USING COUNTERS TO MODEL TEMPORAL RELATIONSHIPS AMONG EVENTS

ANGELO GARGANTINI, ANGELO MORZENTI, AND ELVINIA RICCOBENE

Abstract. In this paper we propose a formalization using state variables like counters to model temporal relationships among events. We prove that the proposed method is correct. We show that counters are more suitable to prove properties and to implement events relationships by means of a programming language.

1. Introduction

Many widely adopted approaches to modeling and specification are based on notations centered on ontological entities like states and/or events. It is a basic fact of the theory of state/transition systems that the formal models based on states and those based on events are to a large extent equivalent, so the question may arise whether it is more convenient to adopt either one of them, or both, when developing computer-based applications. From the standpoint of software engineering, hence considering also matters related to organization, administration, and practical convenience, the concrete choice of the engineers is most times dictated by the ease of use of a notation (a combination of expressiveness, readability, writ ability) and (we point out a bit provocatively) by the actual availability of commercial- and industrial-strength tools that are well engineered, technically and commercially supported, and integrated with the most common hardware/software platforms.

If we focus on the most conceptual aspects, we note that the use of states and events is most useful in the modeling and analysis of reactive, embedded, real-time systems, i.e., those systems that closely interact with an environment that they have to monitor or control. The notations supporting modeling and analysis are most relevant in the highest phases of development: for the purpose of design and implementation the programming languages have reached a level of abstraction and generality that makes them adequate and sufficient.

In the phases of user requirements identification, of design requirements specification, and of System Requirements Analysis (the activity whereby one wishes to establish that the adopted hypotheses on the environment, combined with the envisaged solutions, satisfy the user requirements) the chosen notation and its style (operational versus descriptive, oriented towards properties or towards state/transitions/events/ actions) can influence dramatically the cost effectiveness of the development effort, and can favor or hinder the requirements elicitation, the communication among developers and between developers and users, the ability to reason

Angelo Gargantini - University of Catania, gargantini@dmi.unict.it and Angelo Morzenti - Politecnico di Milano, morzenti@elet.polimi.it and Elvinia Riccobene - University of Catania, riccobene@dmi.unict.it.

on the models and the solutions to prove, at various levels of formality, their correctness and adequacy.

In the above described scenario (which, we note, is mostly influenced by methodological and pragmatic issues, rather than by strictly scientific and foundational ones) the requirements of minimality and essentiality of the adopted formal notation (e.g., questions such as: can I do without events and use only states, or vice versa) become much less relevant. In fact, one can obtain great benefits by combining the notions of state and event, even at the cost of some redundancy, but aiming at producing models and descriptions that are more compact and intuitive. From a pragmatic point of view, it is also important that the adopted notation and style be able to embody some psychological or even philosophical aspects of the developers' mental habits such as: the tendency to describe dynamic systems (and reactive systems among these) in terms of cause-effect relations; the identification of certain necessary or sufficient conditions/events/states in order for other ones to hold; the great ease, for most people, to reason in visual or geometric terms, which favors the adoption of certain conceptual entities that can be represented in a graphical fashion (notice that this can occur for both states and events, as it will be illustrated in the following).

Hence, for what concerns the questions raised by the ST.EVE workshop, our position is based on a quite eclectic and pragmatic attitude: both the notion of state and that of event have virtues and limitations, so that the worst thing to do would be to reject one in favor of the other; the best approach, in our opinion, is to make a constant effort to use them in combination, since most likely this will foster the emergence of new, complementary and synergetic, points of view and ways of describing and reasoning.

The dispute among supporters of state-based and event-based models and analysis is somehow reminiscent of another contention, the one between descriptive notations (i.e., logic-based, best suited for expressing properties and relations) versus operational ones (based on state/transitions and hence more oriented towards describing actions and execution steps of (abstract) machines). This contention, like the one between states and events, is best settled by adopting the so-called dual-language approach, which combines in a synergetic way a descriptive notation (e.g., temporal logic) and an operational one (e.g., Petri nets), thus overcoming the limitations and maintaining the advantages of both; we have applied and illustrated the dual-language approach in [6, 7]

Coming back to the main topic of the workshop, in the present work we will illustrate our position in favor of the combined use of states and events by presenting some new developments of our previous work appeared in [8, 9], where we used the language TRIO (a very general and expressive temporal logic) to characterize axiomatically the various kinds of modeling and specification entities that can be employed in the System Requirements Analysis of reactive, time-critical systems. We formalized in terms of TRIO axioms the notions of time point- and time interval-based predicates and variables (which have a close correspondence with events and states, respectively), of (non)Zeno entities, and a quite rich variety of time and causal (either deterministic or not) relationships among these entities. Here we will provide new elaborations on the use of special, interval-based, integer-valued variables that count the number of event occurrences; such counter variables turn out to be a quite simple and effective means to express and to prove time and

causal relations and properties that would be quite cumbersome and intricate to characterize in terms of a specification alphabet composed of state-like items only or event-like items only.

