Comments on "Temporal Logics for Real-Time System Specification"

Carlo A. Furia[1], Matteo Pradella[2], Matteo Rossi[1]
[1]Dipartimento di Elettronica e Informazione, Politecnico di Milano
[2]CNR IEIIT-MI
{furia, pradella, rossi}@elet.polimi.it

Abstract

The article "Temporal Logics for Real-Time System Specification" [3] surveys some of the relevant literature dealing with the use of temporal logics for the specification of real-time systems. Unfortunately, [3] introduces some imprecisions that might create some confusion in the reader. While a certain degree of informality is certainly useful when addressing a broad audience, imprecisions can negatively impact the legibility of the exposition. We clarify some of the remarks of [3] on a few topics, in an effort to contribute to the usefulness of the survey for the reader.

The article “Temporal Logics for Real-Time System Specification” (ACM Computing Surveys, March 2000 [3]) surveys some of the relevant literature dealing with the use of temporal logics for the specification of real-time systems. Such a topic is of great importance and interest to a large community of both researchers and practitioners, especially during recent years — when substantial successful developments in the formal verification of embedded and real-time systems have occurred. However, while presenting the survey results, [3] also introduces some imprecisions that might create some confusion in the reader. While a certain degree of informality is certainly useful when addressing a broad audience such as the readership of the Computing Surveys, imprecisions can negatively impact the legibility of the exposition. Hence, this correspondence clarifies some of the remarks of [3] on a few selected topics, in an effort to contribute to the usefulness of the survey for the reader.

1 Completeness and Soundness

Section 2.1 of [3] introduces the definitions of *completeness* and *soundness* of a deductive system. According to [3], a deductive system is *complete* when:

> “It is possible to construct a demonstration for all theorems of the theory”

And a deductive system is *sound* when:

> “Each theorem that can be demonstrable with the logic is a theorem of the logic”

The above statements are not correct, since they define soundness and completeness independently of the semantic concept of truth (or, more precisely, validity). In a deductive system a theorem is, by definition [2], a well-formed formula that has a proof (i.e., that can be demonstrated); so the two statements above are tautologies.

Instead, a deductive system $\mathcal{F}$ is *sound* when:

> “every theorem of $\mathcal{F}$ is valid” [2, pg. 80]

and it is *complete* when:

> “every valid well-formed formula of $\mathcal{F}$ is a theorem” [2, pg. 94].

Also, in Section 3.8 of [3] it is said the following about complete deductive systems:

> “It should be noted that it is never possible to build a *complete* deductive system”

The above statement is not specialized nor clarified in any manner in the text, and implies that all deductive systems are necessarily incomplete. This is however false, as there exist numerous complete deductive systems for several logic languages, such as propositional logic [2] or, as [3] itself suggests elsewhere, PTL (propositional temporal logic).

2 Expressiveness

Let us point out two meanings that are associated, in different scientific fields, with the word "expressiveness". This is necessary since they are both used in [3], but without distinguishing precisely between them. Thus, it is not always evident which is the intended meaning when "expressiveness" is mentioned in [3], as Section 7 further discusses.

The first meaning of the word "expressiveness" refers to the technical capability of a formalism to describe extensive classes of properties (e.g., [1, 5]). For instance, the property "*A until B*" is not expressible with a logic that just uses the unary modalities *eventually* $\Diamond$ and *always* $\Box$ (as first proved by Kamp [15]). In the large majority of literature about temporal logics this is the meaning that is implicitly given to the notion of expressiveness. In this case, the meaning of the word "expressiveness" is precise and mathematically-based.

The second meaning, instead, has an informal, subjective nature, and refers to the ease, naturalness, and simplicity with which one can formalize some property with the given formalism. For instance, with reference to programming languages (e.g., [8]), one may say that Pascal is more expressive than Assembly language since it abstracts many details away and allows programmers to express algorithms in a more compact form. This notion has been formalized in programming languages [9].

In the rest of this paper, when using the word "expressiveness" (without further qualifications), we will refer to the first, formal meaning. Whenever we will need to refer to the second, informal meaning, instead, we will explicitly point it out.

3 Metric on Time

Section 3.4 of [3] discusses the issue of metric on time. It is said that

> "The temporal operators [of temporal logic] presented in Section 2.4 are qualitative, since it is not possible to give an exact measure (i.e., duration, timeout) for events and among events."

It is also said that

> "A typical way to add a metric for time is to allow the definition of bounded operators – for example [...] $\Diamond_{\leq 5}A$, which means that A is eventually true within 5 time units."

