On Temporal Logic and Signal Processing

Alexandre Donzé[1], Oded Maler[2] , Ezio Bartocci[4], Dejan Nickovic[3], Radu Grosu[4] and Scott Smolka[5]

[1] EECS Department, University of California, Berkeley, USA donze@imag.fr
[2] Verimag, Université Joseph Fourier/CNRS, Gières, France
[3] Austrian Institute of Technology, Vienna, Austria
[4] Department of Computer Engineering, Vienna University of Technology, Austria
[5] Department of Computer Science, Stony Brook University, USA

Abstract. We present *Time-Frequency Logic* (TFL), a new specification formalism for real-valued signals that combines temporal logic properties in the time domain with frequency-domain properties. We provide a property checking framework for this formalism and illustrate its expressive power in defining and recognizing properties of musical pieces. Like hybrid automata and their analysis techniques, the TFL formalism is a contribution to a unified systems theory for hybrid systems.

1 Introduction

The exportation of Temporal Logic (TL) from philosophy [31,32] to systems design [28,29] is considered a turning point in formal verification [33], putting the focus on the *ongoing input-output behavior* of a *reactive* system [14] rather than on the final output of a complex program. While reactive systems might be a concept worth distinguishing in Computer Science, in other domains such as Control, Signal Processing and Circuit Design, being reactive is the rule, not the exception. Such systems are viewed by designers as networks of transducers (block diagrams) communicating continuously via *signals*: functions from Time to some domain, such as the Reals. This is the world view underlying data-flow languages and frameworks such as Lustre [2], Signal [1], Simulink[6] and Ptolemy [27], as well as transistor-level circuit simulators.

TL provides a convenient framework for writing in a compact and formal way specifications that the system under design should satisfy. It was initially intended to evaluate clean and well-defined *sequences* of states and events as found in digital systems. In the last couple of years, it has been extended to the specification of properties of *real-valued* signals defined over *dense time* [19,21] and applied to diverse domains ranging from analog circuits [17] to biochemical reactions [6]. The logic STL (*signal temporal logic*) allows designers to speak of properties related to the *order* of discrete events and the *temporal distance* between them, where "events" correspond to changes in the satisfaction of some predicate (e.g., threshold crossing) over the real variables. Traditional performance measures used to evaluate signals are more continuous and "event-free",

[6] http://www.mathworks.com/tagteam/43815_9320v06_Simulink7_v7.pdf

for example, averaged/discounted integrals of some variables over time, and the added expressivity is an important contribution to the emergence of a hybrid *Systems Theory*.

One of the technical and cultural impediments to the adoption of STL, especially in analog circuits (its original motivating domain), was its purely *time-domain* nature; i.e., it did not lend itself to *frequency-domain* analysis. This kind of analysis is based on the Fourier spectrum of the signal which in many engineering applications is more important than the properties of the signal itself; i.e., its properties in the time domain, which STL is intended to express. One reason for this bias is that real-life analog signals are accompanied by omnipresent *noise*, i.e., random perturbations of the desired signal. This is conveniently dealt with in the frequency domain via filtering. Typically, the noise component of a signal populates a range of frequencies different from the range of the signal of interest. If we keep only the latter, then the amplitude of the noise is strongly reduced. More generally, an analog signal is usually a composition of multiple sources, and the first purpose of signal processing is the *separation* of these sources. In the frequency domain, this is done by simple operations such as thresholding or filtering the range of frequencies of interest, assuming that each source has a range clearly distinct from the ranges of others.

Source separation, noise removal and signal filtering are fundamental operations which are *time-invariant*. They affect the signal in the same way from time $t = -\infty$ to $t = \infty$. Since the Fourier transform (FT) is defined over this unbounded time-interval, it is appropriate for these operations. However, when it comes to characterize bounded or local-time intervals, the FT becomes cumbersome in practice, as its definition aggregates for each frequency of interest all values along the duration of the signal. This observation naturally led to the search for proper *time-frequency* analysis techniques, beginning from straightforward extensions of FT (such as the Short Time Fourier Transform (STFT) [10]) and culminating with the sophisticated and versatile family of Wavelet transforms [22]. Time-frequency analysis as a branch of Signal Processing have seen contributions of tremendous importance. To name only one, modern compression algorithms at the root of Jpeg, Mpeg, etc, file formats are all based on wavelet theory.

In this paper, we complement this evolution toward a fusion of time-domain and frequency-domain analysis by proposing a unified logical formalism for expressing hybrid (time-frequency) properties of signals. Some preliminary work in this direction is reported in [3] in which the author describes "envelope" predicates over the Fourier coefficients in the context of hybrid systems verification. However they use the standard Fourier transform, thus not treating the tighter coupling of time and frequency domains that we investigate here. Attempts have been done in the past to apply time-frequency analysis as a design methodology to different application domains. In the context of analog circuits, time-frequency analysis was used to study dynamic current supplies, oscillators, leapfrog and state variable filters etc. In the bio-medical domain time-frequency analysis was applied to detect anomalous ECG signals. We observed that the common dis-

advantage of these techniques is the lack of a formalism to express complex temporal patterns of frequency responses and a suitable machinery to automatically detect them. Our formalism, which we call *Time-Frequency Logic* (TFL), is implemented in a generic monitoring framework, part of the tool Breach [5].