The paper is organized as follows. Section 2 briefly presents TRIO, events and counters and reports the related work. In Section 3 we explain the most basic relationship between events we model, namely the cause-effect relationship between two event with bounded delay. In Section 4 we extend the same approach dealing with non deterministic choice between events (with null delay) and in Section 4 we model the most general case with non deterministic choice and non deterministic delay. In Section 6 we discuss the advantages using counters providing some examples.

2. Basics

TRIO. We will use TRIO [10, 15] to express many temporal dependencies and relations. TRIO is a first order logic augmented with temporal operators that allow to express properties whose truth value may change over time and to model variables whose value changes over time. The meaning of a TRIO formula is not absolute, but is given with respect to a current time instant which is left implicit. The basic temporal operator is called *Dist*: for a given formula *W*, *Dist*(*W*, *t*) means that *W* is true at a time instant whose distance is exactly *t* time units from the current instant, i.e., the instant when the sentence is claimed. Many other temporal operators can be derived from *Dist*.

Futr(F, d)	$d \geq 0 \wedge Dist(F, d)$	future
Past(F, d)	$d \geq 0 \wedge Dist(F, -d)$	past
NowOn(F)	$\exists d(d > 0 \wedge Lasts(F, d))$	F holds for an interval after now
Alw(F)	$\forall d\, Dist(F, d)$	F always holds
AlwP(F)	$\forall d(d > 0 \rightarrow Past(F, d))$	F always held in the past
Som(A)	$\exists d Dist(F, d)$	Sometimes F held or will hold

Besides temporal dependent (TD) predicates, TRIO introduces temporal dependent variables with domain *D*, as variables whose value changes in *D* over time. For instance, TD variables are suitable to model enumerate variables and continuously changing variables, as many physical quantities. To refer to values of a variable or term in the past or in the future, the operator *dist* (as generalization of *Dist*) is introduced: for a given term *x*, *dist*(*x*,*t*) has the value that *x* had or will have at a time instant whose distance is t from now. From the *dist* operator, the *futr* and *past* operators are derived, referring to values of variable sin the future and in the past.

Note that in the following, when stating formulas and properties that always hold, we omit an outermost *Alw:* stating such properties in a generic instant of time we mean that they hold forever.

Events and Counters. In our framework, events are predicates that are true only in single time instants and have null duration. A more detailed discussion and a rigorous treatment of events and state (or interval) variables can be found in [9]. Events can be distinguished but event occurrences of the same event cannot be uniquely identified (except for the time they occur) since they do not carry

other information, otherwise other axiomatizations are suitable to model temporal relationships among events [11]; this point will be clear in Section 3. We admit simultaneous occurrences of the same event. We use the notation A, B to indicate events. For events we will use counters: for the event A we call $\#A$ the counter of the event A. $\#A$ in each instant has an integer value indicating the number of occurrences of A till the current instant. Counters are state variables, i.e. they hold their value for temporal intervals and not just for time points.

We assume here that, for any event E, there exists a first occurrence, i.e., that there is an instant before which E never occurred, a fact that is formalized in TRIO as $Som(AlwP(\neg E))$. Therefore, there exists a initial instant when all the counters are null. This hypothesis is quite realistic for real-world systems. A less restrictive assumption is however adopted in [8] in the proof of the theorem 1, showing that it is immaterial from a mathematical viewpoint.

For the sake of simplicity, we assume event counters being incremented just after the time instant when the counted event happens, i.e. we assume counters to be left continuous (for details see [9]). For this reason, the following axiom holds (x is an integer): $B \wedge \#B = x \rightarrow NowOn(\#B = x + 1)$

Related works. There exist several approaches that try to combine concepts from state based methods with concepts from event based notations. In Chapter 18 of [5] the authors discuss how to integrate Z and Object-Z with process algebraic methods like CSP and CCS. In [3] some abstract concepts of process algebra and from ASMs are integrated into Abstract State Processes (ASPs), i.e. evolving processes (extended ASM programs which are structured and evolve like process-algebraic behavior expressions) operating on evolving abstract states the way traditional ASM rules do. This paper tackle a more particular concept but in a general way.