The first statement above is too strong, and distinctions should be made. In fact, the $\bigcirc$ *next* operator (one of those listed in Section 2.4 of [3]) can actually be used, under suitable conditions, to introduce a metric on time.

More precisely, whether one can express metric properties using the $\bigcirc$ operator depends also on the structures on which the temporal logic is interpreted. Whenever the underlying structure of time is isomorphic to the natural numbers with their usual ordering, every natural k can be associated with a state (in some sense) s_k. If one assumes that the time elapsed between any pair of consecutive states s_k and s_{k+1} is one time unit, "next" simply stands for "next time instant". With this assumption one can define, for instance,

$$\Diamond_{\leq 5} A \triangleq \bigcirc(A \vee \bigcirc(A \vee \bigcirc(A \vee \bigcirc(A \vee \bigcirc A))))$$

which shows that, however cumbersome, it is indeed possible to give an exact measure of the elapsing time. Then, operators such as $\Diamond_{\leq \alpha}$ (with α a constant) are introduced to simplify writing temporal constraints, but they do not add expressive power to the logic (e.g., [6] shows this for RTCTL and CTL). Of course, different assumptions are possible (e.g., [4]).

Indeed, if the only metric operator available is $\bigcirc$, it may be clumsy or even impossible to express some complex quantitative constraints. However, bounded durations and timeouts (which are still real-time properties of great interest [16]) can be expressed using only the $\bigcirc$ operator, possibly at the price of sacrificing conciseness. In other words, it is often a matter of convenience, rather than of possibility.

4 Logic Executability

Section 3.9 of [3] presents the problem of executability of temporal logic specifications. It is stated that "there are at least three different definitions of executability".

The first definition is said to correspond "to that of decidability of the validity problem", and the reference [21] is given. The second corresponds to what is also called *history checking* [7]. About the third, it is written in [3] that it

> "consists of using the system specification itself as a prototype or implementation of the real-time system, thus allowing, in each time instant, the *on-line* generation of system outputs on the basis of present inputs and its internal state and past history."

It is also remarked that

> "there exists [sic] only few executable temporal logics that can be used to build a system prototype according to meaning (iii) of executability."

This is further stressed in Table 7 of [3], where a column labeled "Logic executability" summarizes the status of this feature for the temporal logics presented in the paper. According to the caption, the executability of each logic is classified under one of the following five labels: "N=No, (N)=no in the general case, Y=yes, (Y)=yes in some specific case, NA=not available". Only two of the logics listed in Table 7 carry a "Y"; four carry a "(Y)", and the rest (the vast majority) carry a "NA". In fact, of many of the logics surveyed in Section 4 of [3], it is said that "no known results (about executability) are available".

However, most temporal logics can be easily restricted to a proper subset that is isomorphic to basic temporal logic with finite domains only (except for the temporal domain, which is usually assumed denumerable). This subset is clearly executable, since it is reducible to PTL, which is indeed executable (as shown in the seminal work by Gabbay [11] and in accordance with Table 7 of [3]). Under this respect, most (if not all) of the logics that are tagged "NA" in [3, Table 7] in the "executability" column are in fact "executable in some specific case" (hence, "(Y)"), regardless of whether an actual implementation of the execution algorithm for the specific subset has been provided.

In addition, while [3] mentions the issue of the "computational complexity of the algorithms" to execute temporal logics, it does not point out that this is precisely what distinguishes the third definition from the two previous ones, i.e., the possibility of *implementing* an *efficient* algorithm for the execution of a logic specification (that is, building a model for that specification [10]). In other words, executability according to the third definition is a property regarding the complexity of an execution (i.e., model-building) algorithm that is available for a logic language [10]. Indeed, the limitations to the executability of logic languages lie in problems of complexity and incompleteness. Therefore, as Merz indicates [20], "the design of a temporal logic-based programming language involves a trade-off between expressiveness and (efficient) implementability".

5 Past and Future

The availability of past operators is one of the features of temporal logics analyzed in [3]. In this respect, some contradictory statements are introduced; this section aims at clarifying them.

On the one hand, some passages of [3] state that, whenever a temporal logic does not explicitly provide past operators, it is impossible to express requirements about the past in that logic. For example, discussing PTL, [3] states that

> "The asymmetry of the logic [...] and the absence of the operator **since** does not permit specification of requirements about the order of events in the past."

