The Journal of Systems & Software 174 (2021) 110881

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Signal-Based Properties of Cyber-Physical Systems: Taxonomy and Logic-based Characterization☆

Chaima Boufaied[a,*], Maris Jukss[1], Domenico Bianculli[a], Lionel Claude Briand[a,b], Yago Isasi Parache[c]

[a] Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg
[b] School of EECS, University of Ottawa, Canada
[c] LuxSpace Sàrl, Luxembourg

ARTICLE INFO

Article history:
Received 1 December 2019
Received in revised form 24 August 2020
Accepted 1 December 2020
Available online 17 December 2020

Keywords:
Signals
Signal-based properties
Temporal logic
Taxonomy

ABSTRACT

The behavior of a cyber-physical system (CPS) is usually defined in terms of the input and output signals processed by sensors and actuators. Requirements specifications of CPSs are typically expressed using signal-based temporal properties. Expressing such requirements is challenging, because of (1) the many features that can be used to characterize a signal behavior; (2) the broad variation in expressiveness of the specification languages (i.e., temporal logics) used for defining signal-based temporal properties. Thus, system and software engineers need effective guidance on selecting appropriate signal behavior types and an adequate specification language, based on the type of requirements they have to define.

In this paper, we present a taxonomy of the various types of signal-based properties and provide, for each type, a comprehensive and detailed description as well as a formalization in a temporal logic. Furthermore, we review the expressiveness of state-of-the-art signal-based temporal logics in terms of the property types identified in the taxonomy. Moreover, we report on the application of our taxonomy to classify the requirements specifications of an industrial case study in the aerospace domain, in order to assess the feasibility of using the property types included in our taxonomy and the completeness of the latter.

1. Introduction

Cyber–physical systems (CPSs) are systems characterized by a complex interweaving of hardware and software (Lee and Seshia, 2016). They are widely used in many safety-critical domains (e.g., aerospace, automotive, medical) where validation and verification (V&V) activities (Bartocci et al., 2018) of the system's intended functionality play a crucial role to guarantee the reliability and safety of the system.

A typical CPS consists of a mix of analog and digital components, such as sensors, actuators, and control units, which process input and output signals. System engineers specify the desired system behavior by defining requirements in terms of the signals obtained from these components. Such requirements can be specified using *signal-based temporal properties*, which characterize the expected behavior of signals. For example, a property may require that a signal must not exhibit an abrupt increase of amplitude (i.e., a spike or bump) within a certain time interval, or that the signal shall manifest an oscillatory behavior with a particular period.

Expressing requirements in terms of signal-based temporal properties poses a number of challenges for system and software engineers. First, a signal behavior (e.g., a spike) can be characterized using a number of features (e.g., amplitude, slope, width); for example, a total of 16 different features (and eight parameters) have been identified in the literature (Adam et al., 2014) to detect (and thus characterize) a spike in a signal. Engineers may decide to choose various subsets of features; without proper guidelines for selecting the features most appropriate in a certain context and without their precise characterization, the resulting specification of a signal behavior may become ambiguous or inconsistent. The second challenge is related to the expressiveness of the specification languages used for defining signal-based temporal properties. Starting from the seminal work on *STL* (Maler and Nickovic, 2004) (Signal Temporal Logic), there have been several

☆ Editor: Raffaela Mirandola.
* Corresponding author.
E-mail addresses: chaima.boufaied@uni.lu (C. Boufaied), maris.jukss@uni.lu (M. Jukss), domenico.bianculli@uni.lu (D. Bianculli), lionel.briand@uni.lu (L.C. Briand), Isasi@luxspace.lu (Y. Isasi Parache).
[1] This work was done while the author was affiliated with the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg.

proposals of languages that extend more traditional temporal logics like *LTL* (Linear Temporal Logic) to support the specification of signal-based behaviors. Such languages have different levels of expressiveness when it comes to describing certain signal behaviors. For example, *STL* cannot be used to express properties (like those related to oscillatory behaviors) that require to reference the concrete value of a signal at an instant in which a certain property was satisfied (Brim et al., 2014). This means that engineers need guidance to carefully select the language to use for defining signal-based properties, based on the type of requirements they are going to define, the expressiveness of the candidate specification languages, and the availability of suitable tools (e.g., trace checker) for each language.

We remark that these challenges for the specification of signal-based temporal properties have implications also in terms of V&V. The lack of precise descriptions of signal behaviors (and their features) and the use of specification languages with limited expressiveness, may lead engineers to resort to manual checking (e.g., visual inspection of signal waveforms) of properties on signals. Although an anomalous spike in amplitude can be easily spotted by visual inspection of the waveform of a signal that is mostly stable, manually detecting complex signal behaviors on waveforms with intricate shapes is a cumbersome and error-prone process.

In this paper, we tackle these two challenges by proposing a taxonomy of the most common types of signal-based temporal properties and a logic-based characterization of such properties. Based on industrial experience and a thorough review of the literature, our goal is to provide system and software engineers, as well as researchers working on CPSs, with a reference guide to systematically identify and characterize signal behaviors, to support both requirements specification and V&V activities. More specifically, we address the first challenge by providing, through the taxonomy, a comprehensive and detailed description of the different types of signal-based behaviors, with each property type precisely characterized in terms of a temporal logic. As a result, an engineer can be guided by the precise characterization of the property types included in our taxonomy, to derive — from an informal requirements specification — a formal specification of a property, which can then be used in the context of V&V activities (e.g., as test oracle). We take on the second challenge by reviewing the expressiveness of the main temporal logics that have been proposed in the literature for specifying signal-based temporal properties (i.e., *STL*, *STL** (Brim et al., 2014), *SFO* (Bakhirkin et al., 2018) - Signal First-Order Logic), in terms of the property types identified in the taxonomy. In this way, we can guide engineers to choose a specification formalism based on their needs in terms of property types to express.

We developed our taxonomy of signal-based properties based on practical experience in analyzing temporal requirements in CPS domains like the aerospace industry, and by reviewing the literature in the area of verification of cyber-physical systems, starting from the recent survey of specification formalisms in Bartocci et al. (2018). We identified and included in our taxonomy the following property types:

- *Data assertion*, which specifies constraints on the signal value;
- *Signal behavior*, representing a signal behavior in terms of a particular waveform, such as spikes and oscillations;
- *Relationship between signals*, a type that includes *functional relationship* properties, based on the application of a transformation (e.g., differentiation) on signals, and *order relationship* properties, stating constraints on the order of events/states related to signal behaviors. The *order relationship* type also includes properties describing the *transient behavior* of a signal when changing from the current value to a new target value (i.e., rising/falling, overshooting/undershooting behaviors).

For each of these types, we provide a logic-based characterization using *SFO* and also discuss alternative formalizations — when applicable — using also *STL* and *STL**. In this way, we are able to report on the expressiveness of state-of-the-art temporal logics with respect to the property types included in our taxonomy: *SFO* is the only language among the three we considered in which we can express *all the property types of our taxonomy*.

We also report on the application of our taxonomy to classify the requirements specifications of an industrial case study in the aerospace domain. Through this case study we show:

- The feasibility of expressing requirements specifications of a real-world CPS using the property types included in our taxonomy. Indeed, in the vast majority of the cases, the mapping from a specification written in English to its corresponding property type defined in the taxonomy was straightforward.
- The completeness of our taxonomy: all requirements specifications of the case study could be defined using the property types included in our taxonomy.

To summarize, the main contributions of this paper are:

- a taxonomy of signal-based properties;
- a logic-based characterization of the various property types included in the taxonomy;
- a discussion on the expressiveness of state-of-the-art temporal logics with respect to the property types included in our taxonomy;
- the application of our taxonomy to classify the requirements specifications of an industrial case study in the aerospace domain.

The rest of the paper is structured as follows. Section 2 provides background concepts on signals and temporal logics for signal-based properties. Section 3 illustrates our taxonomy of signal-based properties and provides a logic-based characterization of each property type. In Section 4 we discuss the expressiveness of state-of-the-art temporal logics with respect to the property types included in our taxonomy. Section 5 presents the application of our taxonomy to an industrial case study. Section 6 discusses how the paper contributions can support the research community and practitioners. Section 7 discusses related work. Section 8 concludes the paper, providing directions for future work.

2. Background

2.1. Signals

A finite length signal s over a domain $\mathbb{D}$ is a function $s\colon \mathbb{T} \to \mathbb{D}$, where $\mathbb{T}$ is the time domain and $\mathbb{D}$ is an application-dependent value domain. In the context of CPSs, we need to differentiate between *analog*, *discrete*, and *digital* signals (Jakšić et al., 2016).

An analog signal is a signal that is continuous both in the time and in the value domains. The time domain $\mathbb{T}$ of an analog signal is thus the set of non-negative real numbers $\mathbb{R}_{\geq 0}$ and the value domain $\mathbb{D}$ is the set of real numbers $\mathbb{R}$. More formally, we define an analog signal s_a as $s_a\colon \mathbb{T} \to \mathbb{R}$. The domain of definition of s_a is the interval $I_{s_a} = [0, r)$, with $r \in \mathbb{Q}_{\geq 0}$; the length of s_a is defined as $|s_a| = r$; undefined signal values are denoted by $s_a(t) = \bot, \forall t \geq |s_a|$.

In a discrete signal, the value domain is continuous whereas the time domain is the set of natural numbers $\mathbb{N}$. More specifically, a discrete signal can be obtained from an analog signal through *sampling*, which is the process of converting the continuous-time domain of a signal to a discrete-time domain. Throughout this process, the analog signal is read at a regular time interval Δ called the *sampling interval*. The resulting discretized signal s_{dsc} can be represented by the values of an analog signal s_a read at the following time points: $0, \Delta, 2 \times \Delta, \ldots, k \times \Delta$. A digital signal has the set of natural numbers $\mathbb{N}$ as time domain and a finite discrete set as value domain. Such a signal can be obtained from a discrete signal by *quantization*, which is the process of transforming continuous values into their finite discrete approximations.

In the rest of the paper we will consider analog signals, simply denoted by s, unless a specific signal type is explicitly mentioned. This choice is motivated by the context in which this work has been developed, which is the domain of CPS (Matinnejad et al., 2018). In such a domain, model-driven engineering is used throughout the development process and *simulation* is used for design-time testing of system models; simulation models (e.g., those defined in *Simulink*®) capture both continuous and discrete system behaviors and, when executed, produce traces containing analog signals (Gonzalez Perez et al., 2018).

2.2. Temporal logics for signal-based properties

In this section, we provide a brief introduction to the main temporal logics that have been proposed in the literature for specifying signal-based temporal properties. They will be used in the next section to present the formalization of signal-based properties.

2.2.1. Signal Temporal Logic (STL)

STL (Maler and Nickovic, 2004) has been one of the first proposals of a temporal logic for the specification of temporal properties over dense-time (i.e., $\mathbb{T} = \mathbb{R}_{\geq 0}$), real-valued signals.

Let Π be a finite set of atomic propositions, X be a finite set of real variables, and $\mathcal{I}$ be an interval[2] $[a, b]$ over $\mathbb{R}$ with $a, b \in \mathbb{Q}_{\geq 0}$ such that $0 \leq a < b$. The syntax of *STL* with both *future* and *past* operators (Maler and Ničković, 2013) is defined by the following grammar:

$$\varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \mathsf{U}_{\mathcal{I}} \varphi_2 \mid \varphi_1 \mathsf{S}_{\mathcal{I}} \varphi_2$$

where $p \in \Pi$, $x \in X$, $\sim \in \{<, \leq, =, \geq, >\}$, $c \in \mathbb{R}$, $\mathsf{U}_{\mathcal{I}}$ is the metric "*Until*" operator, and $\mathsf{S}_{\mathcal{I}}$ is the metric "*Since*" operator. Additional temporal operators can be derived using the usual conventions; for example, "*Eventually*" $\mathsf{F}_{\mathcal{I}}\varphi \equiv \top \mathsf{U}_{\mathcal{I}}\varphi$; "*Globally*" $\mathsf{G}_{\mathcal{I}}\varphi \equiv \neg\mathsf{F}_{\mathcal{I}}\neg\varphi$; "*Once (Eventually in the Past)*" $\mathsf{P}_{\mathcal{I}}\varphi \equiv \top \mathsf{S}_{\mathcal{I}}\varphi$; "*Historically*" $\mathsf{H}_{\mathcal{I}}\varphi \equiv \neg\mathsf{P}_{\mathcal{I}}\neg\varphi$.

The semantics of *STL* is defined through a satisfaction relation $(s, t) \models_{STL} \varphi$, which indicates that signal s satisfies formula φ starting from position t in the signal. The satisfaction relation is defined inductively as follows:

$$\begin{aligned}
(s,t) &\models_{STL} p \text{ iff } p \text{ holds on } s \text{ in } t, \text{ for } p \in \Pi \\
(s,t) &\models_{STL} x \sim c \text{ iff } x \sim c \text{ holds on } s \text{ in } t, \text{ for } x \in X \\
&\qquad \text{and } c \in \mathbb{R} \\
(s,t) &\models_{STL} \neg\varphi \text{ iff } (s,t) \not\models_{STL} \varphi \\
(s,t) &\models_{STL} \varphi_1 \vee \varphi_2 \text{ iff } (s,t) \models_{STL} \varphi_1 \text{ or } (s,t) \models_{STL} \varphi_2 \\
(s,t) &\models_{STL} \varphi_1 \mathsf{U}_{[a,b]} \varphi_2 \text{ iff } \exists t'.(t' \in [t+a, t+b] \text{ and } (s,t') \models_{STL} \varphi_2 \\
&\qquad \text{and } \forall t''.(t'' \in [t, t'] \text{ and } (s,t'') \models_{STL} \varphi_1)) \\
(s,t) &\models_{STL} \varphi_1 \mathsf{S}_{[a,b]} \varphi_2 \text{ iff } \exists t'.(t' \in [t-a, t-b] \text{ and } (s,t') \models_{STL} \varphi_2 \\
&\qquad \text{and } \forall t''.(t'' \in [t, t'] \text{ and } (s,t'') \models_{STL} \varphi_1))
\end{aligned}$$

We say that a signal s satisfies an *STL* formula φ iff $(s, 0) \models_{STL} \varphi$.

Several extensions of *STL* have been proposed in the literature. For example, STL/PSL (Nickovic and Maler, 2007) adds an analog layer to STL that enables the application of (low-level) signal operations; xSTL (Ničković et al., 2018) adds support for Timed Regular Expressions (Asarin et al., 2002). The *STL* expressions that we will present in the rest of the paper can be written in the same form also in STL/PSL or xSTL since they only rely on the core operators of *STL*.

2.2.2. STL*

*STL** (Brim et al., 2014) is an extension of *STL* that adds a signal-value *freezing* operator that binds the value of a signal to a precise instant of time.

Let $\mathcal{J}$ be a finite index set (e.g., the set $\{1, \ldots, n\}$, $n \in \mathbb{N}$) and let the function $t^*\colon \mathcal{J} \to [0, |s|]$ be the *frozen time vector*; the i-th frozen time can then be referred to with $t^*_i = t^*(i)$. As in the case of *STL*, let Π be a finite set of atomic propositions, X be a finite set of real variables, and $\mathcal{I}$ be an interval $[a, b]$ over $\mathbb{R}$ with $a, b \in \mathbb{Q}_{\geq 0}$ such that $0 \leq a < b$. The syntax of *STL** is defined by the following grammar:

$$\varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \mathsf{U}_{\mathcal{I}} \varphi_2 \mid *_i[\varphi]$$

where $p \in \Pi$, $x \in X$, $\sim \in \{<, \leq, =, \geq, >\}$, $c \in \mathbb{R}$, $\mathsf{U}_{\mathcal{I}}$ is the metric "*Until*" operator, and $*_i$ is the unary signal-value freezing operator for all $i \in \mathcal{J}$. Additional operators like *Eventually* and *Globally* can be defined as done above for *STL*.

The semantics of *STL** is defined through a satisfaction relation $(s, t, t^*) \models_{STL^*} \varphi$, which indicates that signal s satisfies formula φ starting from position t in the signal, taking into account the frozen time vector $t^* \in [0, |s|]^{\mathcal{J}}$. The satisfaction relation is defined inductively as follows:

$$\begin{aligned}
(s,t,t^*) &\models_{STL^*} p \text{ iff } p \text{ holds on } s \text{ in } t, \text{ for } p \in \Pi, \\
&\qquad \text{with the frozen time vector } t^* \\
(s,t,t^*) &\models_{STL^*} x \sim c \text{ iff } x \sim c \text{ holds on } s \text{ in } t, \text{ for } x \in X \text{ and} \\
&\qquad c \in \mathbb{R}, \text{ with the frozen time vector } t^* \\
(s,t,t^*) &\models_{STL^*} \neg\varphi \text{ iff } (s,t,t^*) \not\models_{STL^*} \varphi \\
(s,t,t^*) &\models_{STL^*} \varphi_1 \vee \varphi_2 \text{ iff } (s,t,t^*) \models_{STL^*} \varphi_1 \text{ or } (s,t,t^*) \models_{STL^*} \varphi_2 \\
(s,t,t^*) &\models_{STL^*} \varphi_1 \mathsf{U}_{\mathcal{I}} \varphi_2 \text{ iff } \exists t'.(t' \in [t+a, t+b] \text{ and } (s,t,t^*) \models_{STL^*} \varphi_2 \\
&\qquad \text{and } \forall t''.(t'' \in [t, t'] \text{ and } (s,t'',t^*) \models_{STL^*} \varphi_1)) \\
(s,t,t^*) &\models_{STL^*} *_i[\varphi] \text{ iff } (s,t,t^*[i \leftarrow t]) \models_{STL^*} \varphi
\end{aligned}$$

where $[i \leftarrow t]$ is the operator substituting t with the i-th position in the frozen time vector, defined as $t^*[i \leftarrow t] = \begin{cases} t, & i = j \\ t^*(j), & i \neq j \end{cases}$.

We say that a signal s satisfies the *STL** formula φ iff $(s, 0, \mathbf{0}) \models_{STL^*} \varphi$.

2.2.3. Signal first-order logic (SFO)

SFO (Bakhirkin et al., 2018) is a formalism that combines first order logic with linear real arithmetic and uninterpreted unary function symbols; the latter represent real-valued signals evolving over time.

