Mining Weighted Requirements from Closed-Loop Control Models

Xiaoqing Jin[1], Alexandre Donzé[2], and Gianfranco Ciardo[1]

[1] University of California, Riverside
{jinx,ciardo}@cs.ucr.edu
[2] University of California, Berkeley
donze@eecs.berkeley.edu

Abstract. Specification defects are a major source of ambiguities and errors. We propose a new weighted requirement mining framework to address this problem. We define weighted and parametric weighted signal temporal logic (STL) as the underlying formalism to improve the expressiveness. This also improves convergence of the mining process. We provide some primitive results to show the benefit of the new framework.

1 Introduction

Specification defects are one of the most important issues when verifying the design of a system. Ideally, design requirements should range from higher system-level specifications describing desired performance in a suitable formal logic, to lower implementation-level specifications accurately describing expected inputs and outputs, as well as the functionality of each component. A good system-level requirement for a transmission controller design could be "the car should not shift from gear 2 to gear 1 and back to gear 2 within 0.8 seconds". In reality, industry requirements are rarely documented in a formal language, but are instead given as informal and vague statements, often in natural language. Occasionally, they are open-ended statements, such as "the idle engine speed should be less than 3000rpm", or assume common knowledge, making it difficult to evaluate test results. Moreover, industrial applications are often saddled with large amounts of undocumented legacy code and legacy models.

To address this problem, we proposed a general requirement mining framework [4] that applies temporal logic and is designed for signals to guide counterexample search and parameter synthesis. A counterexample-guided refinement procedure mines requirements from a closed-loop model. In this framework, formalisms such as metric temporal logic (MTL) [5] and signal temporal logic (STL) [6] can effectively capture both the real-valued and the time-varying behavior of hybrid control systems and express the requirements. Parametric signal temporal logic (PSTL) [1], an extension of STL, is suited to express template requirements to be mined. However, STL, MTL, and PSTL do not distinguish the different contributions associated with each component constraint in a complex requirement or consider the effect of the different units of various signals. Thus, we propose new weighted temporal logics with better expressiveness: weighted STL (WSTL) and parametric weighted STL (PWSTL).

2 Preliminaries

A *signal* is a function mapping the time domain $\mathbb{T} = \mathbb{R}^{\geq 0}$ to the reals $\mathbb{R}$. *Boolean signals* restrict their values to *false* (denoted $\bot$) and *true* (denoted $\top$). Vectors are elements of $\mathbb{R}^n$, $n > 1$ (denoted in bold). A *trace* $\mathbf{x}$ is a multi-dimensional signal, i.e., for $t \in \mathbb{T}$, $\mathbf{x}(t) = (x_1(t), \cdots, x_n(t)) \in \mathbb{R}^n$. A *system* $\mathcal{S}$ maps input signals $\mathbf{u}(t)$ to output signals, $\mathbf{x}(t) = \mathcal{S}(\mathbf{u}(t))$, simply written as $\mathbf{x} = \mathcal{S}(\mathbf{u})$.

Signal temporal logic (STL) [6] represents specification *predicates* using inequalities μ of the form "$f(\mathbf{x}) \sim \pi_{const}$", where f is a scalar-valued function over signal $\mathbf{x}$, $\sim$ is one of $\{<, \leq, \geq, >, =, \neq\}$, and π_{const} is a real value. STL extends the LTL [7] temporal operators by indexing them with an interval of the form (a, b), $(a, b]$, $[a, b)$, $[a, b]$, (a, ∞), or $[a, \infty)$, where a and b are nonnegative real constants, with $a \leq b$. If I is an interval, the syntax of an STL formula is:

$$\varphi := \top \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \,\mathsf{U}_I\, \varphi_2.$$

The "always" and "eventually" operators are special cases of the until operator: $\mathsf{G}_I\varphi \triangleq \neg\mathsf{F}_I\neg\varphi$ and $\mathsf{F}_I\varphi \triangleq \top\,\mathsf{U}_I\,\varphi$. If interval I is omitted, the default interval $[0, \infty)$ is implied. Formally, the semantics are given as follows:

$$(\mathbf{x},t) \models \mu \text{ iff } \mathbf{x} \text{ satisfies } \mu \text{ at time } t \tag{2.1}$$

$$(\mathbf{x},t) \models \neg\varphi \text{ iff } (\mathbf{x},t) \not\models \varphi \tag{2.2}$$

$$(\mathbf{x},t) \models \varphi_1 \wedge \varphi_2 \text{ iff } (\mathbf{x},t) \models \varphi_1 \wedge (\mathbf{x},t) \models \varphi_2 \tag{2.3}$$

$$(\mathbf{x},t) \models \varphi_1 \,\mathsf{U}_{[a,b]}\, \varphi_2 \text{ iff } \exists t' \in [t+a, t+b], (\mathbf{x},t') \models \varphi_2 \wedge \forall t'' \in [t,t'], (\mathbf{x},t'') \models \varphi_1. \tag{2.4}$$