The idea of using counters to model relations between events was already presented in [4]. [4] defines events like functions from N to time: for the event e, $e(i)$ is the time of the i-th occurrence of the event e. It defines also counters of events. In that paper constructs to express *precedence* (that is an event or a set of events must precede or follow another event) between two events (or an event and a set of events) are presented and these constructs use counters. Also [16] introduce the counter $\#e$ for each event e. An example is provided where counters are used to specify properties of the system, in a similar fashion proposed by our method, but no formal framework is given. Indeed, counters are directly used to specify system requirements (like, for instance, "the number of missiles fired is no more than the number of targets located so far"). The generic requirement that a certain event must follow (or precede) another can be enriched specifying a bounded delay between such events or requiring that an event can cause another event non deterministically chosen among a set of possible effects. This kind of requirement is widely (also informally) used. Furthermore it is explicitly modeled (somehow embedded within the language itself) in Timed Petri Net (TPN) [13] and in MMT timed Automata [14]. We shown how the use of counters can be extended to specify this relation in a very simple and effective way.

3. Cause-Effect Relationship Between Two Events with Non Deterministic Delay

The first relationship we tackle in this paper is the relationship involving only two events: an event A causes another event B, in a future time that is not known

precisely, due to some nondeterminism of the system. Typically, the delay between A and B is characterized by means of a lower bound and an upper bound denoting the minimal and maximal time distance between related occurrences of the two events. The relation is therefore nondeterministic, being the exact instant in the future, when B will occur after A, unknown. Moreover, we assume that the relation is one-to-one (for instance because A is the unique cause of B: every occurrence of A causes one B and every occurrence of B is caused by one occurrence of A). An example of this kind in a concurrent systems could be the relation between the event of taking a resource and that of returning it.

The preliminary following formal definition of this kind of relation follows.

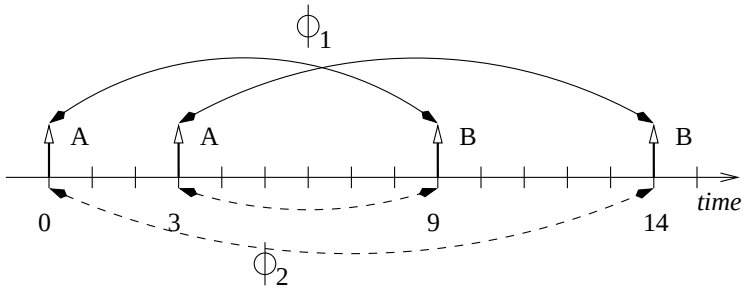
Definition 1. *An event A is a unique cause for an event B in [d,D] time units, where d and D are positive real constants such that $d \leq D$, iff there exists a one-to-one function ϕ from the occurrence times of A to those of B such that $t + d \leq \phi(t) \leq t + D$*

This definition, originally presented in [9], is not suitable to model events that can occur several times in the same time instant. Indeed, if the event A occurs at the time instant t two times, a function ϕ would not suffice to express the relation between occurrences of A and occurrence of B, since the two B events related with the two occurrences of A could occur in distinct time instants, while $\phi(t)$ has an unique value. To correctly model the relationship, we have to extend the definition of the function ϕ, such that ϕ counts the simultaneous occurrences of the same event.

Definition 2. *An event A is a unique cause for an event B in [d,D] time units, where d and D are positive real constants such that $d \leq D$, iff there exists a one-to-one function ϕ from Time $\times$ N to Time $\times$ N such that $\phi(t_A, i_A) = t_B, i_B$ iff A happens at least i_A times at the time instant t_A and B happens at least i_B times at the time instant t_B , and $t_A + d \leq t_B \leq t_A + D$*

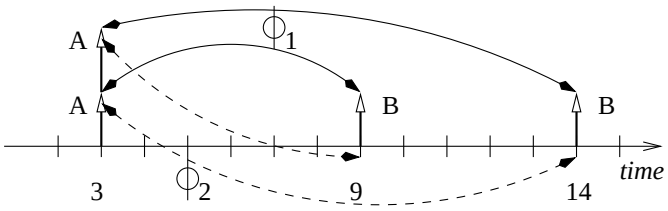
The function ϕ relates the i_A-th occurrence of A at the time instant t_A with the i_B-th occurrence of B at the time instant t_B.

Note that the fact that event occurrences are indistinguishable (except for the time they occur) makes the function generally non unique. For example if A is unique cause of B, $d = 8$, and $D = 14$, A occurs at time 0 and 3 and B occurs at time 9 and 14, we could have to possible functions ϕ_1 and ϕ_2 as depicted in the following figure.

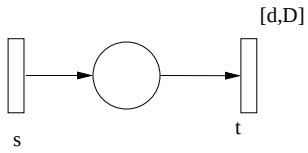


The same observation can be extended to the case of simultaneous occurrences. With simultaneous occurrences, the function ϕ always admits multiple solutions,

since we cannot distinguish two simultaneous occurrences of the same event. This case is depicted in the following figure, where event A happens two times at 3 and each occurrence can be related with B at time 9 or with B at time 14.