A somewhat related work is *Functional reactive programming* (FRP) [16], a paradigm for high-level declarative programming of hybrid systems, instantiated with the Yampa language which was used to program industrial-strength mobile robots [26] but also to synthesize music [11]. Although it shares similar concepts with our work (signal transformers), FRP as a *programming* paradigm remains complementary to TFL, which is a *specification* formalism supporting, in particular, acausality and non-determinism.

The rest of the paper is organized as follows. Section 2 gives a short introduction to STL. Section 3 is a crash course on Fourier analysis.[7] Section 4 explains time-frequency analysis and how it is integrated in the TFL monitoring framework, while Section 5 demonstrates TFL's applicability to Music.

2 Signal Temporal Logic

We present STL in a manner that makes it closer to the world of Control and Signal Processing. Rather than taking the logic as a primary object and put all the numerical predicates and functions as part of domain-specific *signature* (as in [21]), we take a general framework of a data-flow network (as has been done in [4]) and add the temporal operators (*until* and *since*) as a special type of signal transducers. This approach is also close to [24], where AMS-LTL is extended with *auxiliary functions* that allow to embed arbitrary signal transducers in the body of property specifications.

This style of presentation is facilitated by a somewhat non-orthodox way of defining the semantics of temporal logic using *temporal testers* [18,30,23]. This approach has already been applied to STL monitoring [19,21] and to translating the MITL logic into timed automata [20]. An STL formula φ is viewed as a network of signal operators (transducers), starting with the raw signals x (sequences of atomic propositions in the discrete case), and culminating in a top-level signal φ,[8] whose value at t represents the satisfaction of the top-level formula at time t: $\varphi[t] = 1$ iff $(x, t) \models \varphi$. Each sub-formula of the form $\varphi = f(\varphi_1, \varphi_2)$ is thus associated with a signal transducer realizing f, whose inputs are the satisfaction signals of φ_1 and φ_2. The whole apparatus for monitoring the satisfaction of a formula by a signal can thus be viewed as a network of operators working on signals of two major types: *numerical* (raw signals and those obtained by numerical operations on them) and *Boolean* (satisfaction signals of sub-formulae). We assume signals defined as functions from Time $\mathbb{T}$ to some domain D. The

[7] We recommend [15] as a first reading for computer scientists.

[8] We make a compromise between the conventions of Logic and those of other less formal domains by writing abusively φ for both the formula and its satisfaction signal and will do the same for variables x and their associated raw signals.

range $\mathbb{T}$ of the signal can be finite $[0, r]$, infinite $[0, \infty]$ or bi-infinite $[-\infty, \infty]$ and we will make distinctions only when needed.

Definition 1 (Signal Operator). *A* signal operator *is a mapping* $f : (\mathbb{T} \to D_1) \to (\mathbb{T} \to D_2)$*, where* D_1 *and* D_2 *are, respectively, the domains of the input and output signals.*

The domains D_1 and D_2 define the *type* of the operators, and we always assume that the arguments of these operators match the type. We assume that all operators are (approximately) computable so that given some representation of a signal x, it is possible to produce a representation $y = f(x)$.

Definition 2 (Operator Classification). *Let* f *be a signal operator and let* $y = f(x)$*. We say that* f *is*

- Pointwise (memoryless) *if it is a lifting to signals of a function* $f : D_1 \to D_2$*, that is,* $\forall t\ y[t] = f(x[t])$*;*
- Causal *if, for every* t*,* $y[t]$ *is computed based on at most* $x[0], \ldots, x[t]$*;*
- Acausal *(otherwise)*

The causal operators are, for example, *past* temporal operators, back-shifts, or integrals over temporal windows that extend backwards. The advantage of such operators is that they can be naturally monitored *online*, which is particularly important for monitoring *real* systems rather than simulated *models*. The acausal operators are the future temporal operators and other operators that depend on values in temporal windows that extend beyond the current point in time.

Definition 3 (Temporal Operators). *Let* φ_1 *and* φ_2 *be two Boolean signals and let* a, b *be two positive numbers satisfying* $a \leq b$*. Then* $\psi_1 = \varphi_1\ \mathcal{U}_{[a,b]}\varphi_2$ *(*φ_1 until φ_2*) are the signals satisfying*[9]

$$\begin{array}{l}(x,t) \models \varphi_1\ \mathcal{U}_{[a,b]}\varphi_2 \text{ if} \\ \quad \exists t' \in t \oplus [a,b]\ (x,t') \models \varphi_2\ \wedge\ \forall t'' \in [t,t']\ (x,t'') \models \varphi_1[t'])\end{array} \tag{1}$$

Recalling that over the Booleans, $\wedge$ and $\vee$ coincide with min and max, (1) is equivalent to:

$$\psi_1[t] = \max_{t' \in t \oplus [a,b]} \min(\varphi_2[t'],\ \min_{t'' \in [t,t']} \varphi_1[t'']) \tag{2}$$

The derived operator $\Diamond_{[a,b]}\varphi = \mathit{true}\ \mathcal{U}_{[a,b]}\varphi$ (*eventually* φ) is true at t when $\varphi[t']$ holds in *some* $t' \in t \oplus [a, b]$, while $\Box_{[a,b]}\varphi = \neg\Diamond_{[a,b]}\neg\varphi$ (*always* φ) requires φ to hold throughout the interval $t \oplus [a, b]$. The untimed *until*, $\mathcal{U} = \mathcal{U}_{[0,\infty]}$ does not pose any metric constraints on the timing of the future occurrence of φ_2.