Let F be a set of function symbols and let $X = T \cup R$ be a set of variables, where T is the set of *time* variables and R is the set of *value* variables. Let $\Sigma = \langle f_1, f_2, \ldots, \mathbb{Z}, -, +, < \rangle$ be a (first-order) signature where $f_1, f_2, \ldots \in F$ are uninterpreted unary function symbols, $\mathbb{Z}$ are integer constants, and $-, +, <$ are the standard arithmetic functions and order relation. The syntax of *SFO* over Σ is defined by the following grammar:

$$\begin{aligned}
\varphi &::= \theta_1 < \theta_2 \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists r\colon \varphi \mid \exists t \in \mathcal{I}\colon \varphi \\
\theta &::= \tau \mid \rho \\
\tau &::= t \mid n \mid \tau_1 - \tau_2 \mid \tau_1 + \tau_2 \\
\rho &::= r \mid f(\tau) \mid n \mid \rho_1 - \rho_2 \mid \rho_1 + \rho_2
\end{aligned}$$

[2] The restriction on the non-punctual interval $\mathcal{I}$ for *STL* has been lifted in Maler and Ničković (2013).

where $r \in R$, $t \in T$, $n \in \mathbb{Z}$, $f \in F$, $\mathcal{I}$ is a time interval with bounds in $\mathbb{Z} \cup \{\pm\infty\}$. Notice that a term θ can be either a time term τ or a value term ρ. Additional logical connectors can be derived using the usual conventions; for example, $\forall r\colon \varphi \equiv \neg\exists r\colon \neg\varphi$.

Let a trace ω be an interpretation of a function symbol $f \in F$ as a signal, denoted by $[\![f]\!]_\omega$; let a valuation v be an interpretation of a variable $x \in X$ as a real number, denoted by $[\![x]\!]_v$. The valuation function for a term θ over the trace ω and the valuation v, denoted as $[\![\theta]\!]_{\omega,v}$ is defined inductively as follows: $[\![x]\!]_{\omega,v} = [\![x]\!]_v$, $[\![n]\!]_{\omega,v} = n$ for all $n \in \mathbb{Z}$, $[\![f(\tau)]\!]_{\omega,v} = \left[\!\left[f\left([\![\tau]\!]_{\omega,v}\right)\right]\!\right]_\omega$, $[\![\theta_1 - \theta_2]\!]_{\omega,v} = [\![\theta_1]\!]_{\omega,v} - [\![\theta_2]\!]_{\omega,v}$, $[\![\theta_1 + \theta_2]\!]_{\omega,v} = [\![\theta_1]\!]_{\omega,v} + [\![\theta_2]\!]_{\omega,v}$. The semantics of *SFO* is defined through a satisfaction relation $(\omega, v) \models_{SFO} \varphi$, which indicates the satisfaction of formula φ over the trace ω and the valuation v. The satisfaction relation is defined inductively as follows:

$$(\omega, v) \models_{SFO} \theta_1 < \theta_2 \text{ iff } [\![\theta_1]\!]_{\omega,v} < [\![\theta_2]\!]_{\omega,v}$$
$$(\omega, v) \models_{SFO} \neg\varphi \text{ iff } (\omega, v) \not\models_{SFO} \neg\varphi$$
$$(\omega, v) \models_{SFO} \varphi_1 \vee \varphi_2 \text{ iff } (\omega, v) \models_{SFO} \varphi_1 \vee (\omega, v) \models_{SFO} \varphi_2$$
$$(\omega, v) \models_{SFO} \exists r\colon \varphi \text{ iff } (\omega, v[r \leftarrow a]) \models_{SFO} \varphi \text{ for some } a \in \mathbb{R}$$
$$(\omega, v) \models_{SFO} \exists t \in \mathcal{I}\colon \varphi \text{ iff } (\omega, v[t \leftarrow a]) \models_{SFO} \varphi \text{ for some } a \in \mathbb{R}$$

Variants of *SFO* can be defined by opportunely changing the underlying signature Σ.

3. Taxonomy of signal-based properties

One of the main challenges in using signal-based temporal properties for expressing requirements of CPSs is the lack of precise descriptions of signal behaviors. First, a signal behavior (e.g., a spike or an oscillation) can be "described" in different ways, i.e., it can be characterized using various features; for example, a total of 16 different features (and eight parameters) have been identified in the literature (Adam et al., 2014) to detect a spike in a signal. Given the large variety of options, (software and system) engineers may choose various subsets of features for characterizing the same type of signal behavior, leading to ambiguity and inconsistency in the specifications. In addition, slightly different features may have similar names (e.g., "peak amplitude" and "peak-to-peak amplitude"), potentially leading to mistakes when writing specifications. It is then important to define proper guidelines for selecting the features most appropriate in a certain context, and provide engineers with a precise characterization of such features.

In this section, we tackle this challenge by proposing a taxonomy of the most common types of signal-based temporal properties and a logic-based characterization of such properties. Our goal is to provide system and software engineers, as well as researchers working on CPSs, with a reference guide to systematically identify and characterize signal behaviors, so that they can be defined precisely and used correctly during the development process of CPSs, in particular during the activities related to requirements specification and V&V.

Our taxonomy provides a comprehensive and detailed description of the different types of signal-based behaviors, with each property type precisely characterized in terms of a temporal logic. As a result, an engineer can be guided by the precise characterization of the property types included in our taxonomy, to derive—from an informal requirements specification—a formal specification of a property, which can be used in other development activities (e.g., V&V).

We developed this taxonomy based on our general understanding of temporal requirements in CPS domains like the aerospace industry, and by reviewing the literature in the area of verification of cyber-physical systems, starting from the recent survey in Bartocci et al. (2018). The taxonomy focuses on properties specified in the time domain; we purportedly leave out properties specified in the frequency domain (Nguyen et al., 2017; Donzé et al., 2012) because in our context (V&V of CPS) the properties of interest are mainly specified in the time domain.

The taxonomy (with the acronyms) of signal-based property types is shown in Fig. 1. At the top level, it includes three main signal-based property types:

Data assertion (DA): properties expressing constraints on the value of a signal.

Signal behavior (SB): properties on the behavior represented by a signal shape. We further distinguish among two property subtypes:

- properties on signals exhibiting spikes (SPK);
- properties on signals manifesting oscillatory behaviors (OSC).

Relationship between signals (RSH): properties characterizing relationships between signals. This type includes two further property subtypes:

- *functional*, based on the application of a signal transforming function (RSH-F);
- *order*, describing sequences of events/states related to signal behaviors (RSH-F). In this category we also include properties of transient behaviors of a signal when changing from the current value to a new target value, such as:
  - properties on signals exhibiting a *rising* (Rise Time - RT) or a *falling* (Fall Time - FT) behavior;
  - properties on signals exhibiting an *overshoot* (OSH) or an *undershoot* (USH) behavior.

In the following subsections we provide the detailed description of each property type, including a mathematical formalization and examples. We use (a variant of) *SFO* to formalize the various property types; anticipating the results of Section 4, the reason for the adoption of *SFO* is its expressiveness, which allows us to express *all the property types considered in this paper*. We also provide examples of properties in *STL* and *STL** (when applicable). The variant of *SFO* we use for the formalization has the following signature $\Sigma = \langle F, A, Rel, \mathbb{Z}, \mathbb{R}\rangle$, where:

- $F = Sig \cup Aux$ is the set of function symbols, composed of signal functions $Sig = \{s, s_1, s_2, s_{tr}\}$ and auxiliary functions and predicates $Aux = \{\sigma^{\mathbb{B}e}_{s,P}, \sigma^{\mathbb{B}s}_{s,P}, \xi, checkOsc, local\_min, local\_max\}$;
- A is the set of (non-linear) arithmetic functions $A = \{+, -, \times, \div, \text{abs}\}$, where abs represents the absolute value operator;
- Rel is the set of relational operators $Rel = \{<, >, \geq, \leq, =, \neq\}$;
- $\mathbb{Z}$ and $\mathbb{R}$ are integer and real constants, respectively.

3.1. Data assertion

A data assertion specifies a constraint on the value of a signal. This constraint is expressed through a *signal predicate* of the form $s \bowtie expr$, where *expr* is an *SFO* value term defined over the value domain of the signal s and $\bowtie \in Rel$. A data assertion property holds on the signal if the assertion predicate evaluates to *true*. Data assertions can be combined to form more complex expressions through the standard logical connectives. We distinguish between *untimed* data assertions, which are evaluated

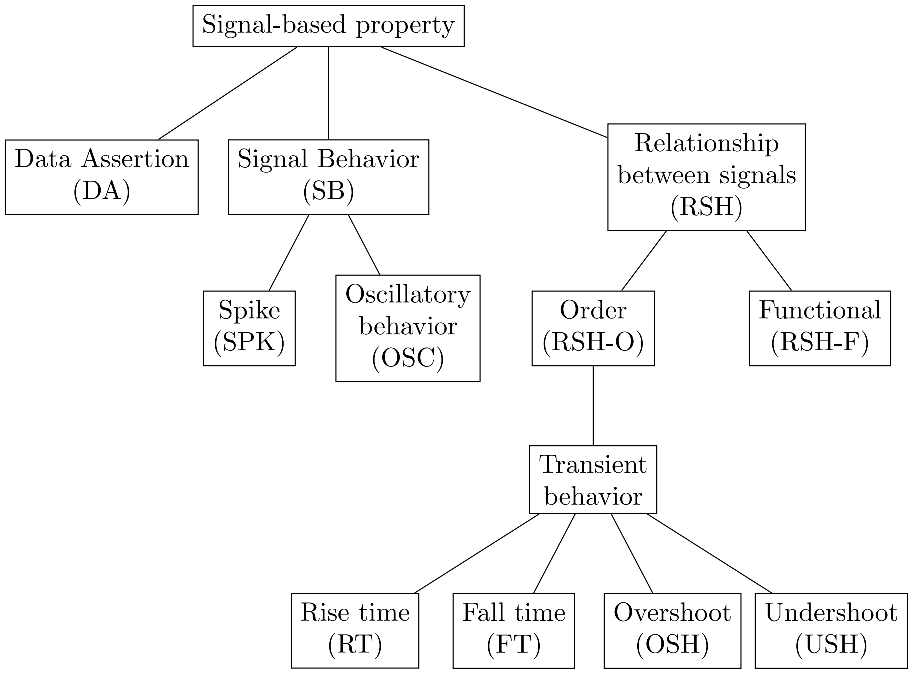


Fig. 1. Taxonomy of signal-based properties.

through the entire domain of definition I_s of a signal s, and *time-constrained* data assertions, which are evaluated over one or more distinct sub-intervals of the signal domain of definition.

More formally, let H be a set of time intervals $H = \{\mathcal{I}_1, \ldots, \mathcal{I}_K\}$, such that $\mathcal{I}_k \subseteq I_s$, $1 \leq k \leq K$, and for all $i, j \in \{1, \ldots, K\}$, $i \neq j$ implies $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$. A data assertion defined over the time intervals in H holds on a signal s if and only if (iff) the *SFO* formula $\bigwedge_{h \in H} \forall i \in h : s(i) \bowtie expr$ evaluates to *true*. Notice that an *untimed* data assertion over a signal s is defined by having $H = \{I_s\}$.

For example, let us consider the property *pDA*: "The signal value shall be less than 3 between 2 tu and 6 tu and between 10 tu and 15 tu", where "tu" is a generic time unit (which has to be set according to the application domain, e.g., seconds). This property is a *time-constrained* data assertion over the two intervals [2, 6] and [10, 15]; it can be expressed in *SFO* as:

SFO *pDA* $\forall t \in [2, 6] : s(t) < 3 \land \forall t \in [10, 15] : s(t) < 3$

Fig. 2 shows two signals, s_1 plotted with a thick line (—), and s_2 plotted with a thin line (—); the threshold on the signal value specified by the property is represented with a dashed horizontal line. Property *pDA* does not hold for s_2 as its value is above the threshold of 3 in the intervals [2, 6] and [10, 15]; however, it holds for s_1 because its value is below the threshold in both intervals.

3.1.1. *Alternative formalizations*

Data assertion properties like *pDA* can be also expressed in *STL* and *STL**:

STL *pDA* ≡ STL* *pDA* $G_{[2,6]}(s < 3) \land G_{[10,15]}(s < 3)$

3.2. *Spike*

A spike[3] can be informally defined as a short-lived, (relatively) large increase or decrease of the value of a signal. Such a signal

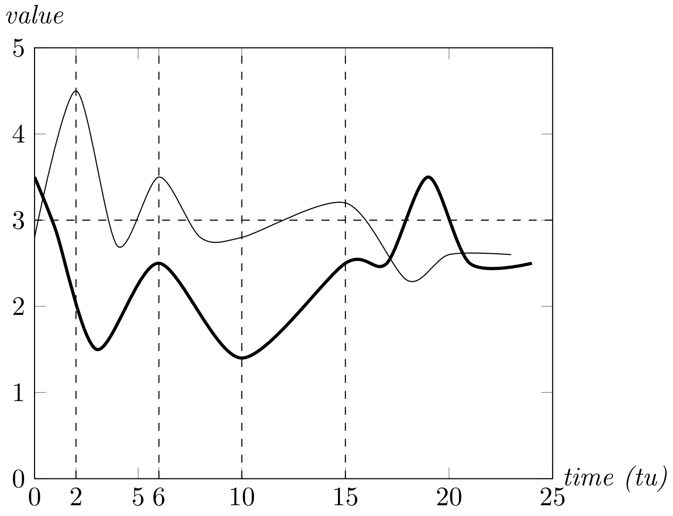


Fig. 2. Two signals used to evaluate property *pDA*: signal s_1 (—) satisfies the property whereas signal s_2 (—) violates it.

behavior is typically undesirable (Bartocci et al., 2018). However, there are situations in which a spike characterized by a set of specific features is desirable, as it is the case for the discovery pulse (Ničković, 2015) in the discovery mode of the DSI3 protocol (DSI consortium, 2011).

Inspired by the definitions in the bio-medical domain (Dumpala et al., 1982), we consider four main features to characterize a spike, based on three extrema of the function corresponding to the signal shape, which are local extrema with respect to an observation interval $[f, g] \subset I_s$. These three points (with their respective coordinates) are: the peak point $(PP, s(PP))$ representing the local maximum of the signal and characterizing the

[3] A spike is also called bump, peak, or pulse in the literature.

actual spike,[4] and the two surrounding valley points $(VP_1, s(VP_1))$ and $(VP_2, s(VP_2))$ representing the local minima (closest to the peak point) of the first and second half of the spike, respectively. These three local extrema are shown in Fig. 3a; we refer the reader to Dumpala et al. (1982) for a detailed description of how to detect these points. The four features (also shown in Fig. 3a) characterizing a spike are:

- Amplitude a of the spike, defined as $a = \psi(a_1, a_2)$, where a_1 is the amplitude of the first-half of the spike shape $a_1 = \text{abs}\,(s(PP) - s(VP_1))$, a_2 is the amplitude of the second-half of the spike shape $a_2 = \text{abs}\,(s(PP) - s(VP_2))$, and ψ is a generic amplitude function[5];
- slope sp_1 between the peak point and the valley point of the first half of the spike shape, $sp_1 = \text{abs}\left(\frac{s(PP)-s(VP_1)}{PP-VP_1}\right)$;
- slope sp_2 between the peak point and the valley point of the second half of the spike shape, $sp_2 = \text{abs}\left(\frac{s(PP)-s(VP_2)}{PP-VP_2}\right)$;
- spike width w between the two consecutive valley points, $w = VP_2 - VP_1$. Note that the width w can be also defined as $w = w_1 + w_2$, where $w_1 = PP - VP_1$ and $w_2 = VP_2 - PP$.

The four features a, sp_1, sp_2, and w can be opportunely combined to define a spike of a particular shape.[6]

A spike property specifies a constraint on the existence of a spike with certain features; it evaluates to true when the signal exhibits a spike whose features satisfy certain criteria. More specifically, when defining a spike property, an engineer has to specify — for each feature — a predicate with a *threshold criterion* whose value depends on the application context. The signal predicates of each feature are then logically conjoined for characterizing the spike.

Formally, given the threshold criteria for the four features (specified as *SFO* terms over the value domain of signal s) $\Gamma_a, \Gamma_{sp_1}, \Gamma_{sp_2}, \Gamma_w$, a spike property holds on a signal s iff the following *SFO* formula evaluates to true:

$$\begin{aligned}\exists VP_1, PP, VP_2 \in [f, g] \colon & \mathit{local\_min}(VP_1, f, PP) \land \\ & \mathit{local\_max}(PP, VP_1, g) \land \\ & \mathit{local\_min}(VP_2, PP, g) \land \\ a \bowtie \Gamma_a \land sp_1 \bowtie \Gamma_{sp_1} \land & sp_2 \bowtie \Gamma_{sp_2} \land w \bowtie \Gamma_w\end{aligned} \tag{1}$$

where $\bowtie \; \in \mathit{Rel}$, *local_min* and *local_max* $\in \mathit{Aux}$ are predicates identifying local extrema, and a, sp_1, sp_2, w are *SFO* terms defined as shown above using the three variables VP_1, VP_2, and PP.

In essence, formula (1) requires (a) the existence of the three local extrema in a proper order characterizing the spike shape (i.e., a local minimum followed by a local maximum, followed by another local minimum), and (b) the satisfaction of the constraints for all the features. More relaxed formulations can be obtained by omitting some of the spike features from the above definition.

The predicate $\mathit{local\_min}(x, y, z)$ (respectively, $\mathit{local\_max}(x, y, z)$) returns true if the time point x is a local minimum (respectively, local maximum) with respect to the interval $[y, z]$. These predicates can be defined in several ways; below we provide three possible definitions.

Definition 1 (*Local Extrema Through Punctual Derivatives*)**.** Some specification languages allow for defining expressions corresponding to punctual derivatives. For example, in *SFO* the punctual derivatives can be defined as language terms as follows:

$$s'_p(t) \equiv \frac{s(t+\epsilon) - s(t)}{\epsilon} \text{ and } s''_p(t) \equiv s'_p(s'_p(t))$$

with ϵ being an arbitrary, small constant.[7] The local extrema predicates can then be defined in *SFO* as follow:

$$\mathit{local\_min}(x,y,z) \equiv \exists x \in [y, z] \colon s'_p(x) = 0 \land s''_p(x) > 0$$
$$\mathit{local\_max}(x,y,z) \equiv \exists x \in [y, z] \colon s'_p(x) = 0 \land s''_p(x) < 0$$

Definition 2 (*Local Extrema - Analytical Formulation*)**.** Another way to characterize local extrema is to write a logical expression corresponding to their analytical definition; in *SFO* we have

$$\mathit{local\_min}(x,y,z) \equiv \exists x \in [y, z] \colon \forall t \in [y, z], x \neq t \colon s(x) \le s(t)$$
$$\mathit{local\_max}(x,y,z) \equiv \exists x \in [y, z] \colon \forall t \in [y, z], x \neq t \colon s(x) \ge s(t)$$

Definition 3 (*Local Extrema Through Pre-computed Derivatives*)**.** When the first and second order derivatives of a signal are available as *(pre-computed), separate signals*, the local extrema can be characterized using such signals. Let s'_c and s''_c be the first and second order derivatives of signal s; the local extrema predicates can defined in *SFO* as follow:

$$\mathit{local\_min}(x,y,z) \equiv \exists x \in [y, z] \colon s'_c(x) = 0 \land s''_c(x) > 0$$
$$\mathit{local\_max}(x,y,z) \equiv \exists x \in [y, z] \colon s'_c(x) = 0 \land s''_c(x) < 0$$

The choice of which definition to use for defining local extrema predicates depends on the specification language and the application context; as shown above, all three definitions can be used with *SFO*.