This kind of relationship (together with the extensions we present in the following sections) is widely used and in several formalisms it is the only temporal relationship between events: see for instance timed Automata in [2, 14] and Timed Petri Net (TPN) in [13]. In TPN, it is pictured as follows (through the paper we often use TPNs to graphically represent the relationship we intend to deal with). We will designate events also using name of transitions in TPNs, with lower case letters like s, r, t, ...



Definition 2 formally states the relationship between events we deal with (and it guarantees to avoid misunderstandings), yet it is apparent that it can be barely useful during analysis or synthesis of real systems. To be useful, the definition can be formulated using TRIO axioms as reported in [9]. However, note that that simple and natural formalizations using only temporal formulas (that is not using predicates) are completely wrong (as shown by [8]). Correct solutions using several additional predicates can be found in [7] and in [12].

In the following we demonstrate that the proposed approach using counters is correct, simpler and more useful.

3.1. **Formalizing Definition 2 Using Event Counters.** We now introduce a way to bind events A and B using the counters of their occurrences, respectively denoted as $\#A$ and $\#B$.

By means of a loose and informal reasoning we first find a suitable relation modeling the cause-effect binding.

First of all, notice that, for any event E, at any time the number of occurrences of E in the interval at distance $[d_1,d_2)$ (i.e., the interval starting d_1 time units in the future and ending d_2 time units in the future [1]) is equal to the total number of events occurred before the end of the interval minus the total number of events occurred before the beginning of the interval: this difference can be is denoted as $dist(\#E, d_2) - dist(\#E, d_1)$.

[1]The right end of the interval is not included because counters are assumed to be left continuous.

The first concept we would like to express is: if in a interval [d_1,d_2) in the future there are x occurrences of event A, then in a interval [d_1+d,d_2+D) there are at least x occurrences of B:

$$dist(\#A, d_2) - dist(\#A, d_1) \leq dist(\#B, d_2 + D) - dist(\#B, d_1 + d) \quad (1)$$

Vice versa if in an interval [d_1,d_2) the future there are x occurrences of B, then in a interval [d_1-D,d_2-d) there are at least x occurrences of A:

$$dist(\#B, d_2) - dist(\#B, d_1) \leq dist(\#A, d_2 - d) - dist(\#A, d_1 - D) \quad (2)$$

We would like to derive, from (1) and (2), a simpler, unique, and equivalent relation. First we can rewrite (1) with $d_1 = d_1$-d and $d_2 = d_2$-D:

$$dist(\#A, d_2 - D) - dist(\#A, d_1 - d) \leq dist(\#B, d_2) - dist(\#B, d_1) \quad (3)$$

From (2) and (3) we find a unique relation:

$$\begin{aligned} dist(\#A, d_2 - D) - dist(\#A, d_1 - d) &\leq dist(\#B, d_2) - dist(\#B, d_1) \\ &\leq dist(\#A, d_2 - d) - dist(\#A, d_1 - D) \quad (4) \end{aligned}$$

Taking some license in notation, we choose $d_1 = -\infty$[2] and $d_2 = 0$:

$$\begin{aligned} dist(\#A, -D) - dist(\#A, -\infty - d) &\leq \#B - dist(\#B, -\infty) \\ &\leq dist(\#A, -d) - dist(\#A, -\infty - D) \end{aligned}$$

Since $dist(\#A,-\infty - d) = dist(\#B,-\infty) = dist(\#A,-\infty\text{-}D) = 0$ we finally obtain:

$$dist(\#A, -D) \leq \#B \leq dist(\#A, -d) \quad (5)$$

We have found a relation between counters (formalized as a TRIO axiom) suitable to model the relation given in definition 2. This fact is stated in the following theorem.

Theorem 1. *Event A is a unique cause of event B in [d,D] time iff:*

$$past(\#A, D) \leq \#B \leq past(\#A, d)$$

The preliminary version of the proof of this theorem can be found in [8].

Intuitively, if the relation of Definition 2 holds, when event A occurs, causing an increment in counter $\#A$, then counter $\#B$ is also bound to increase; however, due to the assumed delay ranging between d and D, counter $\#B$ will increase no earlier than d time units after the increase of $\#A$, hence the inequality $\#B \leq past(\#A, d)$ holds; moreover, and symmetrically, $\#B$ will increase no later than D time units after $\#A$, hence $\#B \geq past(\#A,D)$ holds.

Theorem 1 expresses the concept stated in Definition 2 with very simple relations between event counters. Thanks to their simplicity (they are just linear inequalities) these relation can be used very easily and effectively in the derivation of relevant properties as shown by the following corollary.