Instead of a Boolean function, the *quantitative semantics* of STL uses a real-valued function ρ of a trace $\mathbf{x}$, a formula φ, and a time t, satisfying:

$$\rho(\varphi, \mathbf{x}, t) \geq 0 \;\text{ iff }\; (\mathbf{x}, t) \models \varphi. \tag{2.5}$$

Quantitative semantics capture the notion of *robustness of satisfaction* of φ by a trace $\mathbf{x}$. If the absolute value of $\rho(\varphi, \mathbf{x}, t)$ is large, perturbations in $\mathbf{x}$ are less likely to affect the Boolean satisfaction of φ.

Without loss of generality, an STL predicate μ can be identified with an inequality of the form $f(\mathbf{x}) \geq 0$. From this form, a straightforward quantitative semantics for predicate μ is defined as

$$\rho(\mu, \mathbf{x}, t) = f(\mathbf{x}(t)). \tag{2.6}$$

The semantics of ρ for every STL formula is defined inductively as follows [3]:

$$\rho(\neg\varphi, \mathbf{x}, t) = -\rho(\varphi, \mathbf{x}) \tag{2.7}$$

$$\rho(\varphi_1 \wedge \varphi_2, \mathbf{x}, t) = \min\{\rho(\varphi_1, \mathbf{x}, t), \rho(\varphi_2, \mathbf{x}, t)\} \tag{2.8}$$

$$\rho(\varphi_1 \mathsf{U}_I \varphi_2, \mathbf{x}, t) = \sup_{t' \in t+I} \left\{\min\left\{\rho(\varphi_2, \mathbf{x}, t'), \inf_{t'' \in [t,t')} \rho(\varphi_1, \mathbf{x}, t'')\right\}\right\}. \tag{2.9}$$

Applying (2.9), the semantics of the derived "always" and "eventually" operators is $\rho(\mathsf{G}_I\varphi, \mathbf{x}, t) = \inf_{t' \in t+I} \rho(\varphi, \mathbf{x}, t')$ and $\rho(\mathsf{F}_I\varphi, \mathbf{x}, t) = \sup_{t' \in t+I} \rho(\varphi, \mathbf{x}, t')$.

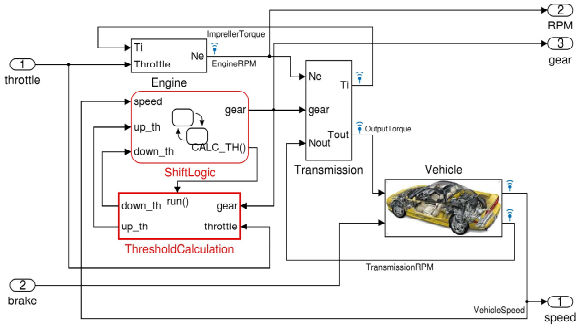

Fig. 1. The close-loop Simulink model of an automatic transmission controller.

Parametric Signal Temporal Logic (PSTL) [1] extends STL with STL *template formulas* having unknown symbolic *Scale* parameters π in the predicate constraints, $\mu = f(\mathbf{x}) \sim \pi$, or *Time* parameters τ in the interval of temporal operators. Given a valuation function assigning a value to each symbolic parameter, an STL formula is instantiated from a PSTL formula.