For example, let us characterize spikes through features width w and amplitude a, with the latter defined by using the maximum function as the amplitude function ψ; let us consider the evaluation of property *pSPK1*: "In a signal, there is a spike with a maximum width of 20 tu and a maximum amplitude of 1". For this property, the parameters of an instance of specification (1) are $\Gamma_a = 1$ and $\Gamma_w = 20$; the resulting *SFO* formula is:

SFO *pSPK1* $\exists t, t', t'' \in [f, g] \colon \mathit{local\_min}(t, f, t') \land$
$\mathit{local\_max}(t', t, g) \land \mathit{local\_min}(t'', t', g) \land$
$\max(\text{abs}\,(s(t') - s(t)), \text{abs}\,(s(t'') - s(t'))) \le 1$
$\land\, \text{abs}\,(t'' - t) \le 20$

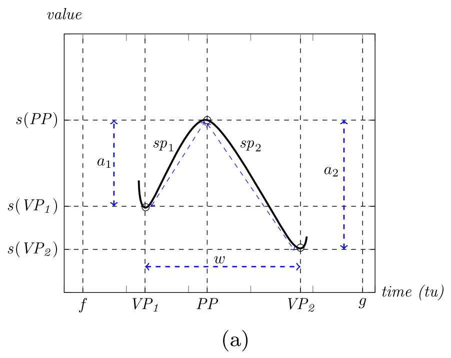
In Fig. 3b, we show two signals, s_1 plotted with a thick line (—) and s_2 plotted with a thin line (—). To evaluate property *pSPK1* on these signals, we first need to evaluate the local extrema predicates in specification (1) (according to one of the three definitions above): signal s_1 exhibits a spike where $VP_1 = 10$, $PP = 20$, and $VP_2 = 30$, while s_2 exhibits a spike where $VP_1 = 10$, $PP = 25$, and $VP_2 = 35$. In both cases, the three points satisfy the local extrema predicates. The second step is to evaluate the threshold criteria of the spike features. We calculate the amplitude a_{s_1} and the width w_{s_1} of the spike in s_1 as: $a_{s_1} = \max(\text{abs}\,(s_1(PP) - s_1(VP_1)), \text{abs}\,(s_1(PP) - s_1(VP_2))) = \max(\text{abs}\,(s_1(20) - s_1(10)), \text{abs}\,(s_1(20) - s_1(30))) = \max$

[4] In the following we only characterize and formalize spikes corresponding to an increase of the signal value; the case of a decrease of the signal value is the dual.

[5] This function depends on the application domain; for example, in the context of bio-medical systems (Dumpala et al., 1982), ψ is the minimum function.

[6] Although other spike features have been proposed in the spike detection literature—such as different types of width, amplitude, and slope (Acır and Güzeliş, 2004; Acır, 2005; Acir et al., 2005; Liu et al., 2002; Dingle et al., 1993), as well as the area under the curve (Hägglund, 1995)—we decided not to adopt them since the features we have selected are sufficient to describe (and specify) the spike behaviors we consider in this paper.

[7] In the context of a discrete signal, the ϵ constant can be replaced with the sampling interval Δ.



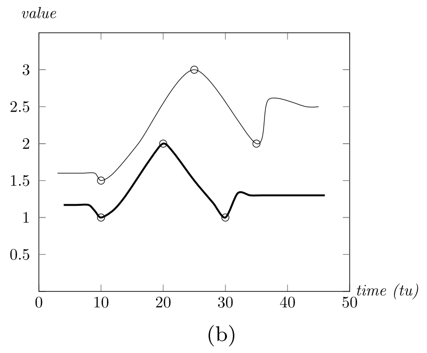


Fig. 3. (a) Main features used to define a spike based on Dumpala et al. (1982). (b) two signals used to evaluate property *pSPK1*: signal s_1 (**—**) satisfies the property, whereas s_2 (—) violates it.

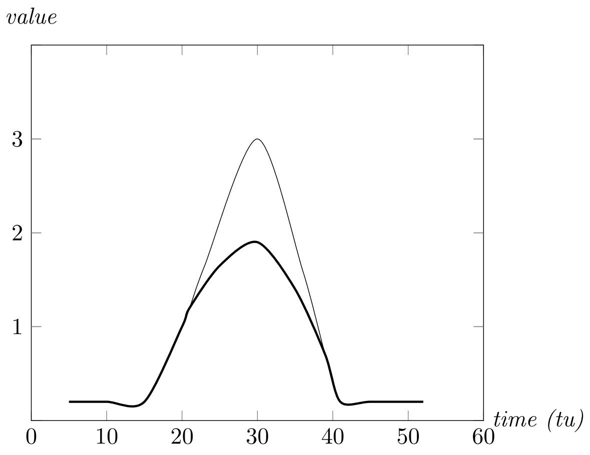


Fig. 4. Characterization of the spike in two signals s_1 (**—**) and s_2 (—) based on the definition in Kapinski et al. (2016a), with parameters $m = 0.1$, $w = 20$.

$(\text{abs}\,(2-1), \text{abs}\,(2-1)) = 1$ and $w_{s_1} = VP_2 - VP_1 = 30 - 10 = 20$. Signal s_1 satisfies property *pSPK1* because the expression $a_{s_1} \leq 1 \wedge w_{s_1} \leq 20 \equiv 1 \leq 1 \wedge 20 \leq 20$ evaluates to true. Following a similar computation, the amplitude a_{s_2} and the width w_{s_2} of the spike in s_2 are $a_{s_2} = \max(1.5, 1) = 1.5$ and $w_{s_2} = 25$; signal s_2 violates property *pSPK1* because the expression $a_{s_2} \leq 1 \wedge w_{s_2} \leq 20 \equiv 1.5 \leq 1 \wedge 25 \leq 20$ evaluates to false.

Another definition, proposed in the context of automotive control applications (Kapinski et al., 2016a), characterizes a spike using two parameters, w and $m = \frac{a}{w}$, where w is the spike width and a the spike amplitude. Formally, a signal s exhibits a spike with parameters m and w (defined as numerical constants) iff the following *SFO* formula evaluates to true:

$$\exists t \in I_s : s'(t) > m \wedge \exists t' \in [t, t+w] : s'(t') < -m \tag{2}$$

where s', denoting the first order derivative of s, can be either a pre-computed, separated signal s'_c or the punctual derivative s'_p introduced above. This characterization identifies two time instants: the first in which the signal derivative is greater than parameter m and another one in which the signal derivative is less than $-m$; the distance between these two points is the spike width w.

The main limitation of this formulation is that it does not allow to express precise constraints on the absolute value of the amplitude of a spike; instead, it uses parameter m that is a quotient between amplitude and width. We illustrate this with the example in Fig. 4, with the signals s_1 plotted with a thick line (**—**) and s_2 plotted with a thin line (—). Let us consider the evaluation of property *pSPK2*: “In a signal, there exists a spike with a maximum width of 20 tu and an amplitude greater than 2”. This property cannot be captured by an instance of specification (2), since the latter does not take into account the concept of amplitude; the property needs to be adapted. Based on the desired values of width and amplitude in property *pSPK2*, the parameters of an instance of specification (2) would be $m = 0.1$, $w = 20$. Therefore, instead of property *pSPK2*, one can consider the following alternative *pSPK3*: “In a signal, there exists a spike with a maximum width of 20 tu and parameter m equal to 0.1”, which can be captured by an instance of specification (2); the corresponding *SFO* formula is:

SFO *pSPK3* $\quad \exists t \in I_s : s'(t) > 0.1 \wedge \exists t' \in [t, t+20] : s'(t') < -0.1$

This formula will evaluate to true for both s_1 and s_2. However, signal s_1 should not satisfy the property, since its peak point does not reach a magnitude (amplitude) of 2 as was required in the original formulation of the property (*pSPK2*). This spurious spike characterization happens with specification (2) because signal s_1 follows the same shape as signal s_2 in the points in which the signal derivative s' is compared to m. We remark that the application of specification (1) to the evaluation of property *pSPK2* would correctly characterize the spike only in signal s_2. Given a lack of precision in specification (2), in the following we will consider spikes defined according to specification (1).

3.2.1. Alternative formalizations

STL. Our characterization of a spike through the *SFO* formulation (1) relies on the existence of three extrema in the function corresponding to the signal shape. In *STL*, the existence of these extrema could be formalized through proper nesting of the “eventually” and “once” operators, in conjunction with a constraint on the width of the spike. However, it would not be possible to include in such a formulation a constraint on the amplitude or on the slope, since in *STL* one cannot refer to the value of the signal at an arbitrary time point. For all these reasons, we cannot express a property like *pSPK1* in *STL*.

On the other hand, spike properties characterized through the *SFO* formulation (2) can be expressed in *STL* when the pre-computed signal derivatives are available. For example, property *pSPK3* can be expressed as

STL *pSPK3* $\quad \mathsf{F}_{[0,|s|)}(s' > 0.1 \wedge \mathsf{F}_{[0,20]} s' < -0.1)$

*STL**. Differently from *STL*, *STL** can refer to the value of the signal at a certain time point in which a local formula holds thanks to the freeze operator; below we discuss how it can be used to express properties *pSPK1* and *pSPK3*.

(Using local extrema expressed through punctual derivatives) Definition 1 for local extrema uses the values of the signal at two consecutive time points, within a small distance ϵ. However, in *STL** it is not possible to explicitly reference the signal value at time points that are not associated with the evaluation of a local (sub-)formula; hence, properties defined using punctual derivatives cannot be specified using *STL**.[8]

(Using local extrema expressed through the analytical formulation) We can characterize local extrema using the analytical formulation (Definition 2) by assuming a variant of *STL** with past operators[9] and using a 3D frozen time vector.

STL* *pSPK*1

$$\begin{aligned}&\mathsf{F}_{[f,g]} *_1 (\mathsf{G}_{[0,w_1]}(s > s^{*_1})\\ &\wedge \mathsf{F}_{[0,w_1]} *_2 (\mathsf{H}_{[0,w_1]}(s < s^{*_2})\\ &\wedge \mathsf{F}_{[0,w_2]} *_3 (\mathsf{H}_{[0,w_2]}(s > s^{*_3})\\ &\wedge \max(\mathrm{abs}(s^{*_1} - s^{*_2}), \mathrm{abs}(s^{*_2} - s^{*_3}))\\ &\leq 1 \wedge w_1 + w_2 \leq 20)))\end{aligned}$$

In the formula above, the expression in the first row states the existence of the first local minimum by checking for the existence, within the observation interval $[f, g]$, of a point (whose time instant is frozen in the first component of the frozen time vector) for which the corresponding signal value is smaller than all other signal values in the interval $[0, w_1]$; this condition is captured by the sub-formula with the "globally" operator. The expression on the second row, nesting the "historically" operator within the "eventually", states the existence of the local maximum (whose time instant is frozen in the second component of the frozen time vector), such that all the signal values between the first local minimum and such a point are indeed smaller than the local maximum. Notice that the distance between the first local minimum and the local maximum is equal to w_1.[10] The expression on the third row checks in a similar way for the existence of the second local minimum within an interval $[0, w_2]$ from the local maximum. The expression on the fourth row checks the constraints on the spike amplitude and on the spike width. For the former, it uses the values of the signal in correspondence of the first local minimum (s^{*_1}), of the local maximum (s^{*_2}), and of the second local minimum (s^{*_3}).

Note that this property relies on a particular sequence of local extrema (i.e., valley-peak-valley); other variants of this property can be specified by changing the order of the sub-formulae stating the existence of a certain extremum. Furthermore, we remark that the specification of this property assumes the knowledge of the signal shape, since it uses the two components of the width w_1 and w_2 as defined on page 8. However, making such an assumption in practice is not reasonable because typically the shape of a spike is unknown.

(Using local extrema defined through pre-computed derivatives) Property *pSPK1* can be expressed using Definition 3 for local extrema, assuming the existence of signals s' and s'' and a 3D frozen time vector.

STL* *pSPK*1

$$\begin{aligned}&\mathsf{F}_{[f,g]} *_1 \big(s' = 0 \wedge s'' > 0 \wedge \mathsf{F}_{[0,w_1]} *_2\\ &(s' = 0 \wedge s'' < 0 \wedge \mathsf{F}_{[0,w_2]} *_3 (s' = 0 \wedge s'' > 0\\ &\wedge \max(\mathrm{abs}(s^{*_1} - s^{*_2}), \mathrm{abs}(s^{*_2} - s^{*_3}))\\ &\leq 1 \wedge w_1 + w_2 \leq 20))\big)\end{aligned}$$

The structure of the formula above is similar to the one for the case of using Definition 2 for local extrema, except for the direct use of the first and second order derivatives, available as pre-computed signals. The same remarks made above in terms of assuming the knowledge of the signal shape also apply in this case.

Furthermore, pre-computed derivative signals can be used to specify property *pSPK3* in *STL** in the same way as it was done above using *STL*.

3.3. Oscillation

An oscillation can be informally described as a repeated variation over time of the value of a signal, possibly with respect to a reference value; often, in the context of CPS, oscillations represent an undesirable signal behavior.

Fig. 5 depicts an analog signal s exhibiting an oscillatory behavior with respect to a reference value *ref*, within an observation interval $oscI = [a, b] \subset I_s$. Such a behavior is characterized by the existence, within the observation interval, of M extrema of the function corresponding to the signal shape; these points are marked with blue squares (□) in the figure. A *cycle* (i.e., a *complete oscillation*) occurs when the signal value swings from one extremum to the adjacent extremum of the same type, by traversing an extremum of the other type; for example, in the figure there is one complete oscillation when the signal goes from p_3 to p_5 (two peak points) through p_4 (a valley point). The figure also shows two additional features typically used to characterize oscillations:

- the *(peak) amplitude*, denoted by *oscA*, is the distance between the maximum magnitude of the signal and its reference value;
- the *period*, denoted by *oscP*, is the time required to complete one cycle. Its reciprocal, called *frequency*, represents the number of complete oscillations occurring in a unit of time.

An oscillation property specifies a constraint on the existence, in a signal, of an oscillatory behavior with certain features; it evaluates to true when the signal exhibits an oscillatory behavior whose features satisfy certain criteria. More specifically, these criteria are expressed as relational expressions, on the oscillation amplitude and/or period, with an application-specific threshold. More formally, given the *SFO* terms representing the threshold criteria Γ_{oscP} (for the period) and Γ_{oscA} (for the amplitude), an oscillation property holds on a signal s in the observation interval $[a, b]$ iff the following *SFO* formula evaluates to true:

$$\begin{aligned}&\forall t \in [a,b]{:}(\exists t', t'' \in [t,b]{:}\\ &\quad \mathit{local\_min}(t,a,t') \rightarrow\\ &\quad (\mathit{local\_max}(t',t,b) \wedge \mathit{local\_min}(t'',t',b)\\ &\quad \wedge \mathit{checkOsc}(t,t',t'',\bowtie_P,\Gamma_{oscP},\bowtie_A,\Gamma_{oscA}))\\ &\quad \wedge \mathit{local\_max}(t,a,t') \rightarrow\\ &\quad (\mathit{local\_min}(t',t,b) \wedge \mathit{local\_max}(t'',t',b)\\ &\quad \wedge \mathit{checkOsc}(t,t',t'',\bowtie_P,\Gamma_{oscP},\bowtie_A,\Gamma_{oscA})))\end{aligned} \tag{3}$$

where *local_min*(x, y, z) (respectively, *local_max*(x, y, z)) is a predicate that returns true if the time point x is a local minimum (respectively, local maximum) with respect to the interval $[y, z]$

8 Such a restriction could be lifted when using discrete signals, since the distance between two consecutive time points is known and is equal to the sampling interval Δ.

9 Although the version of *STL** presented in Brim et al. (2014) does not use past operators, the addition of such operators would be done along the lines of the definition of *STL* with past operators in Maler and Ničković (2013).

10 If the spike shape is symmetrical, the distance between all local extrema is equal to $\frac{w}{2}$.

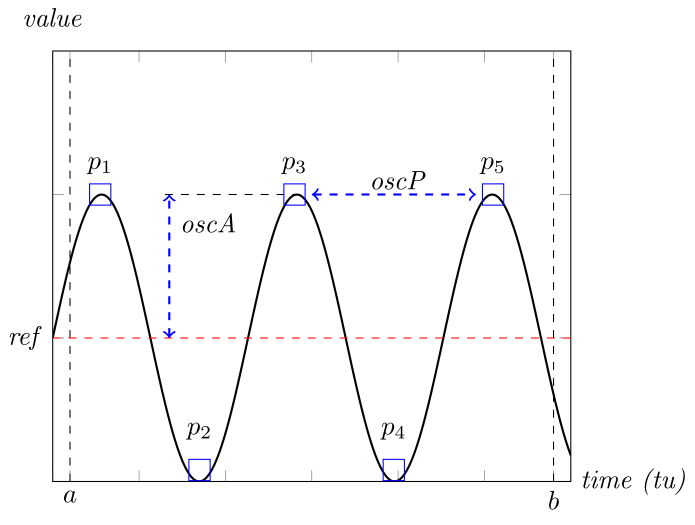


Fig. 5. A signal exhibiting an oscillatory behavior with respect to the reference value *ref*.

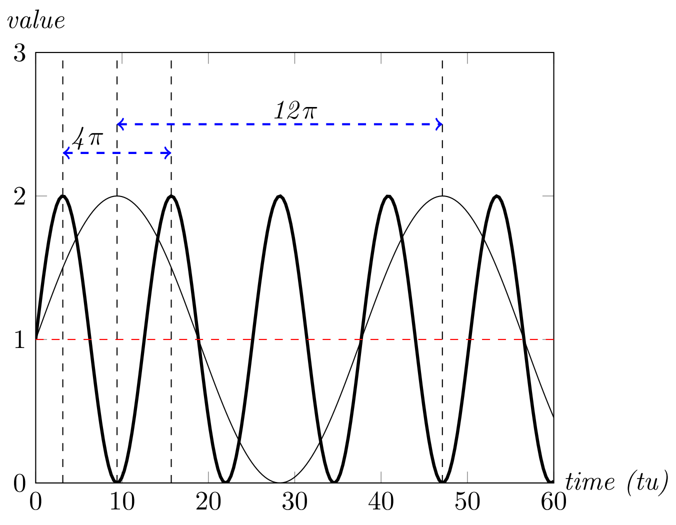


Fig. 6. Two signals used to evaluate property *pOSC*: signal s_1 (**—**) satisfies the property, whereas s_2 (—) violates it.

(see Section 3.2); $checkOsc(t, t', t'', \bowtie_P, \Gamma_{oscP}, \bowtie_A, \Gamma_{oscA})$ is a predicate that returns whether the expression $oscA \bowtie_A \Gamma_{oscA} \wedge oscP \bowtie_P \Gamma_{oscP}$ evaluates to true for the oscillation (with amplitude *oscA* and period *oscP*) determined by its first three arguments t, t', t''; $\bowtie_P$ and $\bowtie_A$ are relational operators in *Rel* of Σ.