Corollary 1. $B \rightarrow \exists t(d \leq t \leq D \wedge Past(A, t))$

[2]By choosing $d_1 = -\infty$ we consider an instant in the past such that neither predicate A nor$B$ ever occurred before that time, and therefore their counters are both zero; such an instant exists, thanks to our previously mentioned assumption that *Som(AlwP(¬A))* and *Som(AlwP(¬B))*.

Proof. if B occurs, its counter increases by one (counter are left continuous):
$B \rightarrow \exists x(\#B = x \wedge NowOn(\#B = x + 1))$
applying the relation between counters:
$B \rightarrow \exists x(\#B = x \wedge Past(\#A \leq x, D) \wedge NowOn(\#B{=}x{+}1 \wedge Past(\ x{+}1 \leq \#A, d)))$
because of $NowOn(A \wedge B) \leftrightarrow NowOn(A) \wedge NowOn(B)$:
$B \rightarrow \exists x(Past(\#A \leq x, D) \wedge NowOn(Past(\ x{+}1 \leq \#A, d)))$
thanks to the equality $NowOn(Past(A,t)){=}Past(NowOn(A),t)$
$B \rightarrow \exists x(Past(\#A \leq x, D) \wedge Past(NowOn(x{+}1 \leq \#A), d)))$
From this , it can be proved that there is an occurrence of A:
$B \rightarrow \exists t(Past(A, t) \wedge d \leq t \leq D)$ □

Particular cases and generalizations. The model can be both generalized and applied to particular cases. For instance, if the minimum delay d is zero (event B can follow immediately event A) the relation becomes: $past(\#A,D) \leq \#B \leq \#A$.

If the delay has no upper bound, then $D = \infty$ and the relation reduces to: $\#B \leq past(\#A,d)$.

A further, interesting generalization of the one-to-one relation introduced in Definition 2 is to allow a negative minimum time d. In this most general case one would not model a "cause-effect" relation, but a correspondence between event occurrences that are somehow related, for instance because they are both effect of a common (unique) cause. Theorem 1 can be generalized in a straightforward way to this case by just changing the *past* operators (which assume a positive time argument and necessarily refer to previous instants) into *dist* operators (which equally admit a positive, null, or negative time argument, thus referring to both past, present and future), obtaining the following relation

$$dist(\#B, d) \leq \#A \leq dist(\#B, D)$$

which holds under the unique assumption that $d \leq D$.

As a concrete example, let us consider an electronic trading system where an order for some goods performed by a client gives rise subsequently, through independent chains of actions, to the physical delivery at the client's address of the parcel containing the ordered item, and to the billing of the price on the client's bank account. An important property of the trading system could be that there is a one-to-one matching between goods delivery and bank account transactions. These two events are clearly related, but there might be no strict temporal precedence between them. If we model goods delivery by the event predicate GD and bank account transactions by event predicate BAT, then we can abstractly specify that each occurrence of GD may at most precede the corresponding occurrence of BAT by 3 days, or at most follow it by 4 days, using the following inequalities

$$dist(\#GD, -3) \leq \#BAT \leq dist(\#GD, 4)$$

Example 1. In the Generalized Rail Cross (GRC) case study (see [9]) a one-to-one temporal relationship obviously exists between the entering of trains in the various regions surrounding the crossing. The system is nondeterministic due to the uncertainty about the trains speed, which may vary between minimum and maximum allowed values.

The informal specification asserts that the trains take a minimum time d_m and a maximum time d_M to go from the beginning of region R to the beginning of region

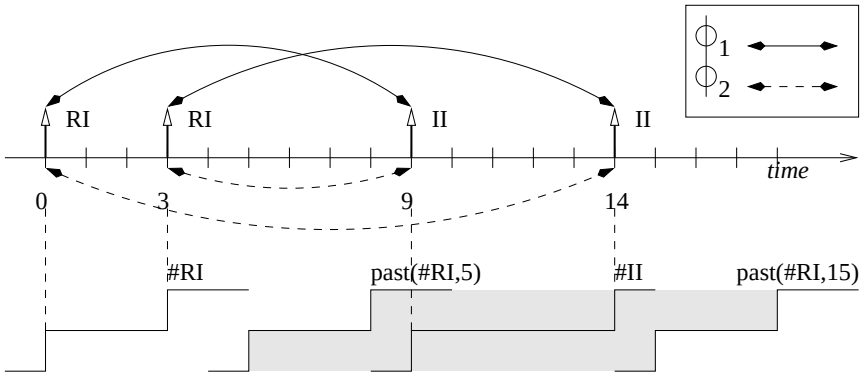


FIGURE 1. . Models for axiomatizations based on special predicates or on counters.

