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]IEIIT, Consiglio Nazionale delle Ricerche
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{furia, pradella, rossi}@elet.polimi.it

April 17, 2008

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.

Completeness and Soundness. Section 2.1 of [3] introduces the definitions of *completeness* and *soundness* of a deductive system which are essentially 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]. In Section 3.8 of [3] it is said that "it is never possible to build a *complete* deductive system". This statement is clearly 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).

Expressiveness. In the literature, two different meanings are associated with term *expressiveness*. The first one, common in much of the literature on temporal logics (e.g.[1, 4]), refers to the ability to describe a class of properties. E.g., the property "A until B" is not expressible with a logic which only uses the unary modalities *eventually* and *always* [12]. The second meaning, sometimes used informally, refers to the ease, and simplicity with which one can specify some behavior with a given formalism. E.g., 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 [6]. Unless otherwise indicated, we will use expressiveness in the first sense.

Metric on Time. Section 3.4 of [3] claims that temporal operators are qualitative as the $\bigcirc$ *next* does not involve an exact measure, therefore to add a metric for time bounded operators are needed, e.g. $\Diamond_{\leq 5} A$, which means that A will be true within

5 time units. This is not always true, since time can be interpreted to be isomorphic to natural numbers: $\bigcirc$ is then durational. 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 it is possible to give an exact measure of the elapsing time.

Logic Executability. Section 3.9 of [3] presents three different definitions of executability of temporal logic specifications. The first is said to correspond "to that of decidability of the validity problem" [16]; the second corresponds to *history checking* [5]; the third consists of using the system specification itself as a prototype or implementation of the system, thus allowing the *on-line* generation of system outputs on the basis of present inputs and its internal state and past history. [3] also states 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, where the column "Logic executability" summarizes this feature for the considered logics. 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 carry a "Y"; four carry a "(Y)", and the rest (the vast majority) carry a "NA". However, most temporal logics can be 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 reducible to PTL, which is executable ([8] and in accordance with Table 7 of [3]). Therefore most of the logics tagged "NA" 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.

Also, while [3] mentions the issue of the "computational complexity of the algorithms" to execute temporal logics, it does not point out that this is what distinguishes the third definition from the two previous ones, i.e., the possibility of *implementing* an *efficient* algorithm for building a model for a logic specification [7].

Past and Future. [3] states that, whenever a temporal logic does not explicitly provide past operators, it is impossible to express requirements about the past in that logic. However, it is a well-established result that PTL (interpreted over time models isomorphic to the natural numbers, as customary) with both past and future operators is (initially) equivalent in expressive power to PTL with future operators only [10, 8, 4, 9]. The presence of explicit past operators does *facilitate* the writing of formulas about the past, and enhance their conciseness, but it is not strictly necessary.

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. 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. In fact, more recently Hirshfeld and Rabinovich proved [11] that PTL with *until* only is strictly less expressive than its variety with *until* and *since*, over the non-negative reals. Another example is MTL with the qualitative *since* operator, which is more expressive than its future-only variety, even if structures with a bounded past are adopted [17].

A Running Example. [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. E.g., 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):
1) $Alw(E \rightarrow \exists t((0 < t < t_e) \wedge Futr(endA, t) \wedge WithinF(startA, t)))$;
2) $Alw(E \rightarrow WithinF(endA, t_e) \wedge \neg Until(\neg startA, endA))$[1]
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.

Point- vs. Interval-Based Logics. Section 3.3 of [3] states that "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 previously discussed, there are two different meanings of "expressiveness", but it is not apparent what is the intended one in the sentence above.

If expressiveness in the formal sense is meant, the above claim is not correct. E.g., over discrete time, every finite interval can be represented by a finite union of discrete points. The MTL logic [13] (a point-based formalism according to the taxonomy of [3, Table 7]) is more expressive than the interval-based TILCO logic [15], since only the former allows explicit quantification over time variables. 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 [4, 14].

If [3] referred to expressiveness in its informal sense, the statement above is too vague. 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.

Implicit and Explicit Time. Section 3.6 of [3] states 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 [18] spawned a number of PTL extensions. [4, 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, allows one to express the property above.

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.

[1] In (2) we have swapped the arguments of the $Until$ to conform with TRIO's usual syntax.

[4] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 996–1072. Elsevier Science Publishers, 1990.

[5] 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.

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

[7] M. Fisher and R. Owens. An introduction to executable modal and temporal logics. In M. Fisher and R. Owens, editors, *Proceedings of the Workshop on Executable Modal and Temporal Logics*. Springer-Verlag, 1993. A satellite workshop of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93).

[8] D. M. Gabbay. The declarative past and imperative future. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Proceeding of TLS'87*, volume 398 of *LNCS*, pages 409–448, 1987.

[9] D. M. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic (vol. 1): mathematical foundations and computational aspects*, volume 28 of *Oxford Logic Guides*. Oxford University Press, 1994.

[10] 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.

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

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

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

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

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

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

[17] 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.

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