In essence, formula (3) requires a) the existence of the three local extrema in a proper order characterizing the complete oscillation (i.e., either a local minimum followed by a local maximum followed by another local minimum, or a local maximum followed by a local minimum followed by another local maximum), and b) the satisfaction of the constraints on the oscillation features evaluated in the *checkOsc* predicate.

As an example, let us consider property *pOSC*: "Within an observation interval of 60 time units (starting from the beginning of the signal), in the signal there exist oscillations with a period less than 20 and an amplitude less than 3". For this property the parameters of an instance of specification (3) are $a = 0, b = 60, \Gamma_{oscP} = 20, \Gamma_{oscA} = 3, \bowtie_A = \bowtie_P = <$. For evaluating the property, we show two signals in Fig. 6: s_1 (drawn with a thick line) corresponds to a sine wave defined as $y = \sin(\frac{x}{2}) + 1$; s_2 (drawn with a thin line) is defined by $y = \sin(\frac{x}{6}) + 1$. In both signals, oscillations have a peak amplitude equal to 1, which satisfies the constraint on the amplitude. The period of signal s_1, calculated from its sine definition, is equal to 4π; similarly, the period of s_2 is equal to 12π (see Fig. 6). Signal s_1 satisfies property *pOSC* because it oscillates by exhibiting alternating local minima and maxima, with a period and an amplitude satisfying the thresholds ($4\pi < \Gamma_{oscP}$ and $1 < \Gamma_{oscA}$). However, signal s_2 violates the property because its period is greater than the threshold value of 20 ($12\pi > \Gamma_{oscP}$).

The pure sine wave shown in Fig. 5 is characterized by a constant period and by a constant amplitude. However, in the context of CPSs, signals may be noisy; this means that the amplitude and the period of their oscillatory behaviors may vary over time. Furthermore, a reference value may be unknown, making the computation of the oscillation amplitude challenging. In such cases one may use an aggregation function (e.g., average, maximum, minimum) over different amplitude values (e.g., peak-to-peak). In the following, we introduce the concepts of *average amplitude* and *average period*; these definitions can easily be adapted to take into account other aggregation functions.

To deal with situations in which the reference value is not known, we will consider the peak-to-peak amplitude, i.e., the difference between two adjacent extrema, denoted by $oscA_{PP}$. The *average peak-to-peak amplitude* $\overline{oscA_{PP}}$ can then be computed as the arithmetic mean of the peak-to-peak amplitude between adjacent extrema. More formally, given the sequence $p_1, \ldots, p_{M-1}, p_M$ of local extrema, $\overline{oscA_{PP}} = \frac{\sum_{i=1}^{M-1} \text{abs}\,(s(p_i) - s(p_{i+1}))}{M-1}$. Other definitions of amplitude (such as the root mean square) can be used too, depending on the application domain.

The *average period* can be defined as the arithmetic mean of the period of each complete oscillation of the signal, computed over pairs of extrema of the same type. More formally, given the sequence $p_1, \ldots, p_{M-1}, p_M$ of local extrema, we define the number *oscN* of complete oscillations within the observation interval of the signal as $oscN = \left\lfloor \frac{M-1}{2} \right\rfloor$; the *average period* $\overline{oscP}$ is then defined as $\overline{oscP} = \frac{\sum_{i=1}^{oscN} \text{abs}\,(p_{2i-1} - p_{2i+1})}{oscN}$.

When the concepts of average amplitude and average period are used to characterize an oscillatory behavior, specification (3) has to be adapted accordingly; more precisely, predicate *checkOsc* has to be redefined to consider the average amplitude $\overline{oscA_{PP}}$ and the average period $\overline{oscP}$.

Damped/driven oscillations. In the real world, oscillatory behaviors may be subject to various forces that reduce or increase their amplitude. More precisely, we distinguish between *damped* and *driven* oscillations: for the former the amplitude decays monotonically, whereas for the latter the amplitude increases monotonically.

The characterization of these specific behaviors can be done by constraining the change of the amplitude of the oscillatory signal. For example, given the sequence $p_1, \dots, p_{M-1}, p_M$ of local extrema, we say that an oscillatory signal s (formalized according to specification (3)) exhibits damped oscillations iff the following *SFO* formula evaluates to *true*:

$$\forall j \in [1, M-2]\colon \mathrm{abs}\,(s(p_j) - s(p_{j+1})) \geq \mathrm{abs}\,(s(p_{j+1}) - s(p_{j+2})) \quad (4)$$

The case for driven oscillations is similar and can be obtained from the expression above by replacing the relational operator with its dual.

The amplitude of signals may not change monotonically; in such cases, statistical trends (e.g., a linear trend) in amplitude changes may be observed. We could account for statistical trends by specifying that, on average, the difference in amplitude tends to decrease/increase; such a constraint would then be included in the formula above.

3.3.1. Alternative formalizations

STL. Similar to the case of spike properties (see Section 3.2), our formalization in *SFO* of oscillation properties relies on the existence of local extrema in the signal. Converting such formalization to *STL* would rely on the use of properly nested "eventually" and "once" operators, in conjunction with a constraint on the oscillation period. However, a constraint on the amplitude could not be expressed because in *STL* one cannot refer to the value of the signal at an arbitrary time point.

STL.* The specification of oscillatory behaviors is one of the main motivations behind the definition of *STL**. Below, we discuss how to specify property *pOSC1* in *STL** using the three local extrema characterization approaches introduced in Section 3.2.

(Using local extrema expressed through punctual derivatives) As discussed for the case of spike properties (see page 12), properties referring to local extrema expressed according to Definition 1 cannot be specified using *STL** because they would require to explicitly reference the signal value at time points that are not associated with the evaluation of a local (sub-)formula.

(Using local extrema expressed through the analytical formulation) We can express local extrema using their analytical formulation (Definition 2) by assuming a variant of *STL** with past operators. Property *pOSC* can be specified in the following way using a 3D frozen time vector:

STL* *pOSC*

$$\begin{aligned}
&\mathsf{G}_{[a,b]}(\mathsf{F}_{[0,b]*_1}(\mathsf{G}_{[0,\frac{\Gamma_{oscP}}{2}]}(s > s^{*1}) \rightarrow\\
&\quad \mathsf{F}_{[0,\frac{\Gamma_{oscP}}{2}]*_2}(\mathsf{H}_{[0,\frac{\Gamma_{oscP}}{2}]}(s < s^{*2})\\
&\quad \wedge \mathsf{F}_{[0,\frac{\Gamma_{oscP}}{2}]*_3}(\mathsf{H}_{[0,\frac{\Gamma_{oscP}}{2}]}(s > s^{*3})\\
&\qquad \wedge \mathrm{abs}\,(s^{*1} - s^{*2}) < 3)))\\
&\quad \wedge \mathsf{F}_{[0,b]*_1}(\mathsf{G}_{[0,\frac{\Gamma_{oscP}}{2}]}(s < s^{*1}) \rightarrow\\
&\quad \mathsf{F}_{[0,\frac{\Gamma_{oscP}}{2}]*_2}(\mathsf{H}_{[0,\frac{\Gamma_{oscP}}{2}]}(s > s^{*2})\\
&\quad \wedge \mathsf{F}_{[0,\frac{\Gamma_{oscP}}{2}]*_3}(\mathsf{H}_{[0,\frac{\Gamma_{oscP}}{2}]}(s < s^{*3})\\
&\qquad \wedge \mathrm{abs}\,(s^{*1} - s^{*2}) < 3))))
\end{aligned}$$

In the formula above, the expression on the first row prescribes the existence of the first local minimum, by checking all points within the observation interval $[a, b]$ for the existence of a point (whose time instant is frozen in the first component of the frozen time vector) for which the corresponding signal value is smaller than all other signal values in the interval $[0, \frac{\Gamma_{oscP}}{2}]$; this condition is captured by the sub-formula with the second "globally" operator. The expression on the second row, nesting the "historically" operator within the "eventually", states the presence of a local maximum (whose time instant is frozen in the second component of the frozen time vector), such that all the signal values between the first local minimum and such a point are indeed smaller than the local maximum. Notice that the distance between two neighboring extrema for an oscillation with period Γ_{oscP} is equal to $\frac{\Gamma_{oscP}}{2}$. The expression on the third row checks for the existence of the second local minimum in a similar way; the expression on the fourth row checks the constraint on the peak-to-peak amplitude using the values of the signal in correspondence of the first local minimum and of the local maximum. The remaining part of the formula has the same structure and considers the dual case, in which the first extremum in the oscillatory behavior is a local maximum.

We remark that this specification assumes that the oscillation is regular, i.e., its period is constant and the constraint on the period is specified as "$oscP$=Γ_{oscP}". However, making such an assumption in practice is not reasonable because typically the shape of oscillations is unknown.

(Using local extrema defined through pre-computed derivatives) Property *pOSC* can be expressed using Definition 3 for local extrema, assuming the existence of pre-computed derivatives as separate signals s'_c and s''_c and a 3D frozen time vector.

STL* *pOSC*

$$\begin{aligned}
&\mathsf{G}_{[a,b]}(\mathsf{F}_{[0,b]*_1}((s' = 0 \wedge s'' > 0) \rightarrow\\
&\quad \mathsf{F}_{[0,\frac{\Gamma_{oscP}}{2}]*_2}((s' = 0 \wedge s'' < 0)\\
&\quad \wedge \mathsf{F}_{[0,\frac{\Gamma_{oscP}}{2}]*_3}((s' = 0 \wedge s'' > 0)\\
&\qquad \wedge \mathrm{abs}\,(s^{*1} - s^{*2}) < 3)))\\
&\quad \wedge \mathsf{F}_{[0,b]*_1}((s' = 0 \wedge s'' < 0) \rightarrow\\
&\quad \mathsf{F}_{[0,\frac{\Gamma_{oscP}}{2}]*_2}((s' = 0 \wedge s'' > 0)\\
&\quad \wedge \mathsf{F}_{[0,\frac{\Gamma_{oscP}}{2}]*_3}((s' = 0 \wedge s'' < 0)\\
&\qquad \wedge \mathrm{abs}\,(s^{*1} - s^{*2}) < 3))))
\end{aligned}$$

The structure of the formula above is similar to the one for the case of using Definition 2 for local extrema, except for the direct use of the first and second order derivatives, available as pre-computed signals. The signal values frozen at the local extrema points are used to compute the peak-to-peak amplitude of the oscillations. The same remarks made above in terms of assuming the knowledge of the signal shape also apply in this case.

3.4. Relationship between signals

The property types illustrated in the previous sections deal with only one signal; in this section we present property types characterizing *relationships* between two (or more) signals. We consider two types of signal relationships:

- *functional*, based on the application of a signal transforming function;
- *order*, describing sequences of events/states related to signal behaviors.

3.4.1. Functional relationship

The concept of a functional relationship between two (or more) signals is captured by the application of a signal transforming function to the signals, which yields a new signal based on the

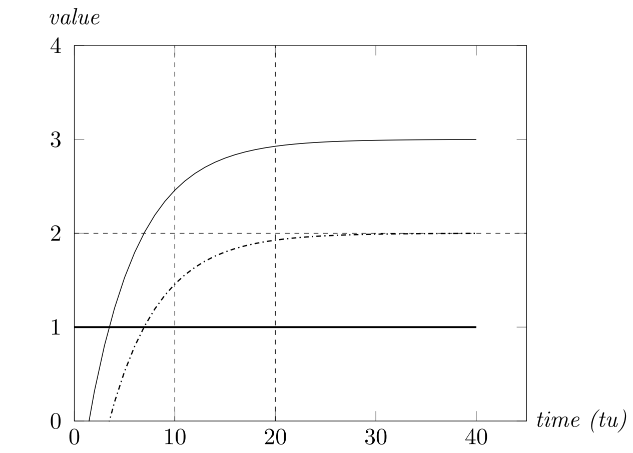


Fig. 7. Signals used to evaluate property *pRSH-F*: the source signals are s_1 (—) and s_2 (-··), the target signal is s_T (**—**); Signal s_T satisfies the property.

semantics of the function. Formally, let $\xi : \mathbb{D}_1 \times \mathbb{D}_2 \rightarrow \mathbb{D}_3$ (with $\xi \in Aux$) be an application-dependent signal transforming function[11] and let s_1 and s_2 be two signals (called *source* signals), with value domains $\mathbb{D}_1$ and $\mathbb{D}_2$ respectively, and domains of definition $I_{s_1} = I_{s_2} = I_s$; the application of ξ to s_1 and s_2 yields a *target* signal s_T over the value domain $\mathbb{D}_3$ defined as $s_T(t) = \xi\,(s_1(t), s_2(t))$, $\forall t \in I_s$. The target signal can then be referred to in the specification of other properties. More precisely, let P be an instance of one of the property types seen in the previous subsections (e.g., a data assertion), with ξ the signal transforming function defined above for the source signals s_1 and s_2. We say that property P holds on the signal representing the functional relationship between s_1 and s_2 captured by ξ iff P holds on the target signal s_T returned by the application of ξ.

For example, let us consider property *pRSH-F*: "The difference between the values of signal s_1 and signal s_2 shall be equal to 1", which contains two parts: a functional relationship part "The difference between the values of signal s_1 and signal s_2..." and a data assertion part "The [difference ...] shall be equal to 1". This property is expressed in *SFO* as follows:

$$\boxed{\text{SFO}\ \ pRSH-F} \quad \forall t \in [0, |s|) : \mathrm{abs}(s_1(t) - s_2(t)) = 1 \tag{5}$$

Fig. 7 shows the two source signals, s_1 plotted with a continuous line (—) and s_2 plotted with a dash-dotted line (-··), as well as the target signal s_T, plotted with a thick line (**—**). Signal s_T is obtained by the application of the signal transforming function ξ defined as $\xi(s_1(t), s_2(t)) \equiv \mathrm{abs}\,(s_1(t) - s_2(t)), \forall t \in I_s$. This signal is then used for the actual evaluation of the data assertion contained in property *pRSH-F*, as if the latter was rewritten as "The value of signal s_T shall be equal to 1"; since signal s_T is equal to 1 across its domain of definition, property *pRSH-F* evaluates to *true*.

3.4.2. Order relationship

This type of signal relationships prescribes a sequence of events/states corresponding to signal behaviors; in practice, it captures the *precedence* and *response* temporal specification patterns proposed in the literature (Dwyer et al., 1999), including their real-time extension (Konrad and Cheng, 2005). More specifically, a precedence property specifies that an event/state (cause) *precedes* another event/state (effect); dually, a response property requires that an event/state (effect) *responds to* the occurrence of another event/state (cause). Notice that a response property allows effects to occur without causes, whereas a precedence property allows causes to occur without subsequent effects. Furthermore, in the context of real-time systems, both a precedence and a response property can include an additional constraint on the temporal distance between a cause and an effect.

When dealing with signals, the events/states used to express order relationships correspond to specific signal behaviors, which can be further expressed (and identified) using one of the property types seen above. More specifically, we define a *signal event* as a change in the signal value (Chechik and Paun, 1999) occurring at a specific time instant, whereas a *signal state* is a signal behavior that holds over an interval delimited by two time boundaries or by the occurrence of two events. In the following, we discuss the concepts of signal events/states in the context of the property types described in the previous sections.

Data assertions. The typical use of data assertions[12] is to represent signal states, as in property *pDAs*: "The signal value shall be greater than or equal to 2". For example, Fig. 8(a) shows a signal that satisfies this property in the interval $[b, c]$.

Another formulation of this type of properties corresponds to signal events. As an example, let us consider property *pDAe*: "The signal value shall become equal to 2". Informally, this property corresponds to a predicate that captures the event of the signal *becoming* equal to 2, i.e., changing from a value different from 2 to the actual value of 2. This behavior can be seen in the signal plotted in Fig. 8b: property *pDAe* holds at time instant c.

Notice that signal events can be used to characterize the boundaries of a signal state: for example, the time instants delimiting the interval in which the state represented by property *pDAs* holds correspond to the time instants in which the event represented by property *pDAe* and by its negation (i.e., "signal s becoming different from 2") occur.

Spike. When a signal satisfies a spike property following the specification template (1) on page 9, the spike behavior of the signal can be associated with three different events, corresponding to the time instants in which the peak point and the two valley points of the spike shape (see Section 3.2) occur. The actual choice of the most relevant event among these three is application-specific. Furthermore, the state induced by such a property type is defined over the interval $[VP_1, VP_2]$; such a state lasts for a duration corresponding to the spike width w.

Oscillation. When a signal satisfies an oscillation property following the specification template (3) in Section 3.3, the oscillatory behavior of the signal can be associated with distinct events, corresponding to the time instants in which the extrema points of the oscillations occur. The choice among these events is application-specific. Moreover, the state induced by such a property type is defined over the interval bounded by the first and last observed extrema of the oscillation.

Functional relationship between signals. Similar to data assertions, functional relationship between signals can represent either signal events (captured by a predicate "*becomes*") or signal states.

Formalization. After defining the concepts of events and states associated with signal property types, we are now ready to formalize the concept of order relationship between signal behaviors.

Given a signal s and an instance P of one of the signal property types described above, we define the *signal event boolean projection* of P on s as the predicate $\sigma^{\mathbb{B}e}_{s,P}(t)$, which evaluates to true

[11] To keep the notation light and without loss of generality, we only consider a signal transforming function with arity 2.

[12] For simplicity, in the following we consider data assertion properties defined on one time interval.

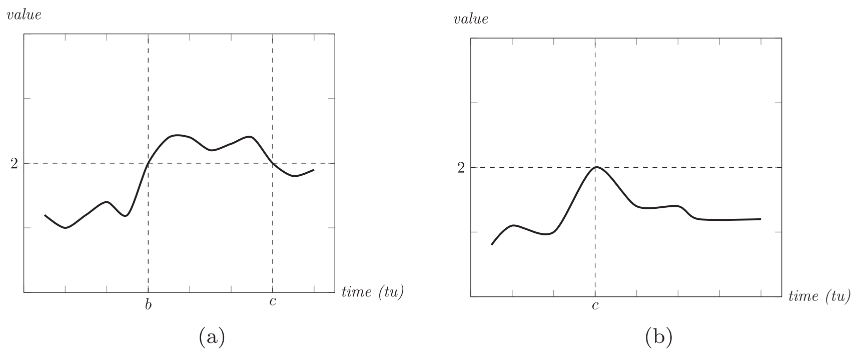


Fig. 8. (a) A signal being in the state characterized by property *pDAs* in the interval $[b, c]$. (b) A signal changing its value to 2 at time instant c, satisfying property *pDAe*.

iff the event associated with the signal behavior specified in P occurs in signal s at time instant t; similarly, we define the *signal state boolean projection* of P on s as the predicate $\sigma^{\mathbb{B}s}_{s,P}(t)$, which evaluates to true iff the state associated with the signal behavior specified in P holds on signal s at time instant t.