R; it takes a minimum time h_m and a maximum time h_M to go from the beginning of region I to its end.

These relations are formalized by the following inequalities between counters of event occurrences (recall that RI, II, and IO are defined, respectively, as the events of a train entering region R, entering region I, and exiting region I, and that they are multiple events).

$$past(\#RI, d_M) \leq \#II \leq past(\#RI, d_m)$$

$$past(\#II, h_M) \leq \#IO \leq past(\#II, h_m)$$

♢

Model for Function $\phi$and Counters. Note that the relation between counters does not establish the exact mapping between the various occurrences of the cause and effect events. It defines just the values of the counters as events occur in time, without providing any information about the "physical" matching between event occurrences. Of course, the simpler but slightly less informative description in terms of counters suffices in the cases where the exact matching between the event occurrences is immaterial with respect to the desired system properties.

As an illustration of this, consider again the GRC example, with a plant where d_m =5 and d_M =15. Suppose there are two occurrences of RI at times 0 and 3 (i.e., a train enters region R at time 0 and a second train enters at time 3), and two occurrences of event II at times 9 and 14 (i.e., a train enters region I at time 9 and another one at time 14). From a physical viewpoint there are two interpretations of this event sequences: either the trains enter in region I in the same order as they entered in region R, or the train that entered region R last passes the first one, and enters region I before it.

Correspondingly, there are two models (i.e. two possible ϕ functions) with these event occurrences: in each model the relation shows which event of the kind "entrance in region I" corresponds to each event of the kind "entrance in region R", as shown in Figure 1. When we use event counters, instead, we model the fact that "sensors do not recognize trains", so that there is just one possible model, shown in Figure 1, accounting for the total number, up to any given time, of occurrences

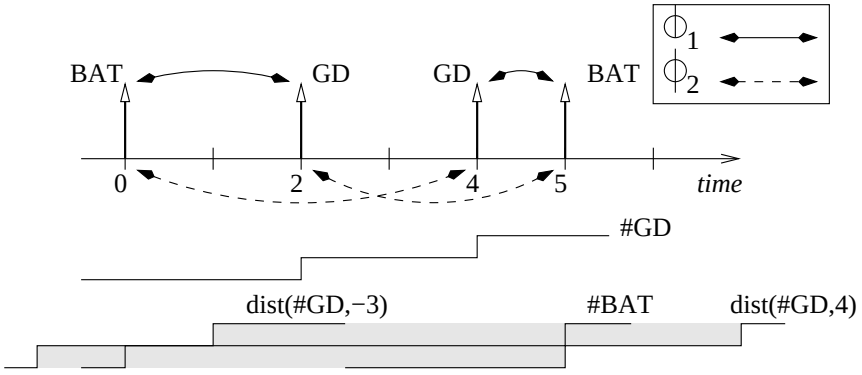


FIGURE 2. BAT and GD

of events of kind "entrance in region R" and "entrance in region I". Notice, however, that the second, less precise description is perfectly adequate to the purpose of governing the Railway Crossing: the safety system does not need to "recognize trains": it can limit itself to counting them.

As an example where the exact matching between event occurrences can be relevant to the desired system properties, consider again the above described electronic trading system, with a sample "history", shown in Figure 2, where two bank account transactions take place at time 0 and 5, and goods delivery occur at time 2 and 4. Figure 2 shows that there is only one model based on counters #GD and #BAT, and there are two models based on the function ϕ.

(Notice that the specification of the electronic trading system can be further enriched by associating to every goods delivery and bank account transaction the description of the acquired item; in this case it can be easily verified that the number of candidate models increases, and a few additional simple axioms are needed to state the property that related BAT and GD occurrences must refer to the same acquired item.)

4. Non Deterministic Choice Between Immediate Effects

The use of counters can be extended to deal with non deterministic choice between two or more events that are immediate (i.e. with no delay) effects of an unique event cause. Let an event A be immediate cause of two concurrent events *B1* and *B2*.

Definition 3. *An event A is the unique immediate cause of two concurrent events B1 and B2 iff there exists a one-to-one function ϕ from Time $\times$ N to Time $\times$ N $\times$ {B1,B2} such that $\phi(t_A, i_A) = t_A, i_e, e$ iff A happens at least i_A times at the time instant t_A and the event e happens at least i_e times at the time instant t_A*

In this case the function ϕ relates each occurrence of A with one occurrence of B_1 or one occurrence of B_2. This situation can modeled by the simple Petri Net shown in Figure 3.

Using counters, we can immediately express the temporal relationship among events s, t_1 and t_2 as follows.