Let us assume a set $x_1, \ldots, x_m$ of variables and a family F of signal operators including

- Pointwise operators that realize standard arithmetical and logical operators such as $+$, $\cdot$, min, max, $\wedge$, $\neg$, $=$ and $<$;

[9] Expression $t \oplus [a, b]$ denotes the interval $[t + a, t + b]$.

– Other useful operators such as integral, convolution, etc.
– As many instances as needed of $\mathcal{U}_{[a,b]}$.

Definition 4 (STL Syntax). *The syntax of an STL formula is defined inductively as*

– *An atomic formula is any variable x_i or any rational constant c;*
– *If φ is a formula, so is any $f(\varphi)$ for any operator $f \in F$ compatible with the type of φ.*
– *If φ_1 and φ_2 are formulae, so is any $f(\varphi_1, \varphi_2)$ for any operator $f \in F$ compatible with the types of φ_1 and φ_2.*

The semantics of an STL formula φ relative to a raw signal $x = (x_1, \ldots, x_m)$ is immediate; that is, the semantics of x is the signal x and the semantics of a constant is the constant signal c. Then the semantics of $f(\varphi)$ or $f(\varphi_1, \varphi_2)$ is obtained by applying the operator f to the semantics of φ or φ_1 and φ_2.

The work in [19,21] shows that given an STL formula φ and a signal x, there is an algorithm that can check whether x satisfies φ by computing the satisfaction signals of all sub-formulae. The algorithm works on the parse tree of φ, scanning the raw signals and propagating values upwards in the tree as well as backwards and forward in time until the satisfaction of all sub-formulae are computed, including $\varphi[0]$. The tool AMT [25] realizing this algorithm solves various practical problems that we do not discuss here such as the interpretation of STL over signals of finite duration, bridging the gap between ideal mathematical signals defined over $\mathbb{R}$ and actual signals given by a finite sequence of sampled values, combining online and offline monitoring, etc. A quantitative semantics for STL has been presented in [7], which returns as output positive or negative numbers indicating how robustly the property is satisfied or violated. These numbers are propagated naturally using a real-valued version of (2). The Breach tool [5] implements monitoring for this semantics.

3 Frequency Analysis in a Nutshell

The essence of frequency analysis [8] is that a signal can be transformed into an alternative representation consisting of a weighted sum of basic elementary signals, namely, sinusoids of various frequencies and phases. E.g., the signal x of Fig. 1 can be written as $x = x_1 + x_2 + x_3$ with $x_i[t] = b_i \sin 2\pi\omega_i t$, where b_i is the amplitude/coefficient of the sinusoid of frequency ω_i. Thus, the signal is transformed from a time-domain representation $x : \mathbb{T} \to D$ to a function $\hat{x}$ mapping frequencies to their coefficients. Many standard signal-processing operations are best defined as manipulating these coefficients. E.g., if we nullify b_3, we remove the high-frequency component of x to obtain signal $\tilde{x}$, which can be viewed as removing noise from x (Fig. 1).

More formally, on any interval of size $T_0 = 1/\omega_0$, a signal x can be decomposed into a *Fourier series*:

$$x[t] = \frac{a_0}{2} + \sum_{k=0}^{+\infty} a_k \cos(2\pi\omega_k t) + b_k \sin(2\pi\omega_k t) \text{ with } \omega_k = k\omega_0.$$

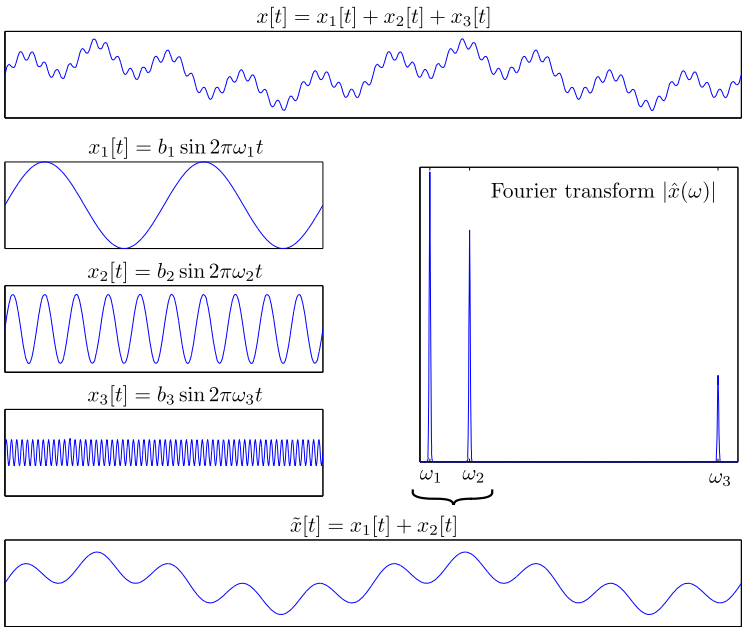

Fig. 1. Fourier transform of a sum of sinusoids and filtering the highest frequency.

