Comments on "An Interval Logic for Real-Time System Specification"

Carlo A. Furia[†], Angelo Morzenti[†],
Matteo Pradella[‡], and Matteo G. Rossi[†]
[†] Dipartimento di Elettronica e Informazione, Politecnico di Milano
[‡] CNR IEIIT-MI
{furia, morzenti, pradella, rossi}@elet.polimi.it

Abstract

The paper "An Interval Logic for Real-Time System Specification" [5] presents the TILCO specification language and compares it to other existing similar languages. In this comment, we show that several of the logic formulas used for the comparison are flawed and/or overly complicated, and we explain why, in this respect, the comparison is moot.

Index Terms: D.2.4.d formal methods, F.4.1.k temporal logic, I.5.5.b real-time systems.

1 Introduction

When the paper "An Interval Logic for Real-Time System Specification" by Mattolini and Nesi (IEEE Transactions on Software Engineering, March 2001 [5]) was published, we believed that it contained some questionable statements; in addition, it seemed to us that the comparison between TILCO, the language presented in [5], and TRIO [3, 2] (of which some of the authors of the present paper are the creators) was inaccurate.

Recently, however, a further analysis of the contents of [5] revealed some important flaws, which, in our opinion, undermine some claims found in the paper.

The goal of this paper is therefore twofold: in the first place, it points out the most relevant technical errors of [5]; second, it defends the TRIO approach against some of the claims made in [5].

In doing so, it is not our goal to claim either of the two languages to be "better" than the other: we believe that each language has its own strengths and weaknesses. At the same time, any comparison must be based on technically sound arguments to be of any value. We believe that pointing out the main errors in [5] is thus useful to the reader to help present a more neutral view on the matter.

This paper is structured as follows: Section 2 presents the major technical flaws in the use of TILCO and temporal logics in general, Section 3 contains some remarks on the comparison with TRIO, and Section 4 draws the conclusions.

2 Technical Errors

We start with a discussion on the expressiveness of TILCO (Section 2.1), then we analyze some technical flaws of the formulas presented in the paper (Section 2.2).

2.1 Temporal Quantification and Expressiveness of TILCO

In describing the TILCO language, the authors of [5] often remark that TILCO prohibits explicit temporal quantification. For example, they state in the Introduction (and repeat in the Conclusions) that:

> [in TILCO] no explicit temporal quantification is allowed

Similarly, in Section 5.1 (pg. 221 of [5]) it is said that:

> TILCO specifications cannot be written in terms of temporal quantifications since this is not allowed by the language. Therefore the analyst must write simple and concise specifications.

These statements, which are also repeated elsewhere in the paper, are in fact consistent with the formal grammar for the language presented in Section 2.1 of [5]. However, the TILCO Formula (3) presented in Section 5.1 of [5] (which is reported in Figure 1 for ease of reading) actually contains *two explicit temporal quantifications*, namely:

$$\ldots \exists \delta.(\mathrm{rq}(a, \delta)?[-\delta, -5] \ldots$$

and:

$$\ldots \neg \exists a', \delta'.(\ldots \mathrm{rq}(a', \delta')?[-\delta', -5] \ldots$$

As a consequence, (3) is *not* a well-formed TILCO formula.

Indeed, there are also explicit temporal quantifications in another formula, namely (2) of [5]. This formula is written in TRIO, a language which allows this kind of quantification. Nonetheless, it is claimed that "(2) has been written by the authors to specifically avoid quantifications over time", even if the formula contains the same explicit temporal quantifications as (3).