Given two signals s_1 and s_2 with domains of definition $I_{s_1} = I_{s_2} = [0, r)$ and lengths $|s_1| = |s_2|$ denoted with $|s|$, and two signal-based properties P_1 and P_2, we say that the event captured by P_2 in s_2 *responds to* (following the "response" pattern in Dwyer et al. (1999)) the event captured by P_1 in s_1 iff the following *SFO* formula evaluates to *true*:

$$\forall t \in [0, |s|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_1,P_1}(t) \rightarrow \left(\exists k \in (t, |s|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_2,P_2}(k)\right) \tag{6}$$

where $\uparrow$ denotes the rising edge operator, defined as $\uparrow s(t) \equiv s(t) = 1 \wedge \exists c \in (0, t) : \forall c' \in (0, c) : s(t - c') = 0$.

If the relevant behavior captured by a property results in a state instead of an event, the formula above becomes:

$$\forall t \in [0, |s|) \colon \sigma^{\mathbb{B}s}_{s_1,P_1}(t) \rightarrow \left(\exists k \in (t, |s|) \colon \sigma^{\mathbb{B}s}_{s_2,P_2}(k)\right) \tag{7}$$

Similarly, we say that the event captured by P_1 in s_1 *precedes* (following the "precedence" pattern in Dwyer et al. (1999)) the event captured by P_2 in s_2 iff the following formula evaluates to *true*:

$$\forall t \in [0, |s|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_2,P_2}(t) \rightarrow \left(\exists k \in [0, t) \colon \uparrow \sigma^{\mathbb{B}e}_{s_1,P_1}(k)\right) \tag{8}$$

When the relevant behavior captured by a property results in a state instead of an event, the formula above becomes:

$$\forall t \in [0, |s|) \colon \sigma^{\mathbb{B}s}_{s_2,P_2}(t) \rightarrow \left(\exists k \in [0, t) \colon \sigma^{\mathbb{B}s}_{s_1,P_1}(k)\right) \tag{9}$$

In some cases, an order relationship may prescribe a temporal distance between the cause and the effect. We assume this distance to be specified as a bound of the form $\bowtie n$, where $\bowtie \in Rel$ and $n \in \mathbb{R}$. In this case the formulae above have to be extended to take the distance into account, by conjoining the clause $\operatorname{abs}(k - t) \bowtie n$ to the consequent. For example, formula (6) will become:

$$\forall t \in [0, |s|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_1,P_1}(t) \\ \rightarrow \left(\exists k \in (t, |s|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_2,P_2}(k) \wedge \operatorname{abs}(k - t) \bowtie n\right) \tag{10}$$

Notice that when one property induces a state and the other induces an event, the resulting formula for the corresponding order relationship is obtained by opportunely combining the occurrences of the signal boolean projection functions for states and events, following one of the above templates.

Order relationship properties can be defined recursively, i.e., when the cause and/or effect sub-property is also an order relationship. In these cases, we consider an event-based interpretation of the cause/effect sub-property.

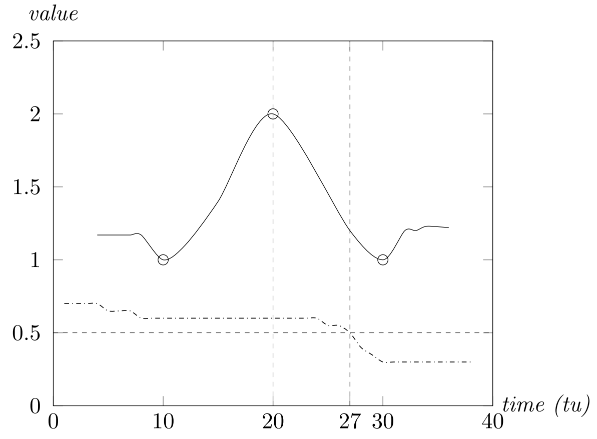


Fig. 9. Signals s_1 (—) and s_2 (-··) used to evaluate property *pRSH-O*; the property holds.

As an example of order relationship property, let us consider the following response property *pRSH-O*: "If in signal s_1 there is a spike with a maximum width of 30 tu and a maximum amplitude of 1, then — within 10 tu — the value of signal s_2 shall become less than 0.5". Assuming we use an event-based interpretation of both cause and effect sub-properties, we can rewrite the property as *pRSH-O′* : "If there is an event corresponding to [signal s_1 having a spike with a maximum width of 30 tu and a maximum amplitude of 1] then—within 10 tu—there shall be an event corresponding to [signal s_2 becoming less than 0.5]". In this instance of the response pattern, the cause is represented by the spike property "In signal s_1 there is a spike with a maximum width of 30 tu and a maximum amplitude of 1", whereas the effect is represented by the data assertion property "Signal s_2 shall become less than 0.5"; furthermore, the temporal distance between the cause and the effect can be at most 10 tu. We refer to the cause and effect sub-properties as P_1 and P_2, respectively.

The specification of property *pRSH-O* in *SFO* is the following:

SFO *pRSH-O*
$$\forall t \in [0, |s_1|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_1,P_1}(t) \\ \rightarrow \left(\exists k \in (t, |s_2|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_2,P_2}(k) \wedge \operatorname{abs}(k - t) \leq 10\right) \tag{11}$$

where $\sigma^{\mathbb{B}e}_{s_1,P_1}$ and $\sigma^{\mathbb{B}e}_{s_2,P_2}$ are the signal event boolean projection predicates.

We evaluate the property with respect to the two signals shown in Fig. 9, s_1 plotted with a continuous line (—) and s_2 plotted with a dash-dotted line (-··). In this example, we assume that the signal boolean projection predicate for spike properties (used for the evaluation of the cause sub-property) is defined such that it is true at the actual time instant at which the spike peak point occurs (i.e., 20 tu). By looking at Fig. 9, we see that property *pRSH-O* holds on s_1 and s_2 because the event captured by the effect sub-property (the change of value of s_2 happening at time instant 27 tu) responds to the occurrence of the event associated with the cause sub-property within the prescribed time bound (since $\mathrm{abs}(27\ \mathrm{tu} - 20\ \mathrm{tu}) = 7\ \mathrm{tu} < 10\ \mathrm{tu}$).

3.4.3. Transient behaviors

We consider transient signal behaviors (i.e., behaviors of a signal when changing from the current value to its target value) as a special case of order relationship. This category includes *rise time* (and *fall time*) and *overshoot* (and *undershoot*) properties.

Rise time (fall time). We say that a signal exhibits a *rising* (dually, *falling*) behavior when its value increases (decreases) towards a target value. Informally speaking, a property on the *rise (fall) time* defines a constraint on the time by which the signal reaches the target value. More specifically, it defines a constraint on the temporal distance between two events: (1) a (generic) cause event, also called *trigger event*, that coincides with the signal starting to manifest a transient behavior; (2) an effect event that represents the signal reaching the target value.

Fig. 10a depicts a signal exhibiting a rising behavior starting from time instant st. The signal rises monotonically from the value $s(st)$ and reaches the target value s_{target} at time instant c; the time interval $[st, c]$ is called *rise interval*. The left bound of the rise interval, also called *trigger time*, corresponds to the time instant at which the trigger event occurs. The right bound of the rise interval corresponds to the occurrence of the effect event, in which signal s reaches the target value. The trigger time can also be expressed in terms of an absolute time reference value; in such a case, the trigger event is the event in which a special *clock* signal reaches a certain value.

A rise time property defines a constraint on the right bound of the rise interval. More formally, given two signals s_{tr} and s with domains of definition $I_{s_{tr}} = I_s = [0, r)$, let P_{tr} and P be two signal-based properties. Property P_{tr} captures the trigger event defined in terms of the behavior of s_{tr}; property P captures the event of s reaching the target value. A rise time property bounds the rise time of s by a threshold $RT \in \mathbb{N}$ (indicated by the end-user); such a property holds iff the following *SFO* formula evaluates to *true*:

$$\forall st \in [0, |s_{tr}|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_{tr},P_{tr}}(st) \rightarrow \big(\exists k \in [st, st + RT] \colon \uparrow \sigma^{\mathbb{B}e}_{s,P}(k)\big) \quad (12)$$

A stricter definition requiring signal s to rise (strictly) monotonically can be expressed by adding the conjunct $\forall j \in [st, st + k) \colon \forall j' \in (j, st + k] : s(j) < s(j')$ to the consequent in the formula above.

A fall time constraint can be expressed in a similar way, replacing the relational operators with their duals.

As an example, let us consider the rise time property *pRT*: “If signal s_{tr} becomes greater than 1, then signal s shall reach the target value of 2 within at most 8 tu”. The trigger event in this property is represented by the data assertion property P_{tr}: “The value of signal s_{tr} becomes greater than 1”. The effect sub-property of this order relationship property can be specified with the data assertion property P: “The value of signal s shall become greater than 2”. The constraint on the rise time is 8 tu. Property *pRT* can be expressed in *SFO* as:

SFO *pRT*
$$\forall st \in [0, |s_{tr}|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_{tr},P_{tr}}(st) \rightarrow \big(\exists k \in [st, st + 8] \colon \uparrow \sigma^{\mathbb{B}e}_{s,P}(k)\big) \quad (13)$$

We evaluate property *pRT* with respect to signal s on the two signals shown in Fig. 10b: s_1 plotted with a thick line (—) and s_2 plotted with a thin line (—). In the figure, an arrow at timestamp 4 tu denotes the trigger time st corresponding to the trigger event captured by property P_{tr} for signal s_{tr} drawn with a dash-dotted line (-··). The maximum allowed value for the right bound of the rise interval ($st + RT = 4 + 8 = 12$ tu) is indicated with a red, vertical dashed line. Signal s_1 satisfies the property because it reaches the target value (2) at time instant 9 tu $< st + RT$. Signal s_2 violates the property because it does not reach the target value by time instant $st + RT = 12$ tu.

The variant *pRT-monot* of property *pRT* with a monotonicity constraint can be expressed in *SFO* as:

SFO *pRT-monot*
$$\forall st \in [0, |s_{tr}|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_{tr},P_{tr}}(st) \rightarrow \big(\exists k \in [st, st + 8] \colon \uparrow \sigma^{\mathbb{B}e}_{s,P}(k) \wedge \forall j \in [st, st + k) \colon \forall j' \in (j, st + k] : s(j) < s(j')\big) \quad (14)$$

Overshoot (undershoot). We say that a signal exhibits an *overshoot* (dually, *undershoot*) behavior when it exceeds (goes below) its target value.[13] Informally speaking, an overshoot property specifies the maximum signal value, above the target value, that a signal can reach when overshooting within a certain time interval; an undershoot property is defined dually.

Fig. 11a depicts a signal exhibiting an overshoot behavior starting from time instant st. This time instant is the *trigger time* and can be specified in different ways, as discussed above in the context of rise time properties. The signal rises from the value $s(st)$ and overshoots the target value s_{target} after time instant c, reaching the maximum magnitude s_{max} at time instant b. The time interval $[c, c + OI]$ is called *overshoot interval*; its width OI is specified by the end-user. This signal overshoots the target value s_{target} by an *overshoot value* $O_s = s_{max} - s_{target}$. An overshoot property defines a boundary on the overshoot value within the overshoot interval; such a boundary is expressed either with an absolute value or with a relative value with respect to the target value.

Similarly to the case of rise time specification, given two signals s_{tr} and s, let P_{tr} and P be two signal-based properties. Property P_{tr} captures the trigger event defined in terms of the behavior of s_{tr}; property P captures the event of signal s reaching the target value. An overshoot property bounds the overshoot of s by a threshold $OI \in \mathbb{N}$; such a property holds iff the following *SFO* formula evaluates to *true*:

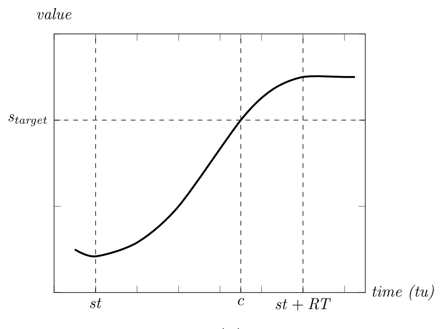
$$\forall st \in [0, |s_{tr}|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_{tr},P_{tr}}(st) \rightarrow (\exists k \in [st, |s|) \colon \uparrow \sigma^{\mathbb{B}e}_{s,P}(k) \wedge \forall i \in [k, k + OI] : s(i) \leq s_{max}) \quad (15)$$

A monotonicity constraint can be added to the formula above in the same way as done for the case of rise time properties. An undershoot constraint can be expressed in a similar way, replacing the relational operators with their duals.

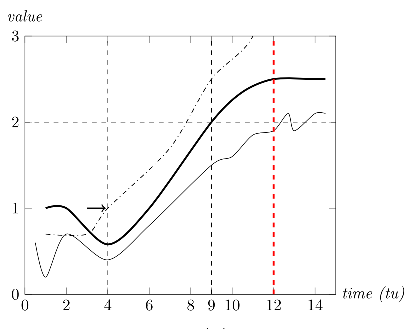
As an example, let us consider property *pOSH*: “If signal s_{tr} becomes greater than 1, then signal s may overshoot the target value of 1 by at most 2 within an overshoot interval of at most 6 tu”. As we did above for the *pRT* property, the trigger event in *pOSH* is represented by the data assertion property P_{tr}. The remaining part of the property represents the effect sub-property. The corresponding *SFO* formula is the following:

SFO *pOSH*
$$\forall st \in [0, |s_{tr}|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_{tr},P_{tr}}(st) \rightarrow (\exists k \in [st, st + |s|) \colon \uparrow \sigma^{\mathbb{B}e}_{s,P}(k) \wedge \forall i \in [k, k + 6] : s(i) \leq 3) \quad (16)$$

[13] Other definitions of overshoot also constrain the behavior of the signal after it exceeds (goes below) the target value, e.g., by requiring it to converge back to the target value.

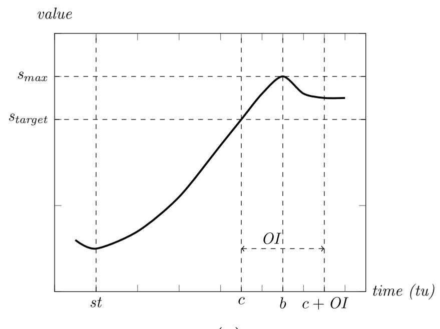


(a)

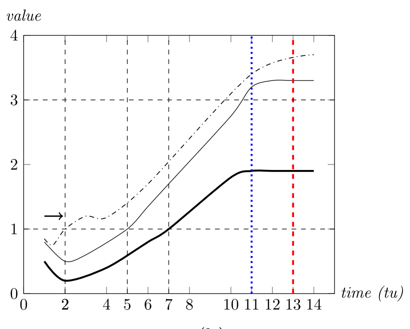


(b)

Fig. 10. (a) Main concepts related to the specification of rise time. (b) two signals used to evaluate property *pRT*: signal s_1 (—) satisfies the property, whereas s_2 (—) violates it.



(a)



(b)

Fig. 11. (a) Main concepts related to the specification of overshoot. (b) two signals used to evaluate property *pOSH*: signal s_1 (—) satisfies the property, whereas s_2 (—) violates it.

The variant of property *pOSH-monot* with a monotonicity constraint can be expressed in *SFO* as:

$$\text{SFO } \textit{pOSH-monot} \quad \forall st \in [0, |s_{tr}|) \colon \uparrow \sigma^{\mathbb{B}e}_{s_{tr}, P_{tr}}(st) \rightarrow (\exists k \in [st, st + |s|) \colon \uparrow \sigma^{\mathbb{B}e}_{s,P}(k) \wedge \forall i \in [k, k+6] : s(i) \le 3 \wedge \forall j \in [st, st+k) \colon \forall j' \in (j, st+k] : s(j) < s(j')) \tag{17}$$

We evaluate property *pOSH* with respect to signal *s* on the two signals shown in Fig. 11b: s_1 plotted with a thick line (—) and s_2 plotted with a thin line (—). In the figure, an arrow at timestamp 2 tu denotes the trigger time *st* corresponding to the trigger event captured by property P_{tr} for signal s_{tr}, drawn with a dash-dotted line (-··). After this time instant, both s_1 and s_2 rise reaching the target value of 1 at time instants 7 tu and 5 tu, respectively. We consider a threshold expressed as a relative value with respect to the target value; i.e., $s_{max} = s_{target} + 2 = 1 + 2 = 3$. The maximum allowed value for the right bound of the overshoot interval for s_1 ($7\,\text{tu} + OI = 7\,\text{tu} + 6\,\text{tu} = 13\,\text{tu}$) is indicated with a red, vertical dashed line. Similarly, in the case of s_2, the right bound for the overshoot interval ($5\,\text{tu} + OI = 5\,\text{tu} + 6\,\text{tu} = 11\,\text{tu}$) is drawn with a blue, dotted vertical line. Signal s_1 satisfies the property because its overshoot value is below the threshold within the overshoot interval [7 tu, 13 tu]; signal s_2 violates the property as its overshoot value exceeds the threshold within the overshoot interval [5 tu, 11 tu].

3.4.4. *Alternative formalizations*

The capability of expressing functional relationship properties in *STL* and *STL** depends on the possibility, in the chosen language, of expressing a certain property type on the target signal resulting from the transforming function.

Similarly, expressing order relationship properties in *STL* and *STL** requires that the cause and effect sub-properties can be expressed in the chosen formalism. For example, the cause sub-property of property *pRSH-O* cannot be expressed in *STL*; however, it can be expressed in *STL** as explained in Section 3.2 (page 12).

The same remarks made above for the general case of order relationships apply also to the case of rise time and overshoot properties. In addition, we remark that the specification of such properties containing a monotonicity constraint requires keeping track of the signal values seen throughout the rise/overshoot interval; this is not supported in *STL* but can be expressed in *STL** using the freeze operator.

4. Expressiveness

Another challenge in using signal-based temporal properties for expressing requirements of CPSs is the expressiveness of the specification languages used for defining such properties. Starting from the seminal work on *STL*, there have been several proposals of languages that extend more traditional temporal logics like LTL to support the specification of signal-based behaviors. For example, in the previous section, we formally specified all property types included in our taxonomy using *SFO* and, when applicable, also using *STL* and *STL**. All these languages have different levels of expressiveness when it comes to describing certain signal behaviors.

In this section, we summarize and discuss the expressiveness of these state-of-the-art temporal logics *with respect to the property types included in our taxonomy*. We remark that we do not aim to provide a complete and formal treatment of the expressiveness of these temporal logics; our main goal is to guide engineers to

Table 1
Expressiveness of *STL*, *STL**, and *SFO* with respect to the property types included in the taxonomy in Fig. 1.