This can be written more concisely using Euler's formula $e^{\mathbf{i}x} = \cos(x) + \mathbf{i}\sin(x)$:

$$x[t] = \sum_{k=-\infty}^{+\infty} c_k e^{2\mathbf{i}\pi k\omega_0 t} \tag{3}$$

Coefficients $\{c_k, k \in \mathbb{Z}\}$ provide a discrete spectrum for x on $[0, T_0]$. The *Fourier transform* maps x on the *whole time domain* to a continuous spectrum $\{c_\omega,\ \omega \in \mathbb{R}\}$ containing all real frequencies. The *inverse Fourier transform* (IFT), which recovers x from its spectrum, can be written as:

$$x[t] = \int_{-\infty}^{+\infty} c_\omega e^{2\pi\mathbf{i}\omega t} d\omega, \text{ where } c_\omega = \hat{x}(\omega) = \int_{-\infty}^{+\infty} x[t] e^{-2\mathbf{i}\pi\omega t} dt. \tag{4}$$

which can be seen as a generalization of the Fourier series (3). In practice, the coefficients c_ω are computed for a finite discrete set of frequencies using the FFT algorithm (Fast Fourier Transform [9]), but in the following, we keep the presentation in the continuous domain. This is in the spirit of STL, whose semantics is defined relative to dense-time signals, leaving to the monitoring algorithm the burden of dealing with time-discretization, interpolation, etc.

As mentioned previously, a convenient interpretation of the FT and its inverse is that of a decomposition of x into a sum of sinusoidal components, taking the family of functions $\phi_\omega : t \to e^{2\pi\mathbf{i}\omega t}$ as elementary "blocks" for the decomposition. With this notation, the IFT (4) becomes $x[t] = \int_{-\infty}^{+\infty} c_\omega\ \phi_\omega[t] d\omega$. However, the use of ϕ_ω as an elementary analysis block has the drawback that its definition involves the values of x for all times t. Thus, finding a subset of frequencies or some transformation of $\hat{x}$ that affects a precise time interval for x is not trivial. This motivated the search for other transforms using analysis functions that have a localization both in frequency and in time.

4 Combining Time and Frequency Properties

The Short-Time Fourier Transform (STFT) In the theory of signal processing, the extension of classical frequency analysis to a combined time-frequency analysis is realized by replacing the analysis function $\phi_\omega[t] = e^{2\pi\mathbf{i}\omega t}$ used in the FT with some other analysis function $\phi_{\omega,\tau}$ such that the new transform involves the values of x not only around a specific frequency ω but also around a given time τ (see, e.g., Chapter 4 of [22]). The *short-time* or *windowed* Fourier transform (STFT), first introduced by Gabor [10], uses a straightforward definition of such an analysis function. It consists of the product of ϕ_ω and a *window function* $g[t-\tau]$ whose purpose is simply to filter the values of x outside a neighborhood of τ by forcing them to be 0. It can be as simple as the *rectangular* function of length $L > 0$:

$$g_L[t] = \begin{cases} 1/L & \text{if } t \in [-\frac{L}{2}, \frac{L}{2}], \\ 0 & \text{else.} \end{cases} \tag{5}$$

but other functions with better properties such as the Hanning or Gaussian window functions are usually preferred [22]. They satisfy the normalization property $\int_{-\infty}^{+\infty} g[t]dt = 1$. In the following we assume that g has the form (5) and that the only parameter varying is its length L.

Having chosen a window function g_L, the new analysis function for the STFT is defined as the product $\phi_{\omega,\tau}[t] = \phi_\omega[t]g_L[t-\tau]$ of ϕ_ω and the translation of g_L around τ. Consequently, the STFT of x in (ω,τ), denoted $\hat{x}_L(\omega,\tau)$, defines a *two-dimensional* spectrum $\{c_{\omega,\tau} : (\omega,\tau) \in \mathbb{R}^2\}$. As with the IFT (4), x can be recovered from its spectrum and the analysis function $\phi_{\omega,\tau}$ using the inverse form of the STFT :

$$x[t] = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} c_{\omega,\tau}\phi_{\omega,\tau}[t]d\omega d\tau \tag{6}$$

For a given pair (ω,τ) of frequency and time, the coefficient $c_{\omega,\tau}$, i.e., the STFT of x in (ω,τ), is given by

$$c_{\omega,\tau} = \hat{x}_L(\omega,\tau) = \int_{-\infty}^{+\infty} x[t]g_L(t-\tau)e^{-2\mathbf{i}\pi\omega t}dt. \tag{7}$$

In practice, it can be computed with a straightforward extension of the FFT algorithm, using a sliding window and multiplying x by the g_L window function. The STFT can be visualized as a *spectrogram*, which plots the norms (or more commonly the squared norm) of the coefficients $c_{\omega,\tau}$ as a surface above the time-frequency plane (ω,τ) as illustrated in Fig. 2. There are inherent limitations (Heisenberg Uncertainty Principle) concerning the trade-offs between precision in frequency and precision in time. They are explained in the appendix.

Defining Time-Frequency Predicates The STFT of a signal x thus defines a two dimensional operator taking time and frequency as arguments. By considering the frequency as a parameter, we obtain a family of signal operators $\{f_{L,\omega}\}$ such that $y = f_{L,\omega}(x)$ if $y[t] = \hat{x}_L(\omega,t)$. In other words, $f_{L,\omega}(x)$ is the projection of the L-spectrogram of x on frequency ω. It yields a *spectral signal* which tracks the evolution of the STFT coefficient at ω over time.

Our logic, *time-frequency logic* (TFL) is obtained by adding the operators $\{f_{L,\omega}\}$ to STL. A spectral signal $y = f_{L,\omega}(x)$, like any other signal, can participate in any TFL formula as an argument to predicates and arithmetic expressions. The monitoring machinery is similar to that of STL except for the fact that the raw signals x are pre-processed to yield the spectrogram from which spectral signals y are extracted. This can be done before the monitoring process starts or be integrated in an online procedure as in [21] where segments of y are computed incrementally upon the arrival of segments of x.