Now, let us briefly consider, on the very same example, how prohibiting explicit temporal quantification impacts the *expressiveness* of the language. Formula (3) is a tentative TILCO specification of a resource allocator also presented, and specified in TRIO, in [3]. In Section 2.2 we will show why (3) is not a correct specification of the allocator, even if explicit temporal quantification were allowed in TILCO. Here, let us consider the more general problem of specifying the allocator in TILCO, in accordance with the informal description given in [3], which we repeat here for the sake of clarity:

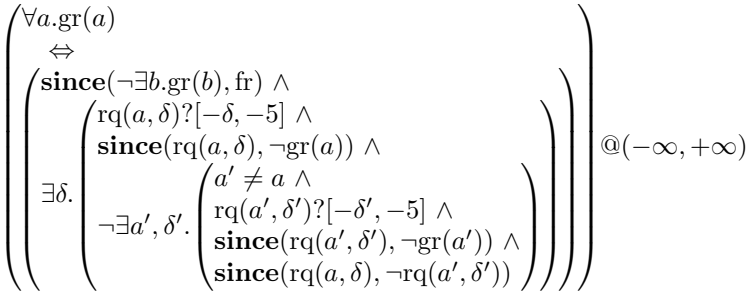

Figure 1: Formula (3) of [5].

> Each process p requires the resource by issuing the message $rq(p, \delta)$, by which it identifies itself to the allocator and provides a time-out δ for the request to be serviced. If the allocator is able to satisfy p's request within the indicated time-out, then it grants the resource to p by a $gr(p)$ signal; otherwise the request must be completely ignored. Once the process has used the resource granted to it by the allocator, it frees the resource by an fr signal. At any time, the allocator will grant the free resource to the least recent pending request.

To formalize such a behavior, we need to express the truth of predicate $rq(p, \delta)$ in a time interval whose length depends on the parameter δ of the predicate itself. Since the timeout δ is arbitrary, and represents a temporal quantity, it must be expressed as a (explicitly or implicitly) quantified temporal variable, and cannot be replaced by a constant. Therefore, in TILCO we *cannot* express the above behavior, since temporal quantification is not allowed (in particular, interval boundaries in TILCO can only be integer constants or $\pm\infty$).

In summary, the real-time allocator can be specified in languages such as MTL [4] and TRIO, but cannot be specified in TILCO. Therefore TILCO, if used consistently with its formal syntax, is strictly *less expressive* than MTL and TRIO.[1]

2.2 Flawed Examples

This section considers some examples of formulas presented in [5] that contain a number of flaws.

[1] The above holds even without considering that MTL and TRIO can deal with both discrete and dense temporal domains, while TILCO is defined only for discrete ones.

2.2.1 Resource Allocator Specification

As a first case, we consider the formulas presented in Section 5.1 ("A More Complex Example") of [5], which formalize the resource allocator presented above.

First, we notice that the natural language description given in [5] is rather vague, and contains undefined terms that are hard to understand without looking at the original description in [3], reported in Section 2.1 of the present paper. We also note that, in [5], it is claimed that the TRIO example used in [3] is a "simpler problem" than that developed in [5]. In fact, even if the "difficulty" of the two problems seems comparable, in this paper we are considering only the formulas presented in [5].

The authors of [5] show three different formulas representing formalizations of the allocator. First, two different TRIO formulas are introduced; then, a TILCO specification is presented.

TILCO formula. Let us start by considering the TILCO Formula (3), shown in Figure 1. In Section 2.1 we discussed why this formula is actually not a well-formed TILCO formula. Nonetheless, for the sake of comparison, let us assume that it is written in an extension of the TILCO language which allows explicit quantification over time, and let us evaluate its correctness as a specification of the allocator.

First of all, we notice that [5] introduces a minimum delay of 5 time units between the request for the resource and the corresponding granting ($\mathrm{rq}(a, \delta)?[-\delta, -5]$). Such a delay is never mentioned in the natural language description given at the beginning of Section 5.1 of [5], and introduces a major change with respect to the system behavior presented in [3].

In addition, because of the meaning of the **since** operator in TILCO, (3) states that *always*:

> the resource is granted to a process
> $\Leftrightarrow$
> - the resource has always been free until now, or
> - there was a moment in which no process was granted the resource, and the resource has since been free (current instant excluded)
> - and, ...