Property type	Example	Formalism		
		STL	*STL**	*SFO*
Data assertions (DA)	*pDA*	+	+	+
Spikes				
SPK with amplitude	*pSPK1*	−	+	+
SPK with slope	*n/a*	−	+	+
SPK with width	*pSPK1*	−	±	+
SPK - punctual derivatives		−	−	+
SPK analytical formulation		−	+	+
SPK pre-computed derivatives	*pSPK3*	+	+	+
Oscillations				
OSC with amplitude	*pOSC*	−	±	+
OSC with period	*pOSC*	±	±	+
OSC punctual derivatives		−	−	+
OSC analytical formulation		−	+	+
OSC pre-computed derivatives		+	+	+
Relationship between signals				
RSH-F	*pRSH-F*	±	±	+
RSH-O	*pRSH-O*	±	±	+
Transient Behaviors				
RT (FT) with monotonicity	*pRT-monot*	−	+	+
RT (FT)	*pRT*	+	+	+
OSH (USH) with monotonicity	*pOSH-monot*	−	+	+
OSH (USH)	*pOSH*	+	+	+

choose a specification formalism based on their needs in terms of the property types to express.

Table 1 provides an overview of the expressiveness of *STL*, *STL**, and *SFO* with respect to the property types included in the taxonomy. The "+ " and "−" symbols denote, respectively, support (or lack of support) for a certain property type; the "±" symbol indicates that the property type can be expressed under certain assumptions. Note that in the table, we also list property subtypes based on a particular feature. For example, "SPK with amplitude" indicates a spike property type (see Fig. 1 for the acronyms) with a constraint on the amplitude. In addition, we list as property subtypes (e.g., "SPK pre-computed derivatives") the three definitions to express the predicates for local extrema for spikes and oscillations (introduced in Section 3.2, page 10). In the second column, we provide examples of properties corresponding to the property (sub)type indicated in the first column.

At a glance, the table shows that *SFO* can be used to express all the property types considered in this paper. *STL** can be used to express most of the property types included in our taxonomy, provided that some assumptions are made (see below). *STL* cannot be used to express all the property types; this is due to the lack of support for referring to signal values at an instant in which a certain property was satisfied. This limitation impacts on the specification of properties that constrain signal values at different time instants, such as spike and oscillation properties. In the following, we discuss the expressiveness for the various property types in details, mainly focusing on *STL* and *STL**.

Data assertion. All three formalisms can express data assertion properties. This is expected since the three logics we have considered were proposed with the goal of expressing predicates on a signal value.

Spike. A formalism supports our definition of spike properties if it allows for the definition of (1) two predicates for detecting local extrema, and (2) constraints on features of the signal shape (e.g., amplitude).

STL can be used to define the predicates for detecting local extrema only through Definition 3 (as indicated with the "+ " mark in the table), which assumes the availability of the first and second order derivatives of a signal. Furthermore, it cannot be used to express spike properties that constrain the spike amplitude or slope, since they refer to signal values at different points in the signal timeline. For example, the only spike property among those presented in the previous section that can be expressed in *STL* is *pSPK3*, because it uses pre-computed derivative signals and does not constrain the spike amplitude.

*STL** can be used to define the predicates for detecting local extrema using two of the definitions we propose (Definition 2 — analytical formulation, and Definition 3 — pre-computed derivatives). Furthermore, it can be used to express constraints on the different features of the signal shape. However, to do so, one has to assume the knowledge of the signal shape, since it uses the two components of the width w_1 and w_2 as defined on page 9. However, making such an assumption in practice is not reasonable because typically the shape of a spike is unknown. Finally, since *STL** (and *STL*) cannot refer to the value of the signal at arbitrary time points, properties defined using local extrema expressed according to Definition 1 (punctual derivatives) cannot be specified.

Oscillation. The expressiveness results in terms of oscillation properties mirror those for spike properties, since the former property type can be seen as an extension of the latter.

STL can be used to express oscillation properties when the oscillatory behavior is defined through the sequence of alternating local extrema, in which the latter are expressed using Definition 3. However, as in the case of spike properties, *STL* cannot be used to express constraints on the oscillation amplitude.

Again, similarly to the case of spike properties, *STL** supports Definitions 2 and 3 for defining local extrema and can be used to express constraints on the different features of an oscillatory behavior. However, such formulations (including the one based on Definition 3 for *STL*) require to assume that (1) the oscillation is regular; (2) its period is known a priori. These assumptions are required to express distance constraints between local extrema. Once again, in practice these assumptions are not realistic because typically the shape of an oscillatory behavior is unknown.

Relationship between signals. Expressing functional relationship properties boils down to expressing a certain property type on the target signal resulting from the transforming function. The type of the property in which the target signal is used ultimately affects (e.g., in case of a spike property) the expressiveness for this type of properties. Furthermore, one has to consider whether the transformed (target) signal is available as a pre-computed signal or as function of other signals; in the latter case, only SFO supports function symbols.

A necessary requirement to express order relationship properties is the support for temporal operators that can capture the *precedence* and *response* temporal specification patterns (Dwyer et al., 1999). This is possible in *STL* and *STL** through the "*Until*" operator and in *SFO* by means of explicit quantification on the time variable. Another requirement is that the properties corresponding to the "cause" and "effect" of an order relationship can be expressed in the chosen formalism; as shown in Table 1, only SFO fulfills such a requirement.

Transient behaviors. Transient behavior properties *without monotonicity constraints* can be expressed with all three formalisms, assuming the trigger property can be expressed in the chosen formalism. When a monotonicity constraint is used (as it is the case in properties *pRT-monot* and *pOSH-monot*), properties cannot be expressed in *STL* because one cannot compare the value of the signals at two different time instants.

Monitoring algorithms and tools. When discussing the expressiveness of specification languages, it is also important to review the complexity of the corresponding verification algorithms and the availability of tools implementing them. Below we discuss the computational complexity of tools for *(offline) monitoring* of *STL*, *STL**, and *SFO* properties; we focus on monitoring because it is one of the most used V&V techniques for CPSs (Bartocci et al., 2018).

The complexity of monitoring *STL* is $O(k \cdot n)$ where k is the number of sub-formulae and n is the number of intervals on which the signal is defined (Maler and Nickovic, 2004). For *STL**, the monitoring complexity is (similarly to *STL*) polynomial in the number of intervals on which the signal is defined and the size of the syntactic parse tree of the formula; however, it is exponential in the number of nested freeze operators in the formula (Brim et al., 2014). The monitoring complexity of *SFO* is $2^{(m+n)^{2^{O(k+l)}}}$, where n is the length of the trace, m is the length of the formula, k is the number of quantifiers in the formula, and l is the number of occurrences of function symbols in the formula; for a fragment of *SFO* in which intervals have bounded duration, the complexity is $n \cdot 2^{(m+j)^{2^{O(k+l)}}}$, where n, m, k, l are defined as above, and j is the maximum number of linear segments in the trace during any time period as long as the sum of the absolute values of all time constants in the formula (Bakhirkin et al., 2018). In general, one can see that the complexity of the monitoring problem becomes harder for more expressive languages like *STL** and *SFO*.

In terms of monitoring tools, *STL* is supported both by *offline* tools — such as *AMT* (Ničković et al., 2018; Nickovic and Maler, 2007) (a stand-alone GUI tool with qualitative semantics), *Breach* (Donzé, 2010) and *S-Taliro* (Fainekos et al., 2012) (two *Matlab*® plugins with quantitative semantics) — and by *online* tools, such as the `rtamt` library (Ničković and Yamaguchi, 2020), which automatically generates online monitors with robustness semantics from *STL* specifications. For *STL**, a prototype implementation in *Matlab* is mentioned in the original paper (Brim et al., 2014) but it has not been made available; furthermore, robustness analysis is supported by an extension of the *Parasim* tool (Brim et al., 2013). No tool implementation is available for *SFO* at the time of writing this paper.

Recently, some of the authors have developed *SB-TemPsy* (Boufaied et al., 2020), a model-driven trace checking approach for the property types included in the taxonomy proposed in this paper. *SB-TemPsy* includes *SB-TemPsy-DSL*, a domain-specific specification language for signal-based properties, as well as the corresponding monitoring algorithm and tool, called *SBTemPsy-Check*. The complexity of the pattern-specific trace checking algorithm implemented in *SBTemPsy-Check* is polynomial in the size of the trace for all property types included in this taxonomy except for data assertions, for which the complexity is linear (in the size of the trace).

In conclusion, *with respect to the property types identified in our taxonomy*, *STL* has limited expressiveness, restricting its application in practice to simple property types (e.g., data assertion); nevertheless, it has a good support from a number of tools. *STL** is more expressive than *STL* provided that some assumptions (e.g., on the signal shape) are made; however, such assumptions are impractical. In addition, *STL** suffers from the limited tool support. *SFO* is the most expressive language for the property types defined in our taxonomy; however, its application in V&V activities is still challenging given the computational complexity of associated monitoring algorithms and the lack of tools.

5. Application to an industrial case study

We applied our taxonomy of signal-based properties to classify the requirements specifications of a case study provided by our industrial partner *LuxSpace Sàrl*,[14] a system integrator of micro-satellites. Our goal is to show (1) the feasibility of expressing requirements specifications of a real-world CPS using the property types included in our taxonomy; (2) the completeness of our taxonomy, so that all requirements specifications of the case study can be defined using the property types included in our taxonomy.

Table 2
Distribution of property types in the case study.

Property type	Total (Main)	Total (Sub)
Data assertion	7	49
Spike	1	1
Oscillation	1	0
Functional relationship	17	0
Order relationship	15	0
▷ Fall time	0	1

The case study deals with a satellite sub-system called *Attitude Determination and Control System* (ADCS), which is responsible for autonomously controlling the attitude of the satellite, i.e., its orientation with respect to some reference point. The ADCS is mainly composed of sensors (e.g., gyroscope, sun sensors), actuators (e.g., reaction wheels, magnetic torquer), and on-board software (e.g., control algorithms). During flight, the ADCS can be in four different modes (represented with an enumeration as integer values), which determine the capabilities of the satellite: *idle* (IDLE), *Safe Mode* (SM), *Normal Mode Coarse* (NMC), and *Normal Mode Fine* (NMF); the logic controlling the switch among modes is encoded in a state machine. Overall, this sub-system has the typical characteristics of a CPS, with a deep intertwining of hardware and software.

The documentation of the ADCS includes 41 specifications written in English. Two of the authors carefully analyzed these specifications, discussed and (in some cases) refined them with a domain expert, and finally classified them using one of the property types in our taxonomy; the resulting classification was then validated by the domain expert. Table 2 shows the number of specifications classified for each property type (column "Total (Main)"); since properties of type functional and order relationship include additional properties as sub-properties (e.g., the type of the "cause" or "effect" sub-property in an order relationship), we indicate their number separately under column "Total (Sub)". From the table we can conclude that *all requirements specifications of the case study could be classified using the property types included in our taxonomy*; this is an indication of the completeness of our taxonomy. In the following we provide some insights for each property type, derived from our classification exercise. We remark that the signal names used in the specifications correspond to the signals of a FES (Functional Engineering Simulator) in Matlab; when possible, we preserved the original signal name.

Data assertion properties (Table 3). This is the most represented category, if one considers the sub-properties included in the properties of type functional and order relationship. The three time-constrained data assertions show different interval types used in such properties. For example, in property P6 both boundaries of the interval are explicitly mentioned. In property P5, only the left boundary is explicitly indicated (with the expression "Starting from 2000 s"), whereas the right boundary is implicit and is assumed to be the end of the (finite) signal. Finally, in property P7 the interval is singular (i.e., the two boundaries coincide) and corresponds to a single time point (as in the expression "At 2000 s"). To express the latter using one of the logic-based

[14] https://luxspace.lu/.

Table 3
Data assertion properties in the case study.

ID	Property
Untimed data assertions	
P1	The value of signal *currentADCSMode* shall be equal to *NMC*, *NMF* or *SM*
P2	The value of signal *pointing_error_above_20* shall be equal to 0 or 1
P3	The value of signal *pointing_error_under_15* shall be equal to 0 or 1
P4	The value of signal *RWs_angular_velocity* shall be equal to 816.814 rad/s
Time-constrained data assertions	
P5	Starting from 2000 s, the value of signal *pointing_error* shall be less than 2°
P6	Between 1500 s and 2000 s, the value of signal *RWs_angular_momentum* shall be less than $0.35\,\mathrm{N}\cdot\mathrm{m}\cdot\mathrm{s}$
P7	At 2000 s the value of signal *pointing_error* shall be between 0° and δ°

Table 4
Spike and oscillation properties in the case study.

ID	Property
Spike	
P8	Between 2000 s and 7400 s, in signal *pointing_error* there shall exist a spike with a maximum width of 20 s
Oscillation	
P9	Between 2000 s and 7400 s, signal *pointing_error* shall exhibit oscillations with a period greater than or equal to 0.01 s

formalizations illustrated above, which does not allow singular intervals (e.g., *STL*), one has to rewrite a singular interval $[a, a]$ as $[a - \epsilon, a + \epsilon]$, for a small $\epsilon > 0$.

We remark that time-constrained data assertions can be used to specify system-level properties such as system stabilization. For example, property P5 was originally expressed as "The stabilization time of signal *pointing_error*, when stabilizing below 2 degrees, shall be under 2000 s"; through the interaction with the domain expert, we further refined it into the version shown in Table 3. The refinement step was straightforward and consisted of rewriting the system-level property (i.e., stabilization) into a low-level one (of type "data assertion"), by expanding the definitions of domain concepts.

Spike and oscillation properties (Table 4). We identified one spike property (P8); furthermore an additional spike property is included in an order relationship property (P41). Both spike properties refer to one feature ("width").

We also identified one oscillation property (P9), which refers to the "period" feature. Initially, the property was defined in the frequency domain (which we did not discuss in this paper). After discussing it with the domain expert, we converted it into a property defined on the time domain by changing the corresponding constraint. This type of transformation is straightforward as it only requires to convert the units in the property (e.g., a 100 Hz frequency is converted into a 0.01 s period).

All three properties include an observation interval. In properties P8 and P9, it is defined explicitly using absolute time boundaries (with the expression "between 2000 s and 7400 s"). In property P41, the observation interval is defined through the event representing the left boundary (denoted with "the value of signal *pointing_error* after 16 200 s goes below the pointing accuracy threshold of 2°") and the duration (5400 s) representing the right boundary.

Functional relationship properties (Table 5). These properties were expressed using several signal transforming functions, such as modulus (P10–P20), vector elements sum (P21), angular difference (P22–P23), scalar difference (P24–P26), and differentiation (P26). Notice that property P26 contains nested applications of signal transforming functions (i.e., the second operand of the scalar difference is the result of the application of the derivative).

In all properties, the signal resulting from the application of the transforming function is used in a data assertion property (see column "Subtype" in Table 5[15]).

Order relationship properties (Table 6). All the order relationship properties we classified were instances of the "response" pattern (see Section 3.4.2); we did not encounter any instance of the "precedence" pattern.

Some properties (P35–P38) contain nested properties of type "order relationship", meaning that the effect of the response pattern is represented by another property of type "order relationship". For example, in property P36, the top-level response property has "the value of signal *currentADCSMode* is equal to *NF*" as cause and "if the value of signal *RWs_command* becomes greater than 0, then the value of signal *pointing_error* shall be less than 2°" as effect. The latter is another response property that can be further decomposed into the cause "the value of signal *RWs_command* becomes greater than 0" and the effect "the value of signal *pointing_error* shall be less than 2°". The same group of properties also includes a temporal distance constraint (expressed with "within") as part of the nested response property.

As shown in column "Subtype" of Table 6, all the sub-properties used as "cause" and the vast majority of the sub-properties used as "effect" were data assertions. For example, in property P27 both the cause "the value of signal *not_Eclipse* is equal to 0" and the effect "the value of signal *sun_currents* shall be equal to 0" are data assertions. This is reflected in the third column of Table 6, with the notation "DA-DA".

Regarding transient behaviors, we only encountered one property of type "fall time", used as effect of the response property P30. Other types of properties (e.g., rise time, overshoot) were not present in this case study.

Summing up, through this case study we have shown the *feasibility* of expressing requirements specifications of a real-world CPS using the property types included in our taxonomy. In the vast majority of the cases, the mapping from a specification written in English to its corresponding property type defined in the taxonomy was straightforward. In two cases, the specifications had to be refined, either by expressing a system-level property into a low-level one (e.g., stabilization being expressed as a (time-constrained) data assertion) or by converting a property defined in the frequency domain into the corresponding one defined in the time domain (e.g., in the case of an oscillation property); both types of refinement were simple and intuitive (with the help of a domain expert). Furthermore, the case study has shown the *completeness* of our taxonomy, since all requirements specifications of the case study could be classified using the property types included in our taxonomy.

Guided by the mapping to one of the property types included in our taxonomy, and by means of the formalization presented

[15] See Fig. 1 for the acronyms used in column "Subtype" of Tables 5 and 6.

Table 5
Properties of type "functional relationship" in the case study.

ID	Property	Subtype
P10	The modulus of signal *sat_init_angular_velocity_degree* shall be less than or equal to $3\,^{\circ}/\mathrm{s}$	DA
P11	After 2000 s, the modulus of signal *sat_real_angular_velocity* shall be less than or equal to $1.5\,^{\circ}/\mathrm{s}$	DA
P12	The modulus of signal *sat_target_attitude* shall be equal to 1	DA
P13	After 2000 s, the modulus of signal *sat_target_angular_velocity* shall be less than or equal to $1.5\,^{\circ}/\mathrm{s}$	DA
P14	The modulus of signal *sat_estimated_attitude* shall be equal to 1	DA
P15	After 2000 s, the modulus of signal *sat_estimated_angular_velocity* shall be less than or equal to $1.5\,^{\circ}/\mathrm{s}$	DA
P16	The modulus of signal *sat_angular_velocity_measured* shall be less than or equal to $1.5\,^{\circ}/\mathrm{s}$	DA
P17	The modulus of signal *earth_mag_field_in_body_measured* shall be less than or equal to 60 000 nT	DA
P18	The modulus of signal *sun_direction_ECI* shall be equal to 1	DA
P19	After 2000 s, the modulus of signal *sat_target_angular_velocity_safe_spin_mode* shall be less than or equal to $1.5\,^{\circ}/\mathrm{s}$	DA
P20	The modulus of signal *RWs_torque* shall be less than or equal to $0.015\,\mathrm{N}\cdot\mathrm{m}$	DA
P21	The elements sum of vector *sun_sensor_availability* shall be at most 3	DA
P22	At 2000 s, the angular difference between signals *q_real* and *q_estimate_attitude* shall be between 0° and δ°	DA
P23	At 2000 s, the angular difference between signals *q_target_attitude* and *q_estimate* shall be between 0° and δ°	DA
P24	The difference between signal *sat_estimated_angular_velocity* and signal *sat_real_angular_velocity* shall be between $0\,^{\circ}/\mathrm{s}$ and $\delta^{\circ}/\mathrm{s}$	DA
P25	The difference between signal *sat_angular_velocity_measured* and signal *sat_real_angular_velocity* shall be between $0\,^{\circ}/\mathrm{s}$ and $\delta^{\circ}/\mathrm{s}$	DA
P26	The difference between signal *RWs_torque* and the derivative of signal *RWs_angular_momentum* shall be equal to $0\,\mathrm{N}\cdot\mathrm{m}$	DA

Table 6
Properties of type "order relationship" in the case study.