On Window Functions and Time-Frequency Resolution To be able to detect the occurrence of a frequency ω at a given time τ, we would need a spectrogram representing a perfect matching between ω and τ for the signal x, i.e.,

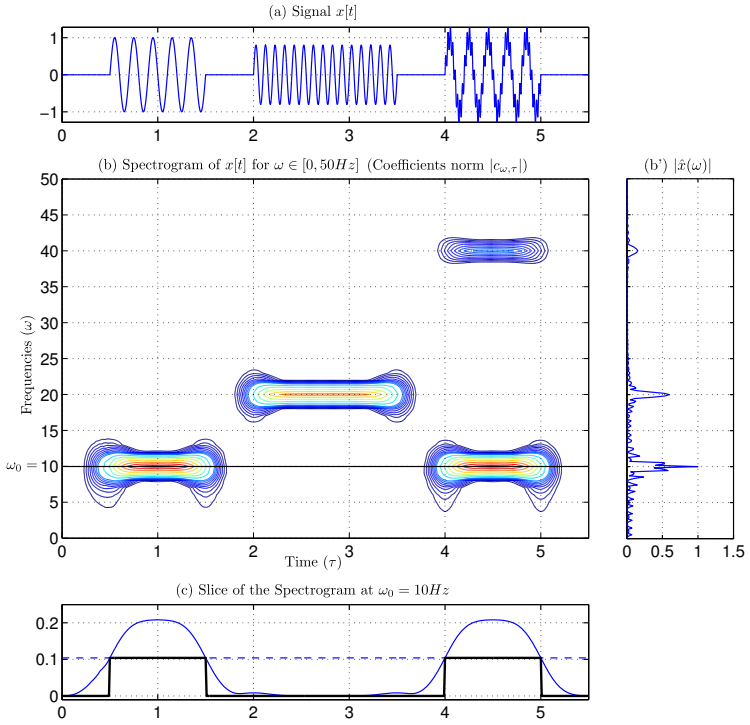

Fig. 2. Signal $x[t]$ in (a) is composed of different sinusoids at different times. A simple FT (b') exhibit peaks at frequencies 10, 20 and 40 without information about when they occur, whereas the spectrogram in (b) provide this information. We can see, e.g., that the frequency ω_0 =10 occurs at two distinct time intervals. In (c), a slice of the STFT at $\omega = \omega_0$ provides a signal that takes higher values at times when ω_0 is in the spectrum of x. Using a simple threshold, it then defines a predicate for the detection of frequency ω_0 along the time axis.

that $|c_{\omega,\tau}|$ be non-zero if and only if x contains a component of frequency ω at time τ. Unfortunately, such ideal mapping cannot be obtained. To begin with, it is intuitively clear that low frequencies require an amount of time at least larger than the corresponding periods to be detectable. Moreover, there is an obvious technical limitation related to the sampling rate of x and the discretization of the FT by the FFT algorithm. But even if we ignore discretization errors and assume that we work in an ideal, continuous domain, there is a fundamental limitation of the time-frequency resolution than can be achieved, related to the Uncertainty Principle of Heisenberg. We sketch its development next (a thorough explanation is provided in [22]) since it provides a practical method for choosing the appropriate window function g for a desired accuracy of time-frequency detection with the STFT. The idea is as follows: assume a hypothetical signal x with an energy concentrated exactly at the time τ_0 and frequency ω_0. Then

the STFT of x using a window function g will "spread" the concentrated energy of x in a box (so called an *Heisenberg box* [22]) in the time-frequency domain which has dimensions $\sigma_\tau(g) \times \sigma_\omega(g)$ given by

$$\sigma_\tau^2(g) = \int_{-\infty}^{+\infty} t^2 g(t)^2 dt \quad \text{and} \quad \sigma_\omega^2(g) = \int_{-\infty}^{+\infty} \nu^2 \hat{g}(\nu)^2 d\nu \tag{8}$$

The Uncertainty Principle asserts that the area of this box satisfies $\sigma_\tau \sigma_\omega \geq \frac{1}{2}$. It means that we cannot increase the accuracy in frequency detection without altering precision in time. The values of $\sigma_\tau(g)$ and $\sigma_\omega(g)$ can be easily estimated from the above formulae and can be used to optimize the trade-off between time and frequency detection. For instance, one has to make sure that the distance of two frequencies of interest is not small with respect to $\sigma_\omega(g)$.

5 Music

As observed in [10], human acoustic perception is a prime example of analyzing signals based on a combination of time and frequency features. In this section, we illustrate the applicability of TFL in formalizing and recognizing melodies starting with the basic task of note detection.

A note is characterized by a strong component at a fundamental frequency, or *pitch* ω. To obtain a note detection predicate, we thus define a spectral operator, pitch_ω, such that $\text{pitch}_\omega(x)[t]$ is the amplitude of frequency ω in signal x around time t. This operator must be able to tolerate small pitch variations while discriminating a note from its two closest neighboring notes, with pitches $\omega_1 = 2^{-\frac{1}{12}}\omega$ and $\omega_2 = 2^{\frac{1}{12}}\omega$. Thus, pitch is defined as the STFT $\text{pitch}_\omega(x)[t] = \hat{x}_L(\omega, t)$, where the size L of the window function is chosen to achieve the required time-frequency resolution (see Section 4). Using pitch_ω, a predicate detecting, e.g., the note A with pitch $\omega_A = 440$Hz can be: $\mu_A = \text{pitch}_{\omega_A}(x) > \theta$. The only parameter which remains to be fixed is the threshold θ. It determines the robustness of the predicate to variations in volume, pitch or duration. If θ is large, it will be more sensitive to such variations, increasing the chance of false negative. Conversely, if it is small, it will tolerate more pitch variation but increase the chance of false positive, e.g., by recognising a wrong note. Fig. 3 displays the result of applying the pitch function to the detection of an F.