Therefore, as it is, (3) forces the resource to be granted the first instant, after five time units from the request, in which no process was granted the resource the instant before. For example, the interpretation of Figure 2[2] is a model for (3): process a is granted the resource at time 6 and never releases it; process

[2] In all interpretations presented in this section, every instant is labeled with all the predicates that are true in that instant; predicates that do not appear in the label of an instant are false there.

b requests the resource at time 10 (and also at time 12; this is not against the specification, nor, for that matter, against natural intuition) and obtains it at time 15 (even if a has yet to release it); process c requests the resource at time 11, and is granted it at time 17 (the resource cannot be granted to a process at time 16 since there is already a grant the instant before). This is against the natural language specification and, in general, against intuition of what a resource allocator should do.

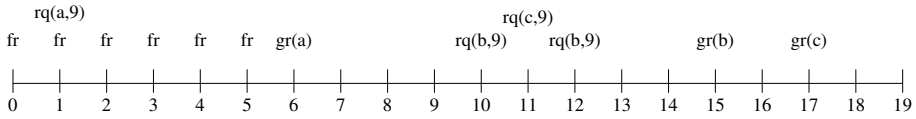

Figure 2: A model for (3).

By comparing the TILCO formula against the TRIO ones, we note that, in the paper, there is some confusion about the order of the arguments to be used with $Since_w$ and $Until_w$. In this respect, TILCO and TRIO use different notations: $\mathbf{since}(A, B)$ in TILCO corresponds to $Since_w(B, A)$ in TRIO. For instance, Table 3 ([5], pg. 220) lists the arguments in the wrong order, while, in (2) and (3), they are sometimes correctly swapped. This confusion is not trivial as far as the meaning of the presented formulas is concerned. In particular, the $\mathbf{since}(\neg\exists b.\mathrm{gr}(b), \mathrm{fr})$ of (3) seems to be basically taken from [3], but keeps the arguments in the TRIO order; therefore, when interpreted in TILCO, it defines an unwanted behavior, as shown in Figure 2.

Therefore, in an attempt to correct (3), let us modify it by swapping the arguments of the first **since** (see (3a)).

$$\left(\forall a.\mathrm{gr}(a) \Leftrightarrow \left(\begin{array}{l} \mathbf{since}(\mathrm{fr}, \neg\exists b.\mathrm{gr}(b)) \wedge \\ \ldots \end{array} \right)\right) @(-\infty, +\infty) \tag{3a}$$

Notice that, now, predicate fr changes its role. In the previous formulation, fr held over intervals when the resource was not taken by any process. Now, fr has basically a point-wise behavior and signals that the resource has *become* free (it is released by the process that held it).

This change fixes the problem of allowing a process to be granted the resource even if the latter has yet to be released; however, the formula is still incorrect, as it accepts a behavior where the resource is granted to a process even if there is another process that requested it earlier. In fact, the interpretation shown in Figure 3 is a model for the new formula. Process b requests the resource before

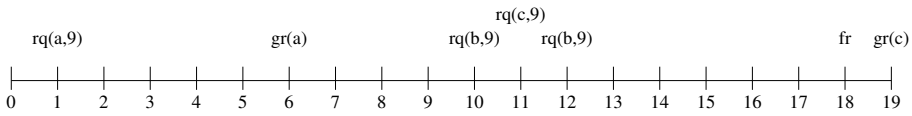

Figure 3: A model for (3a).

process c, but it is c that is granted access, not b. This happens because the last subformula in (3) (i.e., **since**$(\mathrm{rq}(a,\delta), \neg\mathrm{rq}(a',\delta'))$) does not include the case (i.e., it is false in the case) in which b requests the resource twice, once before and once after c requests it, which is precisely the situation of Figure 3.

TRIO formulas. Let us now consider the TRIO formulas for the allocator introduced in [5].

In the TRIO Formula (2) of [5], the first $Since_w$ is correctly employed, so that it does not grant the resource to a process if the resource is not free. On the other hand, its behavior is still incorrect with respect to the natural language specification as it has the interpretation of Figure 3 for a model, too. In addition, even if we consider (3a), (2) and (3a) still have different meaning. In fact, the interpretation of Figure 4 is a model for (3a), but not for (2). This is because (2), as it is, does not permit to grant the resource to a process that repeated its request in the previous five time units.