ID	Property	Subtype
P27	If the value of signal *not_Eclipse* is equal to 0, then the value of signal *sun_currents* shall be equal to 0	DA-DA
P28	If the value of signal *pointing_error_under_15* is equal to 1, then the value of signal *pointing_error_above_20* shall be different from 1	DA-DA
P29	If the value of signal *pointing_error_above_20* is equal to 1, then the value of signal *pointing_error_under_15* shall be different from 1	DA-DA
P30	If the value of signal *RWs_command* is equal to 0, then the value of signal *RWs_angular_velocity* shall monotonically decrease to 0 rad/s within 60 s	DA-FT
P31	If the value of signal *RWs_angular_momentum* is greater than $0.35\,\mathrm{N}\cdot\mathrm{m}\cdot\mathrm{s}$, then the value of signal *RWs_torque* shall be equal to $0\,\mathrm{N}\cdot\mathrm{m}$	DA-DA
P32	If the value of signal *currentADCSMode* is equal to *NMC*, then the value of signal *control_error* shall be greater than or equal to 10°	DA-DA
P33	If the value of signal *control_error* is less than 10°, then the value of signal *currentADCSMode* shall be equal to *NMF*	DA-DA
P34	If the value of signal *currentADCSMode* is equal to *NMF*, then the value of signal *control_error* shall be less than or equal to 15°	DA-DA
P35	If the value of signal *currentADCSMode* is equal to *NMF*, then if the value of signal *RWs_command* becomes greater than 0, then the value of signal *pointing_error* shall be less than 2° within 180 s	DA-DA-DA
P36	If the value of signal *currentADCSMode* is equal to *NMF*, then if the value of signal *RWs_command* becomes greater than 0, then the value of signal *control_error* shall be less than 0.5° within 180 s	DA-DA-DA
P37	If the value of signal *currentADCSMode* is equal to *NMF*, then if the value of signal *Not_eclipse* becomes 1, then the value of signal *knowledge_error* shall be less than 1 within at most 900 s	DA-DA-DA
P38	If the value of signal *currentADCSMode* is equal to *SM*, then if the value of signal *RWs_command* becomes greater than 0, then the value of signal *RWs_angular_momentum* shall be less than $0.25\,\mathrm{N}\cdot\mathrm{m}\cdot\mathrm{s}$ within at most 900 s	DA-DA-DA
P39	If the value of signal *currentADCSMode* is equal to *SM*, then the difference between signal *real_Omega* and signal *target_Omega* shall be equal to 0 within at most 10 799 s	DA-DA
P40	If the value of signal *not_Eclipse* is equal to 1, then the value of signal *sun_angle* shall be less than 45°	DA-DA
P41	If, starting from 16 200 s, the value of signal *pointing_error* goes below the pointing accuracy threshold of 2°, then in signal *pointing_error* there shall exist a spike with a maximum width of 600 s in an interval of 5400 s	DA- SPK

in Section 3, an engineer can obtain a formal specification of a property (e.g, in *SFO*), which can then be used in the context of V&V activities (e.g., as test oracle).

Threats to validity. The results regarding the *feasibility* of expressing requirements specifications of a real-world CPS and the *completeness* of our taxonomy, have been obtained through one large industrial case study, involving a domain expert; this is a threat to the generalization of the results. We tried to mitigate this threat by selecting a case study with a rich set of requirements extracted from the documentation of a complex, production-grade system. Such requirements are representative, in many ways, of those defined in the satellite and other cyber-physical domains. Nevertheless, some CPS domains (e.g., healthcare) may have specific types of requirements (e.g., supporting frequency-domain in the temporal specifications), which could lead to different results.

6. Applications

In this section, we discuss how the main contributions of the papers can support the research community and practitioners working in the CPS domain.

Table 7
Coverage of property types (from our taxonomy, see Fig. 1 for acronyms) in example specifications from the literature.

Reference	Formalism	DA	SPK	RT (FT)	OSH (USH)	OSC	RSH
Maler and Nickovic (2004)	STL	+	−	−	−	−	+
Maler and Ničković (2013)	STL/PSL	+	−	+	−	−	+
Kapinski et al. (2016b)	PSTL	+	+	+	+	−	+
Brim et al. (2014)	STL*	+	−	−	−	+	−
Annapureddy and Fainekos (2010)	MTL	+	−	−	−	−	+
Abbas et al. (2013)	MTL	+	−	−	−	−	+
Abbas et al. (2017)	CTMTL	+	−	−	−	−	−
Abbas et al. (2017)	XCTL	+	−	−	−	−	+
Abbas et al. (2017)	CLTL	+	−	−	−	−	+
Bartocci et al. (2013b)	STL	+	+	−	−	−	+
Akazaki and Hasuo (2015)	STL	+	−	−	−	−	+
Akazaki and Hasuo (2015)	AVSTL	+	−	−	−	−	+
Bartocci et al. (2013a)	STL	+	−	−	−	−	+
Bartocci et al. (2015)	STL	+	+	−	−	+	+
Bartocci et al. (2009)	KSL	+	−	−	−	−	+
Bortolussi et al. (2015)	MITL	+	−	−	−	−	−
Bufo et al. (2014)	MITL	+	−	−	−	−	−
Deshmukh et al. (2017)	STL	+	−	−	−	+	+
Deshmukh et al. (2015)	STL	+	−	−	−	+	+
Dokhanchi et al. (2015b)	MTL	+	−	−	−	−	−
Donzé and Maler (2010)	STL	+	−	−	−	−	+
Dreossi et al. (2015)	STL	+	−	−	−	−	−
Ferrere (2016)	TRE	+	+	−	−	−	+
Hoxha et al. (2015)	STL	+	−	−	−	−	+
Jakšić et al. (2016)	STL	+	−	−	−	−	+
Juniwal et al. (2014)	STL	+	−	−	−	−	+
Juniwal et al. (2014)	PSTL	+	−	−	−	−	−
Cameron et al. (2015)	MTL	+	−	−	−	−	+
Nghiem et al. (2010)	MTL	+	−	−	−	−	+
Dokhanchi et al. (2014)	MTL	+	−	−	−	−	+
Dokhanchi et al. (2015a)	MITL	+	−	−	−	+	+
Dokhanchi et al. (2015a)	STL	+	−	−	−	−	+
Nguyen and Nikovi (2016)	STL	+	+	−	−	−	+
Jin et al. (2015)	STL	+	+	−	−	−	+
Jin et al. (2015)	PSTL	+	+	−	+	−	+
Donzé (2010)	MITL	+	−	−	−	−	+
Donzé et al. (2011)	STL	+	−	−	−	+	+
Eisner and Fisman (2007)	PSL	+	−	−	−	−	+
Fainekos and Pappas (2006)	MTL	+	−	−	−	−	+
Fainekos and Pappas (2006)	MITL	+	−	−	−	−	−
Fainekos and Pappas (2006)	MTL	+	−	−	−	−	+
Fainekos et al. (2012)	MTL	+	−	−	−	−	+
Ferrere et al. (2015)	TRE	+	+	−	−	−	−
Hoxha et al. (2014)	PMTL	+	−	−	−	−	−
Hoxha et al. (2014)	MTL	+	−	−	−	−	+
Hoxha et al. (2018)	MTL	+	−	−	−	−	+
Hoxha et al. (2018)	PMTL	+	−	−	−	−	+
Jakšić et al. (2015)	STL	+	−	−	−	+	+
Kane (2015)	BMTL	+	−	−	−	−	+
Maler et al. (2008)	STL	+	−	−	−	−	+
Nickovic (2008)	MITL	+	−	−	−	−	+
Nickovic (2008)	STL	+	−	−	−	−	+
Nickovic (2008)	STL/PSL	+	−	+	−	−	+
Nickovic (2008)	MTL-B	+	−	−	−	−	+
Nickovic and Maler (2007)	STL/PSL	+	−	−	−	−	+
Pajic et al. (2014)	CTL	+	−	−	−	−	+
Rizk et al. (2008)	LTL(R)	+	−	−	−	+	−
Rizk et al. (2008)	QFLTL(R)	+	−	−	−	+	−
Sankaranarayanan and Fainekos (2012)	MTL	+	−	−	−	−	+
Selyunin et al. (2017)	STL	+	+	+	−	−	−
Selyunin et al. (2017)	TRE	+	+	+	−	−	−
Stoma et al. (2013)	STL	+	−	−	−	−	−
Ulus et al. (2014)	TRE	+	−	−	−	+	+
Yang et al. (2012)	PMTL	+	−	−	−	−	−
Total		64	10	5	2	10	48

Application of the taxonomy. The taxonomy of signal-based temporal properties can be used by researchers to *design new specification languages*, whose constructs can be directly mapped to the main property types identified in the taxonomy. This type of impact has been already observed for similar contributions in the literature, such as the seminal work of Dwyer et al. (1999) on temporal specification patterns, which has influenced the design of many domain-specific languages for temporal specifications (e.g., Temporal OCL (Kanso and Taha, 2013), OCLR (Dou et al., 2014), VISPEC - graphical formalism (Hoxha et al., 2015), TemPsy (Dou et al., 2017), ProMoboBox - property language (Meyers et al., 2020), FRETISH (Giannakopoulou et al., 2020)), and the work on service provisioning patterns (Bianculli et al., 2012), which has led to the design of new specification languages and tools (Bianculli et al., 2013; Bersani et al., 2014; Bianculli et al., 2014a,b; Boufaied et al., 2019). For instance, as mentioned in Section 4, some of the authors have already developed *SB-TemPsy-DSL* (Boufaied et al., 2020), a domain-specific specification language for signal-based properties based on the taxonomy proposed in this paper.

The property types included in our taxonomy can also be used to *assess the expressiveness* of existing languages, in a way similar to what we have done in Section 4. By doing so, researchers can identify expressiveness gaps in existing languages, which could then be extended to support specific constructs. For instance, the motivating example for the development of *STL** (Brim et al., 2014) was the impossibility of expressing oscillatory behaviors in *STL*.

Furthermore, practitioners can use the taxonomy as a reference guide to systematically *identify and characterize signal behaviors*, so that the latter can be defined precisely and used correctly during the development process of CPSs (e.g., when defining system requirements or test oracles).

Application of the logic-based characterization. Researchers can leverage the logic-based characterization of the property types included in our taxonomy to *define the formal semantics* of the constructs of a new language, which has been inspired by the taxonomy itself. In this sense, the logic-based characterization can *guide the implementation* of the core, pattern-specific algorithms of a verification tool, which can be used for checking properties expressed in a language containing constructs derived from the property types included in our taxonomy.

For instance, the formal semantics of the aforementioned *SB-TemPsy-DSL* language and the corresponding trace checking algorithm implemented in *SBTemPsy-Check* (Boufaied et al., 2020) have been developed based on the logic-based characterization introduced in this paper.

Expressiveness results. The expressiveness results of state-of-the-art temporal logics with respect to the property types included in our taxonomy, presented in Section 4, can be used by practitioners to carefully *select the language to use* for defining signal-based properties, based on the type of requirements they are going to define, the expressiveness of the candidate specification language(s), and the availability of suitable tools.

7. Related work

To the best of our knowledge, this is the first paper that presents a comprehensive taxonomy of signal-based temporal properties describing signal behaviors in the CPS domain. The closest work is the taxonomy of automotive controller behaviors presented in Kapinski et al. (2016b), in which behaviors are captured in ST-Lib, a catalog of formal requirements written in *STL*. Although the ST-Lib catalog contains several types of signal-based temporal properties (e.g., spike, overshoot, rise time), the treatment of some property types is limited (e.g., oscillatory behaviors are only discussed for the case of short-period behaviors, i.e., ringing). Furthermore, as we have shown in Section 3.2, the formalization of spike properties proposed in Kapinski et al. (2016b) has some limitations. A specific type of signal-based temporal properties (i.e., oscillations) is discussed in Brim et al. (2014) and used as a motivation for introducing *STL**.

Similarly to what we did in Section 3, most of the papers dealing with the specification or verification of signal-based temporal properties also include examples of such properties written using a specific temporal logic. We systematically reviewed the example properties used throughout all the papers dealing with specification, verification, and monitoring of CPS, cited in a recent survey on these topics (Bartocci et al., 2018); we excluded papers using spatio-temporal and frequency domain properties since they are out of the scope of this work. Table 7 shows, for each of the reviewed papers, the property types (from our taxonomy) to which the examples included in the paper correspond, as well as the temporal logic used for their specification; treatment or lack thereof of a property type is denoted by a "+" or "−" symbol, respectively. One can see that data assertion and relationship between signals are the most common property types covered in the literature, whereas transient behaviors (e.g., rise time, overshoot) properties are the least common; spike and oscillation properties have a similar coverage.

To summarize, we propose in this paper the first comprehensive taxonomy of signal-based properties, formalized in a consistent and precise manner, which accounts for all reported property types in the literature.

8. Conclusion and future work

Requirements of cyber-physical systems are usually expressed using signal-based temporal properties, which characterize the expected behaviors of input and output signals processed by sensors and actuators. Expressing such requirements is challenging because of the many ways to characterize a signal behavior (e.g., using certain features). To avoid ambiguous or inconsistent specifications, we argue that engineers need precise definitions of such features and proper guidelines for selecting the features most appropriate in a certain context. Furthermore, given the broad variation in expressiveness of the specification languages used for defining signal-based temporal properties, our experience indicates that engineers need guidance for selecting the most appropriate specification language, based on the type of requirements they are going to define and the expressiveness of each language.

To tackle these challenges, in this paper we have presented a taxonomy of the most common types of signal-based temporal properties, accompanied by a comprehensive and detailed description of signal-based behaviors and their precise characterization in terms of a temporal logic (*SFO*). Engineers can rely on such characterization to derive — from informal requirements specifications — formal specifications to be used in various V&V activities.

Furthermore, we have reviewed the expressiveness of state-of-the-art signal-based temporal logics (i.e., *STL*, *STL**, *SFO*) in terms of the property types identified in the taxonomy, while also taking into account the complexity of monitoring algorithms and the availability of the corresponding tools. Our analysis indicates that *SFO is the most expressive language for the property types of our taxonomy*; however, the application of *SFO* in V&V activities is still challenging given the computational complexity of the corresponding monitoring algorithm and the lack of tools.

We have also applied our taxonomy to classify the requirement specifications of an industrial case study in the aerospace domain. The case study has shown the feasibility of expressing requirements specifications of a real-world CPS using the property types included in our taxonomy, and has provided evidence of the completeness of our taxonomy.

As part of future work, we plan to assess the expressiveness of other temporal logics (such as SCL — Signal Convolution Logic (Silvetti et al., 2018), the extension of STL proposed in (Bakhirkin and Basset, 2019), and the *shape expressions* formalism (Ničković et al., 2019)) in terms of the property types identified in our taxonomy. Moreover, we plan to collect feedback from practitioners (i.e., software and system engineers) to assess the usefulness of our taxonomy and of the proposed property formalizations for the verification of CPS.

CRediT authorship contribution statement

Chaima Boufaied: Conceptualization, Investigation, Writing - original draft. **Maris Jukss:** Conceptualization, Investigation, Writing - original draft. **Domenico Bianculli:** Conceptualization, Supervision, Writing - review & editing, Funding acquisition. **Lionel Claude Briand:** Supervision, Writing - review & editing, Funding acquisition. **Yago Isasi Parache:** Resources, Validation, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant agreement No 694277), from the University of Luxembourg (grant "MOVIDA"), and from the NSERC Discovery and Canada Research Chair programmes. We also wish to thank Claudio Menghi and Dejan Ničković for their feedback on the paper.

References

Abbas, H., Fainekos, G., Sankaranarayanan, S., Ivančić, F., Gupta, A., 2013. Probabilistic temporal logic falsification of cyber-physical systems. ACM Trans. Embedded Comput. Syst. (TECS) 12 (2s), 95.

Abbas, H., Rodionova, A., Bartocci, E., Smolka, S.A., Grosu, R., 2017. Quantitative regular expressions for arrhythmia detection algorithms. In: Proc. International Conference on Computational Methods in Systems Biology (CMSB2017). Springer, pp. 23–39.

Acır, N., 2005. Automated system for detection of epileptiform patterns in EEG by using a modified RBFN classifier. Expert Syst. Appl. 29 (2), 455–462.

Acır, N., Güzeliş, C., 2004. Automatic spike detection in EEG by a two-stage procedure based on support vector machines. Comput. Biol. Med. 34 (7), 561–575.

Acir, N., Oztura, I., Kuntalp, M., Baklan, B., Guzelis, C., 2005. Automatic detection of epileptiform events in EEG by a three-stage procedure based on artificial neural networks. IEEE Trans. Biomed. Eng. 52 (1), 30–40.

Adam, A., Mokhtar, N., Mubin, M., Ibrahim, Z., Tumari, M.Z.M., Shapiai, M.I., 2014. Feature selection and classifier parameter estimation for EEG signal peak detection using gravitational search algorithm. In: Proc. 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology (AIFU2014). pp. 103–108.

Akazaki, T., Hasuo, I., 2015. Time robustness in MTL and expressivity in hybrid system falsification. In: Proc. International Conference on Computer Aided Verification (CAV2015). Springer, pp. 356–374.

Annapureddy, Y.S.R., Fainekos, G.E., 2010. Ant colonies for Temporal Logic falsification of hybrid systems. In: Proc. 36th Annual Conference on IEEE Industrial Electronics Society (IECON2010). pp. 91–96.

Asarin, E., Caspi, P., Maler, O., 2002. Timed regular expressions. J. ACM 49 (2), 172–206.

Bakhirkin, A., Basset, N., 2019. Specification and efficient monitoring beyond STL. In: Proc. International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS2019). Springer, pp. 79–97.

Bakhirkin, A., Ferrère, T., Henzinger, T.A., Ničković, D., 2018. The first-order logic of signals: Keynote. In: Proc. International Conference on Embedded Software (EMSOFT2018). In: EMSOFT '18, IEEE Press, pp. 1:1–1:10.

Bartocci, E., Bortolussi, L., Nenzi, L., 2013a. A temporal logic approach to modular design of synthetic biological circuits. In: Proc. International Conference on Computational Methods in Systems Biology (CMSB2013). Springer, pp. 164–177.

Bartocci, E., Bortolussi, L., Nenzi, L., Sanguinetti, G., 2015. System design of stochastic models using robustness of temporal properties. Theoret. Comput. Sci. 587, 3–25.

Bartocci, E., Corradini, F., Merelli, E., Tesei, L., 2009. Model checking biological oscillators. Electron. Notes Theor. Comput. Sci. 229 (1), 41–58.