For example, Fig. 1 shows an automatic transmission controller model taking in throttle position and brake torque and computing gear position, vehicle speed, and `RPM`. To verify that, within the first 10 seconds, `speed` never exceeds 120 mph and `RPM` never exceeds 4500 rpm, the following STL formula can be used:

$$\varphi = \mathsf{G}_{[0,10]}(\texttt{speed} \leq 120 \wedge \texttt{RPM} \leq 4500). \tag{2.10}$$

The quantitative semantics of (2.10) from time 0, i.e., $\rho(\varphi, \mathbf{x}, 0)$, or simply $\rho(\varphi, \mathbf{x})$, is $\rho(\varphi, \mathbf{x}) = \inf_{t \in [0,10]}\{\min\{120 - \texttt{speed}(t), 4500 - \texttt{RPM}(t)\}\}$. If the exact values of the maximum speed or RPM are unknown, we can turn them into symbolic parameters π_{speed} and π_{RPM}, transforming (2.10) into a PSTL formula:

$$\varphi(\pi_{speed}, \pi_{rpm}) = \mathsf{G}_{[0,10]}(\texttt{speed} \leq \pi_{speed} \wedge \texttt{RPM} \leq \pi_{rpm}). \tag{2.11}$$

STL formula (2.10) is obtained by the valuation $v = (\pi_{speed} \mapsto 120, \pi_{rpm} \mapsto 4500)$ applied to PSTL formula (2.11). If we want to know how long the system remains in $(\texttt{speed} \leq 120)$ and $(\texttt{RPM} \leq 4500)$, we can use the PSTL formula $\varphi = \mathsf{G}_{[0,\tau_d]}(\texttt{speed} \leq 120 \wedge \texttt{RPM} \leq 4500)$, which uses the time parameter τ_d.

3 Weighted STL and parametric weighted STL

Consider now the example in (2.10), where $\texttt{speed} \leq 120$ and $\texttt{RPM} \leq 4500$ have the same contribution, even though `speed` and `RPM` are measured in different units. For example, $\texttt{speed} = 150$ has the same contribution to robustness value as $\texttt{RPM} = 4530$, since both exceed their respective thresholds by 30 units. However, a counterexample with a 30mph excessive `speed` is much more meaningful than one 30rpm above the threshold. To better describe the desirable behavior of the

system, we then propose new weighted temporal logics: weighted STL (WSTL) and parametric weighted STL (PWSTL).

Similar to STL, a WSTL formula is written using the following grammar:

$$\varphi := \top \mid \mu.\omega \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \,\mathsf{U}_I\, \varphi_2 \ ,$$

which associates a weight parameter ω with each predicate μ, to express the interest or the importance of the predicate. Thus, the quantitative semantics of WSTL is defined using a real-valued function ρ_w satisfying the property

$$\rho_w(\varphi.\boldsymbol{\omega}, \mathbf{x}, t) \geq 0 \text{ iff } (\mathbf{x}, t) \models \varphi.\boldsymbol{\omega}, \tag{3.1}$$

where $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_m)$, $\omega_i \in \mathbb{R}^{>0}$, and m is the number of predicates in φ. The quantitative semantics for each predicate μ with a weight parameter ω in WSTL is defined as:

$$\rho_w(\mu.\omega, \mathbf{x}, t) = \omega \cdot f(\mathbf{x}(t)). \tag{3.2}$$

Note that the semantics defined for WSTL still captures the robustness satisfaction and does not change the temporal semantics of any operator. Thus, ρ_w can be inductively defined using the same rules (2.7-2.9). Analogous to PSTL, we equip WSTL with scale and time parameters to obtain PWSTL and the scalar parameter π in predicates is in the form of $\mu.\omega = \omega \cdot (f(\mathbf{x}) \sim \pi)$. STL and PSTL are the special case for WSTL and PWSTL when all the elements of $\boldsymbol{\omega}$ are one.

In the previous example, we could improve (2.10) by using the WSTL formula

$$\varphi.(\omega_1, \omega_2) = \mathsf{G}_{[0,10]}((\texttt{speed} \leq 120).\omega_1 \wedge (\texttt{RPM} \leq 4500).\omega_2). \tag{3.3}$$

Further, we can change PSTL formula (2.11) into PWSTL formula

$$\varphi(\pi_{speed}, \pi_{rpm}).(\omega_1, \omega_2) = \mathsf{G}_{[0,10]}((\texttt{speed} \leq \pi_{speed}).\omega_1 \wedge (\texttt{RPM} \leq \pi_{rpm}).\omega_2), \tag{3.4}$$

and its qualitative semantics $\rho_w(\varphi(\pi_{speed}, \pi_{rpm}).(\omega_1, \omega_2), \mathbf{x}, t)$ is defined as:

$$\inf_{t \in [0,10]} \{\min\{\omega_1 \cdot (\pi_{speed} - \texttt{speed}(t)), \omega_2 \cdot (\pi_{rpm} - \texttt{RPM}(t))\}\}.$$

An especially meaningful case would be $\omega_1 = 1/\pi_{speed}$ and $\omega_2 = 1/\pi_{rpm}$, which eliminates the impact of measuring signals in different units. Moreover, if $\omega_1/\omega_2 > \pi_{rpm}/\pi_{speed}$, we assign more weight to predicate $\texttt{speed} \leq \pi_{speed}$, indicating that the scenario where speed is over the threshold is more critical.