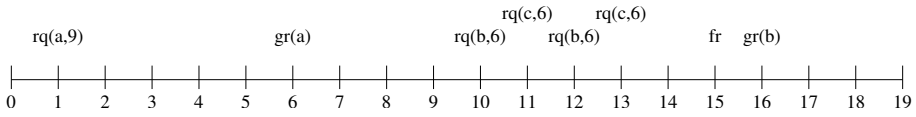

Figure 4: A model for (3a), but not for (2).

Notice that, to have the same (incorrect) meaning of (3), (2) should be modified as in Figure 5, in which case there is very little difference between the two formulas. Finally, let us consider the TRIO formula presented in [5] as (1)[3].

$$
Alw\left(\begin{array}{l}
\forall a.\mathrm{gr}(a) \\
\quad \Leftrightarrow \\
\left(\begin{array}{l}
Since_w(\mathrm{fr}, \neg\exists b.\mathrm{gr}(b)) \wedge \\
\quad \exists\delta.\left(\begin{array}{l}
Past(WithinP_{ii}(\mathrm{rq}(a,\delta), \delta-5), 5) \wedge \\
Since_w(\neg\mathrm{gr}(a), \mathrm{rq}(a,\delta)) \wedge \\
\quad \neg\exists a',\delta'.\left(\begin{array}{l}
a' \neq a \wedge \\
Past(WithinP_{ii}(\mathrm{rq}(a',\delta'), \delta'-5), 5) \wedge \\
Since_w(\neg\mathrm{gr}(a'), \mathrm{rq}(a',\delta')) \wedge \\
Since_w(\neg\mathrm{rq}(a',\delta'), \mathrm{rq}(a,\delta))
\end{array}\right)
\end{array}\right)
\end{array}\right)
\end{array}\right)
$$

Figure 5: A modification of (2) of [5].

It is different from both (2) and (3). Indeed, this formula is very similar to the one presented in [3], but, in addition to introducing the aforementioned delay of five time units between request and grant, it misrepresents the first $Since_w$. In fact, $Since_w(\neg\exists b(\mathrm{gr}(b)), \mathrm{fr})$ in TRIO means that either $\neg\exists b(\mathrm{gr}(b))$ is always

[3]We assume that, in formula $\forall t''(0 < t'' < t \Rightarrow Past(\neg\mathrm{gr}(a), t))$ of (1), subformula $Past(\ldots, t)$ is an obvious typo for $Past(\ldots, t'')$.

true in the past, or *fr* was true in the past, and $\neg\exists b(\mathrm{gr}(b))$ has been true since. This is in sharp contrast with the first subformula in the right-hand side of (1) of [5]:

$$Alw\left(\forall a.\mathrm{gr}(a) \Leftrightarrow \left(\begin{pmatrix}\forall t_3(t_3 > 0 \Rightarrow Past(\mathrm{fr}, t_3)) \vee \\ \exists t_1(t_1 > 0 \wedge Past(\neg\exists b.\mathrm{gr}(b), t_1) \wedge \\ \forall t_2(0 < t_2 < t_1 \Rightarrow Past(\mathrm{fr}, t_2)))\end{pmatrix} \wedge \right)\right)$$

which states instead that $Since_w(\mathrm{fr}, \neg\exists b(\mathrm{gr}(b)))$. As a consequence, (1), much like (3), defines that the resource is granted the first instant after 5 time units from the request, in which no process was granted the resource the instant before, no matter if it has been released or not.

Now, if we modify the first conjunct in (1) to correctly represent $Since_w(\neg\exists b(\mathrm{gr}(b)), \mathrm{fr})$, we finally get what we consider a more reasonable (delay-augmented) specification for the allocator. That is, one that does not allow the resource to be granted if this is not available, nor if there is a process that has a longer pending request. In fact, the interpretation of Figure 3 is not a model for the modified (1), because c cannot be granted the resource before b, while the interpretation of Figure 4 is.