Specifying Melodies Music notation provides means to specify both the duration of a note and the pace of a piece, also called *tempo*. Tempo is expressed in units of *beats per minute* (bpm). The piece in Fig. 4 has a pace of 120 bpm. This means that an eighth note (F, G), a quarter note (E, A, B), a half note (D), and a whole note (C) have durations, respectively, of 0.25, 0.5, 1 and 2 seconds. The pause between G and A is one second long. In our experiments, we have generated, using a MIDI Player, two different signals that correspond to the performance of this melody by a violin and by an organ; see Fig. 4. We

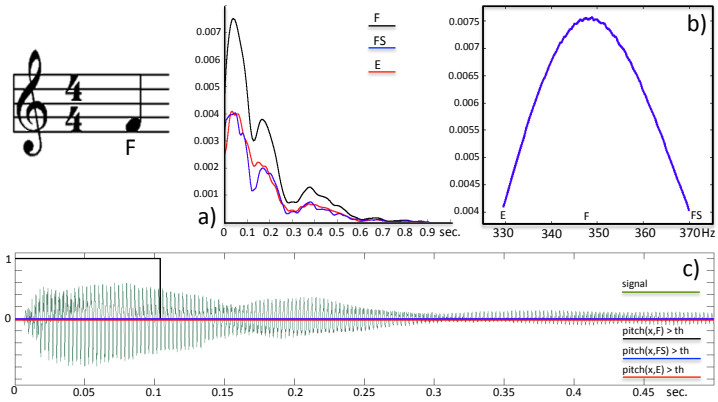

Fig. 3. Note detection: a) The STFT produced by the note F for its nominal frequency and the frequencies of its closest notes E and FS; b) Max amplitudes in time for the note F of a range of frequencies around its nominal frequency; c) The predicate μ_F is satisfied by the signal and the predicates μ_{FS} and μ_E are not.

then checked three properties on both signals. The first two properties are: C is played for two seconds: $\Box_{[0,2]}\mu_C$, and D is played for one second: $\Box_{[0,1]}\mu_D$. As one can see from the corresponding satisfaction signals in Fig. 4, the first holds at the beginning of the signal and the second holds at the beginning of the second note. The property $\Box_{[0,2]}\mu_C \wedge \Box_{[2,3]}\mu_D$ specifies the beginning of the melody and is found to hold at time zero. The last property ignores duration and specifies that the order of the notes corresponds to a diatonic scale of seven notes with a pause (expressed using a time domain predicate of the form $|x[t]| \leq \epsilon$ for some threshold) between G and A: $\mu_C \,\mathcal{U}\, \mu_D \,\mathcal{U}\, \mu_E \,\mathcal{U}\, \mu_F \,\mathcal{U}\, \mu_G \,\mathcal{U}\, \sigma \,\mathcal{U}\, \mu_A \,\mathcal{U}\, \mu_B$.

Recognizing a Blues Melody For the last experiment, we tested our framework by trying to verify that a guitar melody, (imperfectly) played and recorded by one of the authors, was indeed a Blues melody. To do this, we built a formula based on the fact that standard blues is characterized by a 12-bar structure (a bar being basically four beats). In the key of E, it is as follows: `E E E E | A A E E | B A E E`. Note that a bar in E does not mean that we play four beats of E notes in a row. There can be different notes, but the overall bar should *sound like* a melody in the key of E. If we assumed that in a bar of E there should be *at least* one E note played, and similarly for A and B, it would be easy to write a formula that directly translates the above structure. However, this would be too strict in transcribing the above blues pattern for the blues line that we recorded. Indeed, our melody does not have an E in the fourth bar. Instead, we verified a simpler (but definitely bluesy) formula which looks for a starting E, an A in bars 5-6, and the so-called "turn-around" (the sequence B A E) in bars 9-11: $\varphi_{\text{blues}} = \mu_E \wedge \Diamond_{[5b,6b]}\mu_A \wedge \Diamond_{[8b,9b]}(\mu_B \wedge \Diamond_{[b,2b]}\mu_A \wedge \Diamond_{[2b,3b]}\mu_E)$.