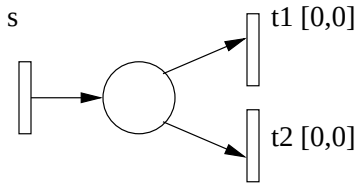


FIGURE 3. Non deterministic choice between two events

Theorem 2. *An event s is the unique immediate cause of two concurrent events t_1 and t_2 iff*

$$\#t_1 + \#t_2 = \#s$$

When s fires once, it increases its counter by one, and one and only one counter $\#t_1$ or $\#t_2$ is bound to increase, i.e. one transition between t_1 and t_2 must fire. The choice between t_1 and t_2 is not deterministic. If s fires more than once, t_1 and $t_2$must fire such that the sum of their firings is equal to the number of firings of transition s. In case of simultaneous firings of s, this equation admits many solutions, and, therefore, it is (as expected) not deterministic.

Starting from the equation above, we can prove many useful relations between the number of firings of t_1 and the number of firing of t_2. For example, it is immediate proving that if t_1 never fires, whenever s fires, t_2 must fire and that, in this case, the number of firings of s and t_2 is equal. Indeed if t_1 never fires, then $\#t_1 = 0$, and $\#t_2 = \#s$. If s fires, its counter increases and the counter of t_2 is bound to increase and the two counters will always hold the same value.

5. Non Deterministic Temporal Choice Between Multiple Effects

In this section, we want to model, using counters, the complex case when an event A is cause of one event among a possible set of events with a delay that is not fixed. In this case the non determinism regards both the delay (as in Section 3) and the choice between two or more effects (as in Section 4).

First, we try to model this kind of relationship by means of a function. Let an event A be cause of two concurrent events B_1 and B_2 with bounded delay. We can introduce the following definition.

Definition 4. *An event A is an unique cause of two concurrent events B_1 and B_2 in, respectively, $[d_{B_1}, D_{B_1}]$ and $[d_{B_2}, D_{B_2}]$ time units iff there exists a one-to-one function ϕ from Time $\times$ N to Time $\times$ N $\times$ {B_1, B_2} such that $\phi(t_A, i_A) = t_e, i_e, e$ iff A happens at least i_A times at the time instant t_A and the event e happens at least i_e times at the time instant t_e and $t_A + d_e \leq t_e \leq t_A + D_e$*

This situation can be modeled by the simple Timed Petri Net shown in Figure 4 (a). We assume that there is *real* conflict between the two transitions, i.e. $l_1 \leq u_2$ and $l_2 \leq u_1$.

To model this case using counters, we can transform the TPN shown in Figure 4 (a) to the equivalent TPN shown in Figure 4 (b). Now we can model this new equivalent TPN by means of the following relations between event counters, exploiting Theorems 1 and 2:

(1) $\#r_1 + \#r_2 = \#s$

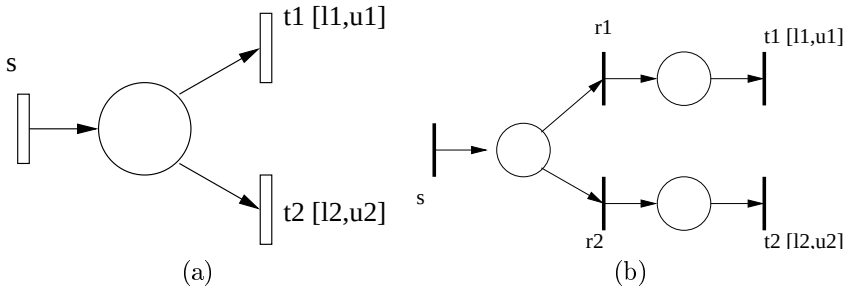


FIGURE 4. Non deterministic temporal choice between multiple effects

(2) $past(\#r_1, u_1) \leq \#t_1 \leq past(\#r_1, l_1)$
(3) $past(\#r_2, u_2) \leq \#t_2 \leq past(\#r_2, l_2)$

Now we want to translate the relation above in equivalent relations but without the fictitious events $r_1$and r_2. Since relations (2) and (3) above are always true, we can temporally translate them by l_1, u_1,l_2, u_2, to obtain the following 4 relations:

(2') $\#r_1 \leq futr(\#t_1, u_1)$ and (2") $futr(\#t_1, l_1) \leq \#r_1$
(3') $\#r_2 \leq futr(\#t_2, u_2)$ and (3") $futr(\#t_2, l_2) \leq \#r_2$

Combining (2'), (2"), (3') and (3"), we obtain the following 4 relations without any reference to $r_1$and r_2

(1) $futr(\#t_1, u_1) \geq \#s - \#r_2 \geq \#s - futr(\#t_2, u_2)$
(2) $futr(\#t_1, l_1) \leq \#s - \#r_2 \leq \#s - futr(\#t_2, l_2)$
(3) $futr(\#t_2, u_2) \geq \#s - futr(\#t_1, u_1)$
(4) $futr(\#t_2, l_2) \leq \#s - futr(\#t_1, l_1)$

The intuitive meaning of such relations is that if s fires, then its counter is increased and t_1 must fire in a time instant in the future between l_1(equation 2) and u_2(equation 1), unless a firing of t_2 has consumed already the firing of s (the minus sign in 1 and 2). Note that (1) is equivalent to (3) and (2) is equivalent to (4). We can now formulate the following theorem.

