Towards Fast Parametric Identification for STL

We work on a new algorithm for parametric identification in signal temporal logic. For a real-valued signal and a parameterized temporal logic formula, we want to find the set of parameter values that makes the formula satisfied by the signal. We have a working algorithm for identification of spatial parameters in piecewise-constant signals, which performs well in our experiments. Identification of time parameters is an ongoing work.

Introduction and Background

Signal temporal logic (STL) [3] is designed to handle real-valued dense-time signals and describe behaviors of continuous and hybrid systems. The problem of parametric identification in STL is introduced in [1]. It uses PSTL, a version of STL, admitting formulas where some of the constants are replaced by parameters. Every set of parameter values $v \in V$ transforms a PSTL formula into an STL formula. The problem is to compute the *validity domain* of a PSTL formula relative to a signal, i.e., the set of parameter valuations that make the formula satisfied by the signal. Such a procedure can be used in principle, to derive a concise description of the input-output relation of a large SPICE-level circuit model based on simulation traces. In [1], two approaches are proposed for *piecewise-linear* signals: using quantifier elimination in a formula of the size linear in the length of the signal, and using search. We aim to produce an efficient algorithm for *piecewise-constant* signals. One can view it as a specialized quantifier elimination procedure exploiting the structure of the problem, e.g., that input is ordered by time.

The syntax of PSTL is as follows: $\varphi ::= \mathbf{true} \mid x \le a \mid x \ge a \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{F}_{[a,b]}\varphi \mid \varphi_1 \,\mathbf{U}\, \varphi_2$ where x is a signal component, and a, b are real constants or parameters. We restrict a parameter to have fixed type (*space* parameters appear in inequalities; *time* parameters – in temporal windows) and polarity (in the negation normal form, *positive* parameters appear as upper bounds; *negative* – as lower bounds).

Identifying Space Parameters

Our algorithm works by induction on the formula structure. For every sub-formula we build the *validity signal*. For every time point, it gives the set of parameter values that make the subformula satisfied by the signal, starting from that time point. Here, we show how this works for the *atomic comparison* $x \le p$ and *timed eventually* $\mathbf{F}_{[a,b]}\varphi$ where a, b are constants. In Fig. 1, we show an example of a piecewise signal component x, taking values c_1 through c_3 as time progresses.

Atomic Comparison For an atomic comparison $x \le p$, for a segment where x equals c_i, we assign the validity set $p \ge c_i$. This is shown in Fig. 2.

Timed Eventually For a $\mathbf{F}_{[a,b]}\varphi$ where a, b are constants, parameter identification can be solved as a quantifier elimination problem. We show an example of that in Fig. 3. In the figure, the validity signal of φ (in the top) can be seen as a disjunction: $(0 = t_0 \le t < t_1 \wedge V_1) \vee \cdots \vee (t_2 \le t \le t_3 \wedge V_4)$. Then, the validity signal of $\mathbf{F}_{[a,b]}\varphi$ can be seen as $\exists t.\ (t + a \le t' \le t + b) \wedge (t_0 \le t' < t_1 \wedge V_1) \vee \cdots \vee (t_2 \le t' \le t_3 \wedge V_3)$, which after quantifier elimination becomes $(t_0 - b \le t < t_1 - a \wedge V_1) \vee \cdots \vee (t_2 - b \le t \le t_3 - a \wedge V_3)$. The intervals $[t_{i-1} - b, t_i - a)$ in general overlap, and in our algorithm, we create from them a partition of the time domain, taking union of the validity sets, where intervals overlap, and merging

Submitted to:
FAC 2018

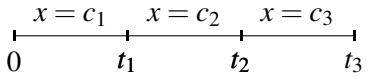

Figure 1: A component of a piecewise-constant signal.

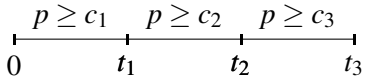

Figure 2: Validity signal for the expression $x \leq p$.

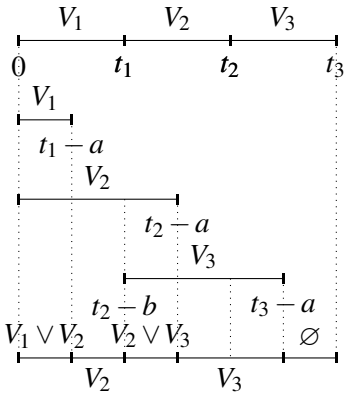

Figure 3: Constructing the validity signal for $\mathbf{F}_{[a,b]}\varphi$.

consecutive intervals labelled by identical validity sets. This is shown in the bottom of Fig. 3. In our setting, every interval is mapped to a union of upward-closed rectangles, with dimensions of a rectangle corresponding to parameters. The main technical challenges here are computing unions of validity sets over a running window (we use a variant of Lemire's algorithm [2]), and efficient manipulation of Pareto-like validity sets (planned for future work). When resulting validity sets have small representations, our implementation can process signals with millions of intervals in under a minute.

Identifying Time Parameters

With time parameters, validity signals can still be piecewise-constant, if we allow validity sets to be *template polyhedra*. Then temporal operators will induce templates that are expressions over time parameters and absolute time. Fig. 4 shows an example of constructing the validity signal for the expression $\mathbf{F}_{[a,p]}\varphi$ where p is a parameter and φ does not contain time parameters. The upper bound of the temporal window p, instead of affecting the start time of the shifted intervals, now creates constraints on $t + p$, where t is the value of time; further operations will also create constraints on just p. We are now working on an algorithm that would compute validity signals for different combinations of parameters in temporal windows and in validity signals. In general, a temporal operators with a time parameter creates two new templates in the validity sets and changes existing templates that depend on t.

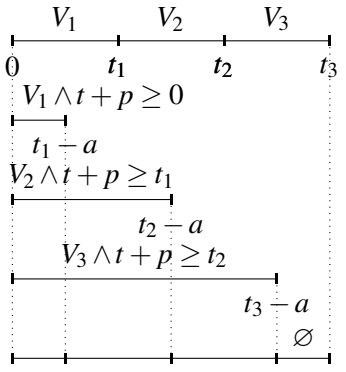

Figure 4: Constructing the validity signal for $\mathbf{F}_{[a,p]}\varphi$.

References

[1] Eugene Asarin, Alexandre Donzé, Oded Maler & Dejan Nickovic (2011): *Parametric identification of temporal properties*. In: *Runtime Verification*, Springer, pp. 147–160.

[2] D. Lemire (2006): *Streaming Maximum-Minimum Filter Using No More than Three Comparisons per Element*. *CoRR* abs/cs/0610046.

[3] Oded Maler & Dejan Nickovic (2004): *Monitoring Temporal Properties of Continuous Signals*. In: *FORMATS/FTRTFT*, pp. 152–166.