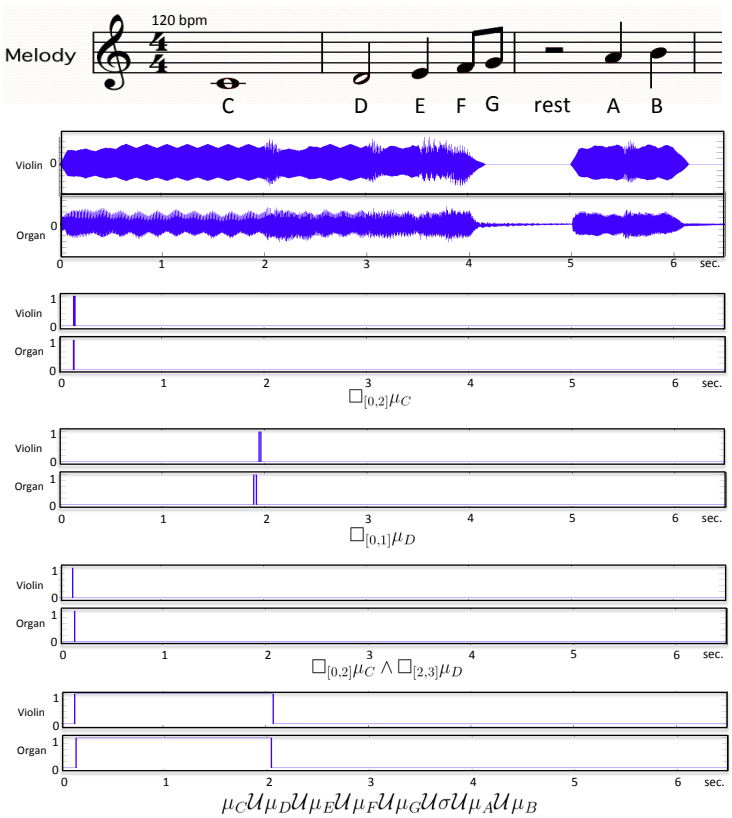

Fig. 4. Note and melody detection in violin and organ performances.

Our results are presented in Figure 5. The signal was recorded at 44 kHz for a length of 1320960 samples. The formula takes 6.9 s to be evaluated on a laptop with a Core i7 processor and 8 GB of memory.

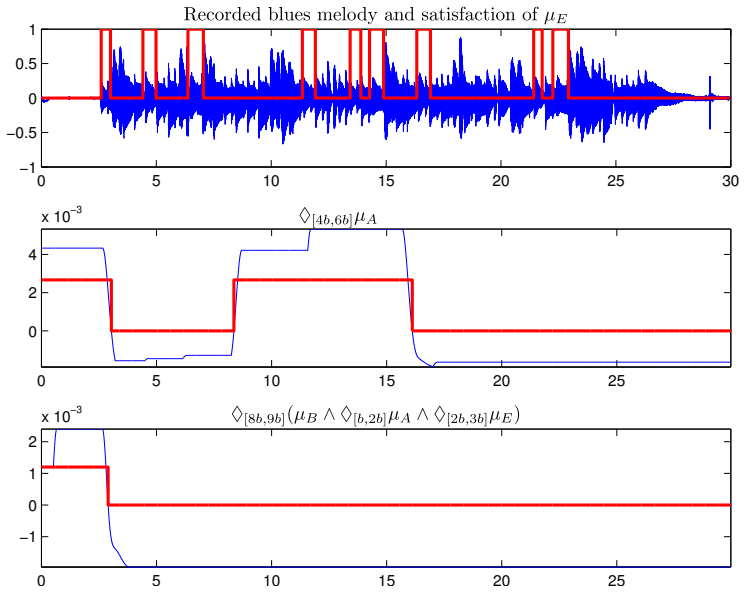

Fig. 5. Formula φ_{blues} is the conjunction of μ_E (top, with signal), $\Diamond_{[4b,6b]}\mu_A$ (middle), and $\Diamond_{[8b,9b]}(\mu_B \wedge \Diamond_{[b,2b]}\mu_A \wedge \Diamond_{[2b,3b]}\mu_E)$ (bottom). Since the three formulae are satisfied at the time of the first E, φ_{blues} is satisfied by our recording.

Implementation We implemented the function pitch in Matlab and defined STL formulae using the Breach tool [5] (examples available at `http://www-verimag.imag.fr/~donze/breach_music_example.html`). Breach implements the full STL syntax (Boolean and temporal operators) on top of STL predicates of the form $\mu = f(x, p) > \theta$, where f is some signal operator and p is a parameter vector. The function f can be an arithmetic expression such as $2 * x[t] + p$ or a routine implemented separately and available in the Matlab environment, such as the pitch_ω routine or any other implementation of spectral operators $f_{L,\omega}$. This makes the implementation of TFL straightforward in the Breach framework.

6 Discussion

We have presented TFL, a specification formalism for time and frequency properties of signals, supported by a monitoring algorithm implemented in the Breach

tool, and showed it in action on real acoustic signals. We believe that the expressivity added by the temporal operators can lead to new ways to specify music and, in particular, will allow us to define formulae that can quantify the amount of deviation of a performance from the "nominal" melody. Combining our formalism with complementary learning machinery (such as hidden Markov models, used in speech recognition), one could automatically "tune" note predicates to a given performance.

Although Music can definitely benefit from TFL, it is not necessarily the primary application domain we have in mind for this formalism. The convergence of technologies, where a typical system-on-a-chip features digital and analog components, including radio transmitters, likewise requires a convergence of modeling and analysis techniques used by the different engineering communities involved, and TFL is a step in this direction. Other extensions include the use of the more versatile Wavelet Transform for time-frequency analysis and the extension of the logic to *spatially extended* phenomena such as wave propagation in the spirit of [12,13], where dynamic cardiac conditions are specified and detected. Also note that the spectrogram is a two-dimensional entity indexed by both time and frequency, and that TFL is currently biased toward time. It would be interesting to explore a specification formalisms that can alternate more freely between temporal, frequential and spatial operators. On the engineering application side, we intend to apply the logic to specifying and monitoring the behavior of analog circuits.