Bartocci, E., Deshmukh, J., Donzé, A., Fainekos, G., Maler, O., Ničković, D., Sankaranarayanan, S., 2018. Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In: Lectures on Runtime Verification. Springer, pp. 135–175.

Bartocci, E., Grosu, R., Karmarkar, A., Smolka, S.A., Stoller, S.D., Zadok, E., Seyster, J., 2013b. Adaptive runtime verification. In: Proc. International Conference on Runtime Verification (RV2013). Springer Berlin Heidelberg, pp. 168–182.

Bersani, M.M., Bianculli, D., Ghezzi, C., Krstić, S., San Pietro, P., 2014. SMT-based checking of SOLOIST over sparse traces. In: Proc. of FASE 2014. In: LNCS, vol. 8411, Springer, pp. 276–290.

Bianculli, D., Ghezzi, C., Krstić, S., 2014a. Trace checking of metric temporal logic with aggregating modalities using MapReduce. In: Proc. of SEFM 2014. In: LNCS, vol. 8702, Springer, pp. 144–158.

Bianculli, D., Ghezzi, C., Krstić, S., San Pietro, P., 2014b. Offline trace checking of quantitative properties of service-based applications. In: Proceedings of the 7h International Conference on Service Oriented Computing and Application (SOCA 2014). IEEE, pp. 9–16. http://dx.doi.org/10.1109/SOCA.2014.14.

Bianculli, D., Ghezzi, C., Pautasso, C., Senti, P., 2012. Specification patterns from research to industry: a case study in service-based applications. In: Proc. ICSE2012. IEEE, Los Alamitos, CA, USA, pp. 968–976.

Bianculli, D., Ghezzi, C., San Pietro, P., 2013. The tale of SOLOIST: a specification language for service compositions interactions. In: Proc. FACS'12. In: LNCS, vol. 7684, Springer, Heidelberg, Germany, pp. 55–72.

Bortolussi, L., Milios, D., Sanguinetti, G., 2015. U-check: Model checking and parameter synthesis under uncertainty. In: Proc. Quantitative Evaluation of Systems (QEST2015). Springer International Publishing, pp. 89–104.

Boufaied, C., Bianculli, D., Briand, L.C., 2019. A model-driven approach to trace checking of temporal properties with aggregations. J. Objectg Technol. 18 (2), 15:1–15:21. http://dx.doi.org/10.5381/jot.2019.18.2.a15.

Boufaied, C., Menghi, C., Bianculli, D., Briand, L., Isasi-Parache, Y., 2020. Trace-checking signal-based temporal properties: A model-driven approach. In: Proc. International Conference on Automated Software Engineering (ASE2020). IEEE.

Brim, L., Dluhoš, P., Šafránek, D., Vejpustek, T., 2014. STL*: Extending signal temporal logic with signal-value freezing operator. Inform. and Comput. 236, 52–67.

Brim, L., Vejpustek, T., Šafránek, D., Fabriková, J., 2013. Robustness analysis for value-freezing signal temporal logic. In: Proc. Second International Workshop on Hybrid Systems and Biology (HSB2013). In: Electronic Proceedings in Theoretical Computer Science, vol. 125, Open Publishing Association, pp. 20–36.

Bufo, S., Bartocci, E., Sanguinetti, G., Borelli, M., Lucangelo, U., Bortolussi, L., 2014. Temporal logic based monitoring of assisted ventilation in intensive care patients. In: Proc. International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA2014). Springer Berlin Heidelberg, pp. 391–403.

Cameron, F., Fainekos, G., Maahs, D.M., Sankaranarayanan, S., 2015. Towards a verified artificial pancreas: Challenges and solutions for runtime verification. In: Proc. International Conference on Runtime Verification (RV2015). Springer, pp. 3–17.

Chechik, M., Paun, D.O., 1999. Events in property patterns. In: Proc. 5th and 6th International SPIN Workshops on Theoretical and Practical Aspects of SPIN Model Checking (SPIN1999). Springer-Verlag, pp. 154–167.

Deshmukh, J.V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., Seshia, S.A., 2015. Robust online monitoring of signal temporal logic. In: Proc. International Conference on Runtime Verification (RV2015). Springer International Publishing, pp. 55–70.

Deshmukh, J.V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., Seshia, S.A., 2017. Robust online monitoring of signal temporal logic. Form. Methods Syst. Des. 51 (1), 5–30.

Dingle, A.A., Jones, R.D., Carroll, G.J., Fright, W.R., 1993. A multistage system to detect epileptiform activity in the EEG. IEEE Trans. Biomed. Eng. 40 (12), 1260–1268.

Dokhanchi, A., Hoxha, B., Fainekos, G., 2014. On-line monitoring for temporal logic robustness. In: Proc. International Conference on Runtime Verification (RV2014). Springer, pp. 231–246.

Dokhanchi, A., Hoxha, B., Fainekos, G., 2015a. Metric interval temporal logic specification elicitation and debugging. In: Proc. International Conference on Formal Methods and Models for Codesign (MEMOCODE2015). IEEE, pp. 70–79.

Dokhanchi, A., Zutshi, A., Sriniva, R.T., Sankaranarayanan, S., Fainekos, G., 2015b. Requirements driven falsification with coverage metrics. In: Proc. International Conference on Embedded Software (EMSOFT2015). pp. 31–40.

Donzé, A., 2010. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: Proc. International Conference on Computer Aided Verification (CAV2010). Springer, pp. 167–170.

Donzé, A., Fanchon, E., Gattepaille, L.M., Maler, O., Tracqui, P., 2011. Robustness analysis and behavior discrimination in enzymatic reaction networks. PLoS One 6 (9), e24246.

Donzé, A., Maler, O., 2010. Robust satisfaction of temporal logic over real-valued signals. In: Proc. International Conference on Formal Modeling and Analysis of Timed Systems (Formats2010). Springer Berlin Heidelberg, pp. 92–106.

Donzé, A., Maler, O., Bartocci, E., Nickovic, D., Grosu, R., Smolka, S., 2012. On temporal logic and signal processing. In: Proc. International Symposium on Automated Technology for Verification and Analysis (ATVA2012). Springer, pp. 92–106.

Dou, W., Bianculli, D., Briand, L., 2014. OCLR: a more expressive, pattern-based temporal extension of OCL. In: Proc. ECMFA 2014. In: LNCS, vol. 8569, Springer, Heidelberg, Germany, pp. 51–66.

Dou, W., Bianculli, D., Briand, L., 2017. A model-driven approach to trace checking of pattern-based temporal properties. In: Proc. MODELS2017. IEEE Computer Society, Los Alamitos, CA, USA, pp. 323–333.

Dreossi, T., Dang, T., Donzé, A., Kapinski, J., Jin, X., Deshmukh, J.V., 2015. Efficient guiding strategies for testing of temporal properties of hybrid systems. In: Proc. NASA Formal Methods (NFM2015). Springer International Publishing, pp. 127–142.

DSI consortium, DSI consortium, 2011. DSI3 bus standard.

Dumpala, S.R., Reddy, S.N., Sarna, S.K., 1982. An algorithm for the detection of peaks in biological signals. Comput. Programs Biomed. 14 (3), 249–256.

Dwyer, M.B., Avrunin, G.S., Corbett, J.C., 1999. Patterns in property specifications for finite-state verification. In: Proc. 21st International Conference on Software Engineering (ICSE1999). ACM, pp. 411–420.

Eisner, C., Fisman, D., 2007. A Practical Introduction to PSL. Springer Science & Business Media.

Fainekos, G.E., Pappas, G.J., 2006. Robustness of temporal logic specifications. In: Formal Approaches to Software Testing and Runtime Verification. Springer, pp. 178–192.

Fainekos, G.E., Sankaranarayanan, S., Ueda, K., Yazarel, H., 2012. Verification of automotive control applications using s-taliro. In: Proc. American Control Conference (ACC2012). Citeseer, pp. 3567–3572.

Ferrere, T., 2016. Assertions and Measurements for Mixed-Signal Simulation (Ph.D. thesis). University of Grenoble.

Ferrere, T., Maler, O., Ničković, D., Ulus, D., 2015. Measuring with timed patterns. In: Proc. International Conference on Computer Aided Verification (CAV2015). Springer, pp. 322–337.

Giannakopoulou, D., Pressburger, T., Mavridou, A., Schumann, J., 2020. Generation of formal requirements from structured natural language. In: Requirements Engineering: Foundation for Software Quality (REFSQ 2020). Springer International Publishing, Cham, pp. 19–35.

Gonzalez Perez, C.A., Varmazyar, M., Nejati, S., Briand, L., et al., 2018. Enabling model testing of cyber-physical systems. In: Proc. 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS2018). pp. 176–186.

Hägglund, T., 1995. A control-loop performance monitor. Control Eng. Pract. 3 (11), 1543–1551.

Hoxha, B., Bach, H., Abbas, H., Dokhanchi, A., Kobayashi, Y., Fainekos, G., 2014. Towards formal specification visualization for testing and monitoring of cyber-physical systems. In: Proc. Int. Workshop on Design and Implementation of Formal Tools and Systems (DIFTS2014). pp. 1–9.

Hoxha, B., Dokhanchi, A., Fainekos, G., 2018. Mining parametric temporal logic properties in model-based design for cyber-physical systems. Int. J. Softw. Tools Technol. Transfer 20 (1), 79–93.

Hoxha, B., Mavridis, N., Fainekos, G., 2015. VISPEC: A graphical tool for elicitation of MTL requirements. In: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2015). pp. 3486–3492.

Jakšić, S., Bartocci, E., Grosu, R., Kloibhofer, R., Nguyen, T., Ničković, D., 2015. From signal temporal logic to FPGA monitors. In: Proc. Formal Methods and Models for Codesign (MEMOCODE2015). IEEE, pp. 218–227.

Jakšić, S., Bartocci, E., Grosu, R., Ničković, D., 2016. Quantitative monitoring of STL with edit distance. In: Proc. International Conference on Runtime Verification (RV2016). Springer International Publishing, pp. 201–218.

Jin, X., Donzé, A., Deshmukh, J.V., Seshia, S.A., 2015. Mining requirements from closed-loop control models. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 34 (11), 1704–1717.

Juniwal, G., Donzé, A., Jensen, J.C., Seshia, S.A., 2014. CPSGrader: Synthesizing temporal logic testers for auto-grading an embedded systems laboratory. In: Proc. International Conference on Embedded Software (EMSOFT2014). pp. 1–10.

Kane, A., 2015. Runtime Monitoring for Safety-Critical Embedded Systems (Ph.D. thesis). Carnegie Mellon University.

Kanso, B., Taha, S., 2013. Temporal constraint support for OCL. In: Proc. SLE 2012. In: LNCS, vol. 7745, Springer, Berlin, Heidelberg, pp. 83–103.

Kapinski, J., Deshmukh, J., Jin, X., Ito, H., Butts, K., 2016a. Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques. IEEE Control Syst. Mag. 36, 45–64.

Kapinski, J., Jin, X., Deshmukh, J., Donze, A., Yamaguchi, T., Ito, H., Kaga, T., Kobuna, S., Seshia, S., 2016b. ST-Lib: A Library for Specifying and Classifying Model Behaviors. Technical Report, SAE Technical Paper.

Konrad, S., Cheng, B.H.C., 2005. Real-time specification patterns. In: Proc. 27th International Conference on Software Engineering (ICSE2005). ACM, pp. 372–381.

Lee, E.A., Seshia, S.A., 2016. Introduction to Embedded Systems: A Cyber-Physical Systems Approach, second ed. The MIT Press.

Liu, H.S., Zhang, T., Yang, F.S., 2002. A multistage, multimethod approach for automatic detection and classification of epileptiform EEG. IEEE Trans. Biomed. Eng. 49 (12), 1557–1566.

Maler, O., Nickovic, D., 2004. Monitoring temporal properties of continuous signals. In: Proc. FTRTFT2004. Springer, pp. 152–166.

Maler, O., Ničković, D., 2013. Monitoring properties of analog and mixed-signal circuits. Int. J. Softw. Tools Technol. Transfer 15 (3), 247–268.

Maler, O., Nickovic, D., Pnueli, A., 2008. Checking temporal properties of discrete, timed and continuous behaviors. In: Pillars of Computer Science2008. Springer, pp. 475–505.

Matinnejad, R., Nejati, S., Briand, L., Bruckmann, T., 2018. Test generation and test prioritization for Simulink models with dynamic behavior. IEEE Trans. Softw. Eng. 45 (9), 919–944.

Meyers, B., Vangheluwe, H., Denil, J., Salay, R., 2020. A framework for temporal verification support in domain-specific modelling. IEEE Trans. Softw. Eng. 46 (4), 362–404.

Nghiem, T., Sankaranarayanan, S., Fainekos, G., Ivancić, F., Gupta, A., Pappas, G.J., 2010. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In: Proc. 13th ACM International Conference on Hybrid Systems: Computation and Control (HSCC2010). In: HSCC '10, ACM, pp. 211–220.

Nguyen, L.V., Kapinski, J., Jin, X., Deshmukh, J.V., Butts, K., Johnson, T.T., 2017. Abnormal data classification using time-frequency temporal logic. In: Proc. 20th International Conference on Hybrid Systems: Computation and Control (HSCC2017). ACM, pp. 237–242.

Nguyen, T., Nikovi, D., 2016. Assertion-based monitoring in practice checking correctness of an automotive sensor interface. Sci. Comput. Program. 118 (C), 40–59.

Nickovic, D., 2008. Checking Timed and Hybrid Properties: Theory and Applications (Ph.D. thesis). Université Joseph-Fourier-Grenoble I.

Ničković, D., 2015. Monitoring and measuring hybrid behaviors. In: Proc. International Conference on Runtime Verification (RV2015). Springer International Publishing, pp. 378–402.

Ničković, D., Lebeltel, O., Maler, O., Ferrère, T., Ulus, D., 2018. AMT 2.0: Qualitative and quantitative trace analysis with extended signal temporal logic. In: Proc. International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS2018). Springer, pp. 303–319.

Nickovic, D., Maler, O., 2007. AMT: A property-based monitoring tool for analog systems. In: Proc. International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS2007). Springer Berlin Heidelberg, pp. 304–319.

Ničković, D., Qin, X., Ferrère, T., Mateis, C., Deshmukh, J., 2019. Shape expressions for specifying and extracting signal features. In: Proc. International Conference on Runtime Verification (RV2019). Springer, pp. 292–309.

Ničković, D., Yamaguchi, T., 2020. RTAMT: Online robustness monitors from STL. In: Proc. International Symposium on Automated Technology for Verification and Analysis (ATVA 2020). Springer.

Pajic, M., Mangharam, R., Sokolsky, O., Arney, D., Goldman, J., Lee, I., 2014. Model-driven safety analysis of closed-loop medical systems. IEEE Trans. Ind. Inf. 10 (1), 3–16.

Rizk, A., Batt, G., Fages, F., Soliman, S., 2008. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In: Proc. International Conference on Computational Methods in Systems Biology (CMSB2008). Springer, pp. 251–268.

Sankaranarayanan, S., Fainekos, G., 2012. Falsification of temporal properties of hybrid systems using the cross-entropy method. In: Proc. 15th ACM International Conference on Hybrid Systems: Computation and Control (HSCC2012). ACM, pp. 125–134.

Selyunin, K., Jaksic, S., Nguyen, T., Reidl, C., Hafner, U., Bartocci, E., Nickovic, D., Grosu, R., 2017. Runtime monitoring with recovery of the SENT communication protocol. In: Proc. International Conference on Computer Aided Verification (CAV2017). Springer, pp. 336–355.

Silvetti, S., Nenzi, L., Bartocci, E., Bortolussi, L., 2018. Signal convolution logic. In: Proc. International Symposium on Automated Technology for Verification and Analysis (ATVA2018). Springer International Publishing, pp. 267–283.

Stoma, S., Donzé, A., Bertaux, F., Maler, O., Batt, G., 2013. STL-based analysis of TRAIL-induced apoptosis challenges the notion of type i/type II cell line classification. PLoS Comput. Biol. 9 (5), e1003056.

Ulus, D., Ferrère, T., Asarin, E., Maler, O., 2014. Timed pattern matching. In: Proc. International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS2014). Springer International Publishing, pp. 222–236.

Yang, H., Hoxha, B., Fainekos, G., 2012. Querying parametric temporal logic properties on embedded systems. In: Proc. International Conference on Testing Software and Systems (IFIP2012). Springer, pp. 136–151.

Chaima Boufaied is a Ph.D. candidate at the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. She started her Ph.D. in January 2017, after obtaining a M.Sc. in information systems modeling and decision support from ISGT University, Tunisia. Prior to her master studies, she obtained a bachelor degree from ESCT university (Tunisia). Her doctoral work focuses primarily on model-driven trace checking of temporal properties for cyber–physical systems.

Maris Jukss is a software engineer working on cyber–physical systems. Previously, he was a research associate at the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. He holds a Ph.D. degree from the School of Computer Science of McGill University, Canada. He received his B.Sc. in Computer Science in 2003 at the Riga Technical University in Latvia. His research interests include software engineering for cyber–physical systems, run-time model transformation optimizations, model transformation debugging.

Domenico Bianculli is associate professor/chief scientist II at the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. He holds a Ph.D. degree from Università della Svizzera italiana (Switzerland), a M.Sc. in Computing Systems Engineering and a B.Sc. in Computer Engineering, both from Politecnico di Milano (Italy). His research focuses on the specification and verification of evolvable software systems. His research interests include: run-time verification, compliance analysis, modeling and enforcing of access control policies, program analysis for security, incremental verification techniques, and verification of service-oriented systems.

Lionel C. Briand is professor of software engineering and has shared appointments between (1) School of Electrical Engineering and Computer Science, University of Ottawa, Canada and (2) the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. He is the head of the SVV department at the SnT Centre and a Canada Research Chair in Intelligent Software Dependability and Compliance (Tier 1). He holds an ERC Advanced Grant, the most prestigious European individual research award, and has conducted applied research in collaboration with industry for more than 25 years, including projects in the automotive, aerospace, manufacturing, financial, and energy domains. In 2010, he was elevated to the grade of IEEE Fellow for his work on testing of object-oriented systems. He was also granted the IEEE Computer Society Harlan Mills award (2012) and the IEEE Reliability Society Engineer-of-the-year award (2013) for his work on model-based verification and testing. His research interests include: Model-driven development, testing and verification, search-based software engineering, requirements engineering, and empirical software engineering.

Yago Isasi Parache is Head of the Software division at LuxSpace Sàrl, and leads the development of satellite simulators, satellite on-board software, satellite data applications, software product assurance, and software tooling for space software development. He holds a Diploma of Advanced Studies (DEA - Master Degree) in Applied Mathematics from Universitat Politècnica de Catalunya, a Diploma of Advanced Studies (DEA - Master Degree) in Astrophysics, Particle Physics, and Cosmology from Universita de Barcelona, and a degree in Industrial Engineering from Universitat Politècnica de Catalunya.