may consider response properties of the form $\Box(\varphi \Rightarrow \Diamond_I \psi)$ where both φ and ψ are Boolean combinations of a small number of simple predicates.

In the more general context, the technique presented here may occupy an interesting niche in all domains that deal with this kind of reverse engineering, e.g. *system identification* [16], machine learning [2] or *inductive inference* [1]. In all these areas one wants to generalize from observations and find a mathematical model compatible with them. In the context of signals, one can think of two extreme classes of target models: detailed models of *dynamical systems* that produce traces which are close to the observed ones or more abstract *logical theories* that define logical dependencies between observations. Temporal logic [22], which is a logic tailored for describing dynamic behaviors, augmented with *quantitative* constructs in time and space as in STL, can offer an interesting tradeoff between the over determination of dynamic models and the quantitative vagueness of too abstract logical statements such as *A causes B* that are sometimes used to summarize experimental findings in the life sciences. A temporal formula expressing the quantitative temporal constraints between the evolution of real-valued observed quantities might provide an optimal level of detail in some application domains.

References

1. D. Angluin and C. H. Smith. Inductive inference: Theory and methods. *ACM Comput. Surv.*, 15(3):237–269, 1983.
2. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer Verlag, 2006.
3. L. Bozzelli and S. La Torre. Decision problems for lower/upper bound parametric timed automata. In *ICALP*, pages 925–936, 2007.
4. W. Chan. Temporal-locig queries. In *CAV*, pages 450–463, 2000.
5. A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *CAV*, pages 167–170, 2010.

6. A. Donzé, G. Clermont, and C. J. Langmead. Parameter synthesis in nonlinear dynamical systems: Application to systems biology. *Journal of Computational Biology*, 17(3):325–336, 2010.
7. A. Donzé, B. H. Krogh, and A. Rajhans. Parameter synthesis for hybrid systems with an application to simulink models. In *HSCC'09*, LNCS. Springer-Verlag, April 2009.
8. A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *FORMATS 2010*, volume 6246 of *LNCS*, pages 92–106. Springer, 2010.
9. M. Ehrgott. *Multicriteria optimization*. Springer Verlag, 2005.
10. F. Fages and A. Rizk. From model-checking to temporal logic constraint solving. In *CP*, pages 319–334, 2009.
11. G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.
12. B. Di Giampaolo, S. La Torre, and M. Napoli. Parametric metric interval temporal logic. In *LATA*, pages 249–260, 2010.
13. K. D. Jones, V. Konrad, and D. Nickovic. Analog property checkers: a ddr2 case study. *Formal Methods in System Design*, 36(2):114–130, 2010.
14. A. Lasaruk and T. Sturm. Effective quantifier elimination for presburger arithmetic with infinity. In *CASC*, pages 195–212, 2009.
15. J. Legriel, C. Le Guernic, S. Cotton, and O. Maler. Approximating the Pareto front of multi-criteria optimization problems. In *TACAS 2010*, volume 6015 of *LNCS*, pages 69–83. Springer, 2010.
16. L. Ljung. *System Identification - Theory For the User*. Prentice Hall, 1999.
17. O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *FORMATS/FTRTFT*, pages 152–166, 2004.
18. O. Maler, D. Nickovic, and A. Pnueli. Checking temporal properties of discrete, timed and continuous behaviors. In *Pillars of Computer Science*, pages 475–505, 2008.
19. D. Monniaux. A quantifier elimination algorithm for linear real arithmetic. In *LPAR'08*, number 5330 in LNCS, pages 243–257. Springer Verlag, 2008.
20. D. Monniaux. Automatic modular abstractions for linear constraints. In *POPL'09*, pages 140–151. ACM, 2009.
21. D. Nickovic and O. Maler. AMT: A property-based monitoring tool for analog systems. In *FORMATS*, pages 304–319, 2007.
22. A. Pnueli. The Temporal Semantics of Concurrent Programs. *Theoretical Computer Science*, 13:45–60, 1981.
23. A. Rizk, G. Batt, F. Fages, and S. Soliman. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *CMSB*, pages 251–268, 2008.

Appendix: Parameter Polarity

Let $\top, +, -$ and $\bot$ indicate, respectively, undefined, positive, negative and mixed polarities. The polarity of a magnitude parameter p in a formula φ is defined inductively as follows.

$$\pi(p, f(x) < c) = \pi(p, f(x) > c) = \top$$
$$\pi(p, f(x) < p) = + \quad \pi(p, f(x) > p) = -$$
$$\pi(p, \neg\varphi) = \sim \pi(p, \varphi)$$
$$\pi(p, \varphi \, \mathcal{U}_I \psi = \pi(p, \varphi \wedge \psi) = \pi(p, \varphi) \circ \pi(p, \psi)$$

For a timing parameter s we have

$$\pi(s, \mu) = \top$$
$$\pi(s, \varphi \, \mathcal{U}_I \psi) = u \circ (\pi(p, \varphi) \circ \pi(p, \psi))$$

where
$$u = \begin{cases} + \text{ when } I = [a, s] \\ - \text{ when } I = [s, b] \\ \top \text{ otherwise} \end{cases}$$

The rules for negation and conjunction are identical to the rules for magnitude parameters. Operations $\sim$ and $\circ$ are defined as

$\circ$	$\top$	$+$	$-$	$\perp$
$\top$	$\top$	$+$	$-$	$\perp$
$+$	$+$	$+$	$\perp$	$\perp$
$-$	$-$	$\perp$	$-$	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

	$\sim$
$\top$	$\top$
$+$	$-$
$-$	$+$
$\perp$	$\perp$

A formula is fine if the polarity of every parameter is either $+$ or $-$. We consider only fine formulae.