References

1. A. Benveniste, P. Le Guernic, and C. Jacquemot. Synchronous programming with events and relations: the signal language and its semantics. *Sci. Comput. Program.*, 16(2):103–149, 1991.
2. P. Caspi, D. Pilaud, N. Halbwachs, and J. Plaice. Lustre: A declarative language for programming synchronous systems. In *POPL*, pages 178–188, 1987.
3. A. Chakarov, S. Sankaranarayanan, and G. Fainekos. Combining time and frequency domain specifications for periodic signals. In *RV*, 2011.
4. B. d'Angelo, S. Sankaranarayanan, C. Sánchez, W. Robinson, B. Finkbeiner, H. Sipma, S. Mehrotra, and Z. Manna. Lola: Runtime monitoring of synchronous systems. In *TIME*, pages 166–174. IEEE, 2005.
5. A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *CAV*, pages 167–170, 2010.
6. A. Donzé, E. Fanchon, L. M. Gattepaille, O. Maler, and P. Tracqui. Robustness analysis and behavior discrimination in enzymatic reaction networks. *PLoS One*, 6(9), 2011.
7. A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *FORMATS*, volume 6246 of *LNCS*, pages 92–106. Springer, 2010.
8. J. B. G. Fourier. *Theórie analytique de la chaleur*. Gauthier-Villars et fils, 1888.
9. M. Frigo and S. G. Johnson. The fastest Fourier transform in the west. Technical Report MIT-LCS-TR-728, Massachusetts Institute of Technology, September 1997.
10. D. Gabor. Theory of communication. part 1: The analysis of information electrical engineers. *Journal of the IEEE - Part III: Radio and Communication Engineering*, 93(26):429 – 441, November 1946.

11. G. Giorgidze and H. Nilsson. Switched-on yampa: declarative programming of modular synthesizers. In *PADL*, pages 282–298. Springer-Verlag, 2008.
12. R. Grosu, E. Bartocci, F. Corradini, E. Entcheva, S. Smolka, and A. Wasilewska. Learning and detecting emergent behavior in networks of cardiac myocytes. In *HSCC'08,*, volume 4981 of *LNCS*, pages 229–243. Springer Verlag, 2008.
13. R. Grosu, S. Smolka, F. Corradini, A. Wasilewska, E. Entcheva, and E. Bartocci. Learning and detecting emergent behavior in networks of cardiac myocytes. *Communications of the ACM*, 52(3):1–10, March 2009.
14. D. Harel and A. Pnueli. On the development of reactive systems. In K. R. Apt, editor, *Logics and Models of Concurrent Systems*, NATO ASI Series, pages 477–498. Springer-Verlag, 1985.
15. B. B. Hubbard. *The world according to wavelets. The story of a mathematical technique in the making.* CRC Press, 2nd edition, 2010.
16. P. Hudak, A. Courtney, H. Nilsson, and J. Peterson. Arrows, robots, and functional reactive programming. In *Advanced Functional Programming, 4th International School, volume 2638 of LNCS*, pages 159–187. Springer-Verlag, 2002.
17. K. Jones, V. Konrad, and D. Nickovic. Analog property checkers: a DDR2 case study. *Formal Methods in System Design*, 2009.
18. Y. Kesten and A. Pnueli. A compositional approach to CTL* verification. *Theor. Comput. Sci.*, 331(2-3):397–428, 2005.
19. O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *FORMATS/FTRTFT*, pages 152–166, 2004.
20. O. Maler, D. Nickovic, and A. Pnueli. From MITL to timed automata. In *FORMATS 2006*, volume 4202 of *LNCS*, pages 274–289. Springer, 2006.
21. O. Maler, D. Nickovic, and A. Pnueli. Checking temporal properties of discrete, timed and continuous behaviors. In *Pillars of Computer Science*, pages 475–505, 2008.
22. S. Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, 2008.
23. M. Michel. Composition of temporal operators. *Logique et Analyse*, 110-111:137–152, 1985.
24. S. Mukherjee, P. Dasgupta, and S. Mukhopadhyay. Auxiliary specifications for context-sensitive monitoring of AMS assertions. *IEEE Transactions on CAD*, 30(10):1446 –1457, 2011.
25. D. Nickovic and O. Maler. AMT: A property-based monitoring tool for analog systems. In *FORMATS*, volume 4763 of *LNCS*, pages 304–319. Springer, 2007.
26. I. Pembeci, H. Nilsson, and G. Hager. Functional reactive robotics: an exercise in principled integration of domain-specific languages. In *PADP*, pages 168–179. ACM, 2002.
27. J. L. Pino, S. Ha, E. A. Lee, and J. T. Buck. Software synthesis for DSP using Ptolemy. *VLSI Signal Processing*, 9(1-2):7–21, 1995.
28. A. Pnueli. The temporal logic of programs. In *Proc. 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57, 1977.
29. A. Pnueli. The Temporal Semantics of Concurrent Programs. *Theoretical Computer Science*, 13:45–60, 1981.
30. A. Pnueli and A. Zaks. On the merits of temporal testers. In *25 Years of Model Checking*, pages 172–195, 2008.
31. A. N. Prior. *Past, Present, Future.* Oxford, 1969.
32. N. Rescher and A. Urquhart. *Temporal Logic.* Springer, 1971.
33. M. Y. Vardi. From philosophical to industrial logics. In *ICLA*, pages 89–115, 2009.