However, it is a well-established result that PTL (interpreted over time models isomorphic to the natural numbers, as it is customary) with both past and future operators is (initially) equivalent in expressive power to PTL with future operators only [13, 11, 5, 12]. Obviously, the presence of explicit past operators does *facilitate* [18] the writing of formulas about the past, and enhance their conciseness, but it is not strictly necessary.

As a simple example, consider the specification "A can happen only if B took place 2 time instants before", which is written in PTL with past operators as $A \rightarrow \newcommand{\pnext}{\bullet} \bullet\bullet B$. This formula involves both past and metric operators; nonetheless it is expressible in PTL with future operators only as: $\neg B \rightarrow \bigcirc\bigcirc\neg A$.

On the other hand, other passages in [3] maintain that the availability of past operators is not necessary — as far as the expression of requirements about the past is concerned — whenever the past is bounded. For instance, at the end of Section 2, [3] states that

> "If the temporal logic has the *begin* property (e.g., stating that the temporal domain is bounded in the past [. . .]), the operator **until** is enough to complete the logic's expressiveness: when the past is limited, the operator **since** is not necessary."

However, this is also not true in general, as there exist temporal logics that are strictly more expressive when endowed with past operators even when they are interpreted over structures with a bounded past.

This is the case also of "classical" temporal logic PTL, when interpreted over the non-negative real numbers as time domain with a bounded past. Since Kamp's seminal work [15], the expressiveness of several PTL variants — with both past and future temporal operators, with only future ones, with a discrete time domain, with a dense/continuous one, etc. — has been thoroughly studied. In particular, we already remarked above that it has been long known [13] that PTL with future operators only is as expressive (with respect to initial satisfiability) as its variant with both past and future operators, when the temporal domain is the set of natural numbers. On the contrary, it has never been conjectured that the same holds when the temporal domain is the set of non-negative reals. In fact, more recently Hirshfeld and Rabinovich proved [14] that PTL with *until* only is strictly less expressive than its variety with *until* and *since*, over the non-negative reals. Another example is the (metric) temporal logic MTL with the qualitative *since* operator, which is strictly more expressive than its future-only variety, even if structures with a bounded past are adopted [22].

In summary, the relationship between the availability of past operators and the expressiveness of a temporal logic is subtle and depends on several different semantic aspects.

6 A Running Example

To demonstrate the use of temporal logics, [3] informally introduces a simple example of real-time specification (Figure 6 of [3]), which is then formally specified with each of the considered metric temporal logics. The example is that of a predicate E whose occurrence triggers predicates $startA$ and $endA$ within t_e time units, thus marking an interval in which A is true.

[3] in some cases presents formulas that are claimed to be equivalent (i.e., to have the same models). For instance, consider the two TRIO formulas for the example presented in Section 4.14 of [3] (similar considerations can be made for the MTL formulas of Section 4.15):

$$Alw(E \rightarrow \exists t((0 < t < t_e) \wedge Futr(endA, t) \wedge WithinF(startA, t))) \qquad (1)$$

and[1]

$$Alw(E \rightarrow WithinF(endA, t_e) \wedge \neg Until(\neg startA, endA)) \qquad (2)$$

It is said that they represent the "same specification", while they do not. In fact, Formula (2) is stronger than Formula (1), since (2) forces the first occurrence of *endA* after E to be preceded by an occurrence of *startA*, while (1) does not.

7 Point- vs. Interval-Based Logics

In Section 3.3 of [3], the differences between logics based on time points and logics based on time intervals are discussed. The following is stated:

> "Interval-based temporal logics are more expressive [than point-based logics], since they are capable of describing events in time intervals, and a single time instant is represented with a time interval of one."

As discussed in Section 2, there are two different meanings that can be given to the word "expressiveness", but it is not apparent what is the intended one in the sentence above.

If expressiveness in the formal sense is meant, the claim that interval-based logics are more expressive is not correct. For example, over discrete time, every finite interval can be represented by a finite union of discrete points. More specifically, there exist point-based temporal logics that are strictly more expressive than others based on intervals. For instance, the MTL logic [16] (a point-based formalism according to the taxonomy of [3, Table 7]) is strictly more expressive than the interval-based TILCO logic [19], since only the former allows explicit quantification over time variables. In general, the introduction of temporal logics based on intervals, rather than points, has been supported essentially by claims of simplification in writing specifications, not for reasons of expressiveness [5, 17].