2.2.2 A Simple Example

Section 2.3 of [5] endeavors to formalize the behavior of a signal B which is "periodic with a duty-cycle of 50 percent and a period of 20 time units while, being associated with each transition of B (from false to true), signal A stays true for two time units". For this, they suggest the following formula:

$$\begin{aligned}&(\neg B@(0,10] \Leftrightarrow (B@(10,20] \wedge \neg B@(20,30]))\wedge \\ &\quad((\neg B?[-1,-1] \wedge B) \Leftrightarrow (A@[0,2] \wedge \neg A@(2,20)))\end{aligned}$$

which is incorrect as it admits, for example, the interpretation of Figure 6 as a model.

A more appropriate TILCO specification for the behavior of signals A and B is the formula:

$$\begin{aligned}&((B@[0,10) \wedge \neg B@[10,20))?[0,20))\wedge \\ &\quad((\neg B?[-1,-1] \wedge B) \Rightarrow (A@[0,2] \wedge \neg A@(2,20)))\end{aligned}$$

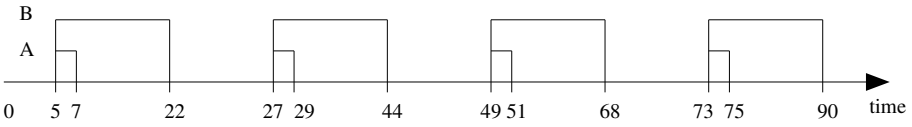

Figure 6: A behavior for signals A and B.

3 Remarks about the Comparison

Let us now consider the general approach used for comparing TILCO with TRIO.

First and most notable, in [5], the *Dist* temporal operator, introduced in this journal in 1994 [2], is not acknowledged. In fact, the authors are aware of it [1], as they briefly remark on it in Section 5, but it is not used in their comparison. *Dist* eliminates any "sharp distinction" between past and future (a purported flaw of TRIO cited in [5]) and, actually, it is the only basic temporal operator in TRIO. This approach, in our opinion, indicates partiality which is evident in Table 3 of [5] (also called "A Comparison between TILCO and TRIO on the Basis of Typical Temporal Specifications."). The table is divided into four columns: the first contains brief natural language tags (e.g., *Lasts*), the second contains very small TILCO formulas — every one a direct application of a single temporal operator, the third contains the corresponding TRIO versions using only "elementary" temporal operators (i.e., as noted in [5], $Futr$ and $Past$); the last one contains general TRIO formulas, possibly including applications of derived temporal operators.

The concept of "elementary operator" used in [5] is unusual. For instance, $Futr$ and $Past$ are both derived operators (e.g., $Futr(A,t) \equiv Dist(A,t) \wedge t \geq 0$), but the same holds for some quantifications and propositional operators.

Table 3 of [5] starts from simple TILCO formulas, to write what are sometimes quite convoluted and rather long equivalent TRIO expressions. For instance, the basic TILCO quantifications over temporal intervals, that is $A@(t_1,t_2)$ and $A?(t_1,t_2)$, correspond respectively to the following TRIO formulas: $\forall t(t_1 < t < t_2 \Rightarrow Dist(A,t))$, and $\exists t(t_1 < t < t_2 \wedge Dist(A,t))$. Note that neither formula nests temporal operators, therefore the objection at the end of Section 5 of [5], that nesting of temporal operators is necessary in TRIO, is incorrect. Instead, the authors of [5] chose to express them, restricted to $t_1, t_2 \geq 0$, as the difficult to read $Futr(\forall t'(0 < t' < t_2 - t_1 \Rightarrow Futr(A,t')),t_1)$ and $Futr(\neg\forall t'(0 < t' < t_2 - t_1 \Rightarrow Futr(\neg A,t')),t_1)$.

As a further example, using the basic *Dist* operator, the derived operator $Alw(A)$ can be defined in TRIO as:

$$Alw(A) \equiv \forall t\ Dist(A,t)$$

which is considerably simpler than the one reported in Table 3 of [5]:

$$Alw(A) \equiv \forall t(t > 0 \Rightarrow Futr(A,t)) \wedge A \wedge \forall t(t > 0 \Rightarrow Past(A,t))$$

and does not appear significantly more complex than the TILCO formula $A@(-\infty,+\infty)$.