We improve the general requirement mining framework [4] by using WSTL and PWSTL as the underlying formalism instead of STL and PSTL. The framework contains a falsification engine and a synthesis engine. The framework takes in a Simulink model and a PWSTL formula, and synthesizes a candidate WSTL formula from a random trace. The falsification engine searches the input domain and tries to find a counterexample. Then, the synthesis engine takes in a set of found counterexamples and searches the parameter domain for a tight valuation function from the corresponding PWSTL formula. The valuation function generates the new candidate WSTL for the falsification engine. This counterexample-guided refinement procedure repeats until the falsification engine fails to find any counterexample. The final candidate WSTL will be the mined specification.

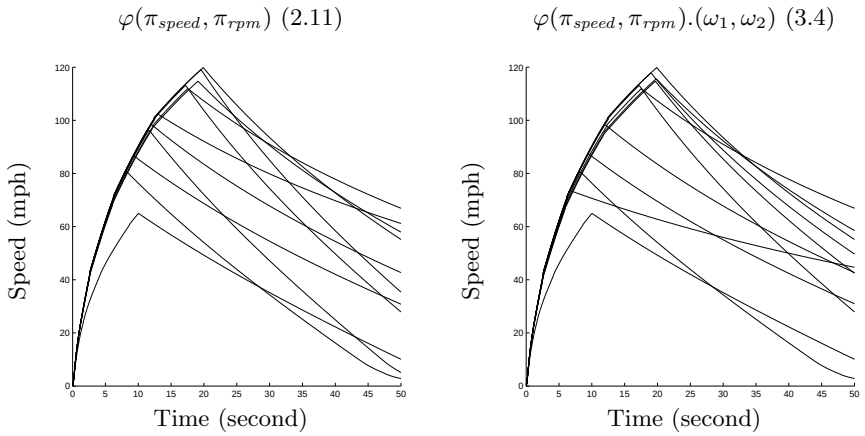

Fig. 2. The vehicle speed of the counterexamples collected during the mining procedure for PSTL (2.11) and PWSTL (3.4).

We choose the tool Breach [2] for both engines in our framework. Fig. 2 demonstrates the benefit of using WSTL and PWSTL. We use $\omega_1 = 50$ and $\omega_2 = 0.01$ to give more weight to vehicle speed. The mined requirement is $\varphi.(50, 0.01) = \mathsf{G}_{[0,10]}((\texttt{speed} \leq 120).50 \wedge (\texttt{RPM} \leq 4773).0.01)$. We observe that our choice of weights causes the falsification engine to find counterexamples where `speed` is higher than `RPM`. In this example, speed and RPM values are correlated, so the use of weights reduces the runtime for the entire mining procedure only by about 20 seconds. For weakly-related constraints, especially for the non-convex problems, the previous solution [4] by using priority might cause the mining to linger at the alternating local pole, while overlooking the possible global optimizations. Thus, weights can help accelerate mining convergence and provide more useful information about the system's behavior when debugging. In the future, we plan to provide a better evaluation of this framework.

References

1. E. Asarin, A. Donzé, O. Maler, and D. Nickovic. Parametric identification of temporal properties. In *RV*, pages 147–160, 2011.
2. A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *CAV*, pages 167–170, 2010.
3. A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *FORMATS*, pages 92–106, 2010.
4. X. Jin, A. Donzé, J. Deshmukh, and S. Seshia. Mining requirements from closed-loop control models. In *Processings of HSCC 2013: International Conference on Hybrid Systems: Computation and Control (to be appear)*. ACM, 2013.
5. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Syst.*, 2(4):255–299, 1990.
6. O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *FORMATS/FTRTFT*, pages 152–166, 2004.
7. A. Pnueli. The temporal logic of programs. In *Proc. 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57, 1977.