If, on the other hand, [3] referred to expressiveness in its informal sense, the statement above is too vague and needs clarification. To compare meaningfully the informal expressiveness of formalisms one should first establish that they have "similar" formal expressiveness. Otherwise, the comparison is irrelevant because the classes of properties they are capable of representing are too different. Since, as we remarked above, the relationship between the expressiveness (in the formal sense) of a point-based logic and that of an interval-based one depends crucially on the specific formalisms considered, the statement above should be further qualified.

[1]Notice that in Formula (2) we have swapped the arguments of the *Until* to conform with TRIO's usual syntax (which is different than that used in the paper).

8 Implicit and Explicit Time

Section 3.6 of [3] introduces the concepts of implicit and explicit time. It is said that

> "The explicit specification of time allows the specification of expressions that have no sense in the time domain — e.g., the activation of a predicate when the time is even."

On the contrary, a property such as "predicate P occurs at all time instants that are multiple of a constant n" is of interest in timed systems (consider for instance the behavior of a counter, or the clock signal of a synchronous integrated circuit), and the impossibility of expressing such a property in PTL [23] spawned a number of PTL extensions. However, this limitation of PTL does not depend on the fact that time in that logic is implicit. In fact, [5, Sec. 6.1.1] shows that a second-order extension of PTL, in which it is possible to quantify over propositions and where time is still implicit, precisely allows one to express the property above.

In summary, the possibility of expressing formulas that describe particular temporal patterns — such as a property holding at all even instants of time — is not a consequence of adopting explicit, rather than implicit, time, as the statement above suggests.

References

[1] R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104:35–77, 1993.

[2] P. B. Andrews. *An Introduction to Mathematical Logic and Type Theory.* Academic Press, 1992.

[3] P. Bellini, R. Mattolini, and P. Nesi. Temporal logics for real-time system specification. *ACM Computing Surveys*, 32(1):12–42, March 2000.

[4] E. Y. Chang, Z. Manna, and A. Pnueli. Compositional verification of real-time systems. In *Proceedings of LICS'94*, pages 458–465, 1994.

[5] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, pages 996–1072. 1990.

[6] E. A. Emerson, A. K. Mok, A. P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. *Real-Time Systems*, 4(4):331–352, 1992.

[7] M. Felder and A. Morzenti. Validating real-time systems by history-checking TRIO specifications. *ACM Transactions on Software Engineering and Methodology*, 3(4):308–339, October 1994.

[8] M. Felleisen. On the expressive power of programming languages. In *Proceedings of ESOP'90*, volume 432 of *LNCS*, pages 134–151, 1990.

[9] M. J. Fischer. Lambda-calculus schemata. *Lisp and Symbolic Computation*, 6(3/4):259–288, 1993.

[10] M. Fisher and R. Owens. An introduction to executable modal and temporal logics. In *Proceedings of the Workshop on Executable Modal and Temporal Logics*, 1993.

[11] D. M. Gabbay. The declarative past and imperative future. In *Proceeding of TLS'87*, volume 398 of *LNCS*, pages 409–448, 1987.

[12] D. M. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic (vol. 1): mathematical foundations and computational aspects.* Oxford University Press, 1994.

[13] D. M. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal basis of fairness. In *Proceedings of POPL'80*, pages 163–173, 1980.

[14] Y. Hirshfeld and A. M. Rabinovich. Future temporal logic needs infinitely many modalities. *Information and Computation*, 187(2):196–208, 2003.

[15] J. A. W. Kamp. *Tense Logic and the Theory of Linear Order.* PhD thesis, University of California at Los Angeles, 1968.

[16] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[17] R. Koymans. (Real) Time: a philosophical perspective. In *Real-Time: Theory in Practice*, volume 600 of *LNCS*, pages 353–370, 1992.

[18] F. Laroussinie, N. Markey, and P. Schnoebelen. Temporal logic with forgettable past. In *Proceedings of LICS'02*, pages 383–392, 2002.

[19] R. Mattolini and P. Nesi. An interval logic for real-time system specification. *IEEE Transactions on Software Engineering*, 27(3):208–227, March 2001.

[20] S. Merz. Efficiently executable temporal logic programs. In M. Fisher and R. Owens, editors, *Proceedings of the Workshop on Executable Modal and Temporal Logics (IJCAI'93).* Springer-Verlag, 1993.

[21] B. Moszkowski. *Executing temporal logic programs.* Cambridge University Press, May 1986.

[22] P. Prabhakar and D. D'Souza. On the expressiveness of MTL with past operators. In *Proceedings of FORMATS'06*, volume 4202 of *LNCS*, pages 322–336, 2006.

[23] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1):72–99, 1983.