Sometimes, previously defined operators are forgotten, e.g., the last line of Table 3 states that $A?(t_1,t_2)$ is equivalent to $Futr(\neg Lasts(\neg A, t_2 - t_1), t_1)$, where the use of $\neg Lasts(\neg A,\ldots)$ instead of the simpler $WithinF(A,\ldots)$ is difficult to understand, given that it is used a few lines before.

Moreover, in Table 3 the arguments of $Since_w$ and $Until_w$ are erroneously swapped. Also, the choice of using weak operators is by itself TILCO-biased: the weak *since* and *until* are an atypical choice for basic operators, and in most languages they are derived from the strong ones. On the other hand, to state $Until(A, B)$ (strong until) in TILCO, one must use two temporal operators, such as $\mathbf{until}(B, A) \wedge B?(0, +\infty)$. Unfortunately, in [5], these commonly encountered operators were not cited.

In addition, it seems that a similar treatment was applied to MTL [4], although limited to the comparison of simple expressions. In fact, while we do not claim expertise in the MTL language, we cannot help but notice that Table 4 of [5] avoids to use the MTL **until** and **since** operators, replacing them with more complex expressions that use explicit temporal quantification, such as $\exists t(t \gg 0 \ \wedge \ \mathbf{F}_t B \ \wedge \ \mathbf{G}_{<t} A)$ for $\mathbf{until}(A, B)$.

Concerning the comparison between TRIO and TILCO carried out in Section 5.1 of [5], besides the technical errors reported in Section 2 of this paper, we note that it is pointless to consider (1) of [5] as a reference for evaluating the readability of the TRIO specification of the real-time allocator example, because (1) contains only "basic" operators, except for the outermost Alw: the basic operators are chosen to pursue minimality of definitions and economy of reasoning in the proofs of meta-theorems of the logic, not to obtain conciseness or readability in the specifications.

Before closing this section let us note that one objective way of classifying formal languages is by comparing their expressiveness (i.e., the possibility of the language to express extensive classes of properties).

In this regard, TILCO is clearly less expressive than TRIO (as explained in Section 2.1); TILCO's authors seem to be unaware of this: in an effort to make it more concise and readable the language was not endowed with explicit quantification over the time variable, but, in [5], the authors use temporal quantification when necessary to express complex properties without even acknowledging the violation of the very syntax of the language.

4 Conclusions

In this paper we have drawn the readers' attention to the errors, omissions and imprecisions that undermine the comparison between TILCO and TRIO carried out in [5].

In [5], a real-time allocator example was chosen as a benchmark on which to compare conciseness and readability of the two languages, but the authors did not realize that the real-time allocator cannot be specified in TILCO because this logic does not allow explicit quantification over time. Therefore TILCO is less expressive than TRIO.

Even considering a hypothetical extension of the TILCO language that includes explicit temporal quantification, we showed that in the analysis of the

formalizations of the real-time allocator in TILCO and TRIO the authors of [5] made several errors.

In the comparison of the two languages concerning brevity and simplicity in the definition of derived temporal operators, [5] failed to note that the TRIO language is based on the *Dist* operator, which makes most definitions much more compact than those reported in the comparative tables of [5]. The *Dist* operator appeared in the open literature [2] two years before the submission of [5] and four years before its acceptance; the authors of [5] were aware of its existence [1], as they briefly mentioned it, but they did not use it.

Our discussion, therefore, shows that the claim that TILCO is more concise and readable than TRIO is both questionable (because it compares TILCO formulas with redundant and needlessly long TRIO formulas) and irrelevant (because TILCO is less expressive than TRIO). This does not imply that any language is "better" than the other, but it simply reminds the reader that a valid comparison must be based on technically sound arguments.

References

[1] Giacomo Bucci, Maurizio Campanai, and Paolo Nesi. Tools for specifying real-time systems. *Real-Time Systems*, 8(2–3):117–172, 1995.

[2] Miguel Felder, Dino Mandrioli, and Angelo Morzenti. Proving properties of real-time systems through logical specifications and Petri net models. *IEEE Transactions on Software Engineering*, 20(2):127–141, February 1994.

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

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

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