Theorem 3. *An event s is an unique cause of two concurrent events t_1 and t_2 in, respectively, $[l_1, u_1]$ and $[l_2, u_2]$ time units iff*

$$futr(\#t_1, l_1) + futr(\#t_2, l_2) \leq \#s \leq futr(\#t_1, u_1) + futr(\#t_2, u_2)$$

The meaning of this theorem is that the number of firings of t_1 and t_2 after (before) the two upperbounds (lowerbounds) have elapsed must be greater (smaller) than the number of firings of s.

Example 2. Thanks to the relations above, we can prove some interesting properties. For example we can prove that if t_2 never fires, then t_1 fires as many times as s. If we assume that initially the counter of t_2 is equal to 0 (without loss of generality), then $\#t_2 = 0$ is always true. The relations become: $futr(\#t_1, u_1) \geq \#s$ and $futr(\#t_1, l_1) \leq \#s$, i.e. t_1 is unique effect of s in $[l_1, u_1]$ time units.

Example 3. An ordering and delivering system is specified as follows. When a good (whose identity is not important) is ordered, then it can be shipped using the normal postal service in 0 to 5 days. If the item is not shipped after 3 days, then

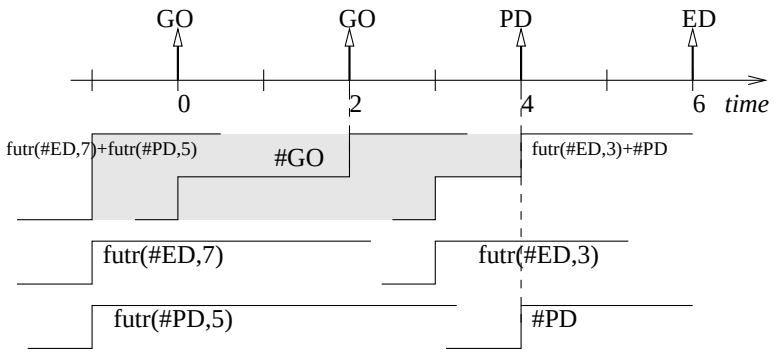
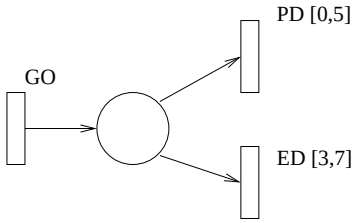


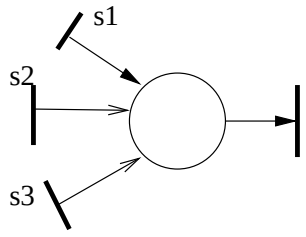
FIGURE 5. A scenario for the ordering-delivering system

the shipping department can use an express currier, that, in any case, must deliver the ordered good in 7 days from the date of the order. The event of ordering a good GO is the unique cause of two concurrent events PD (postal delivery) and ED (express delivery).



Applying Theorem 3 we obtain:
$futr(\#PD, 5) + futr(\#ED, 7) \geq \#GO$ and $\#PD + futr(\#ED, 3) \leq \#GO$
Figure 5 depicts a scenario where an item is ordered at time 0 and another one at time 2, and an item is delivered by postal service at time 4 and another item is delivered at time 6. You can see that the exact mapping between orders and delivering events is not modeled, and GO could happens in all the gray area.

5.1. **Generalization to multiple causes.** The proposed framework can be easily extended to deal with the case of multiple causes (s1, s2, s3, ...). This case is depicted in the following TPN. To model this, one must substitute in the relation presented before, the counter of s, #s, with the sum of the counters #s1, #s2, ...

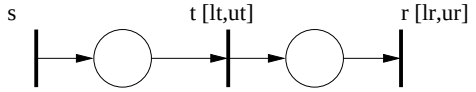


6. ADVANTAGES OF USING COUNTERS

We argue that counters and the proposed relations among them constitute a viable means for proving properties of systems and for implementing such relations

in real programming languages. Indeed, functions ϕ and their properties as proposed in definitions 2, 3, and4 (as well as predicates introduced in [12]), are difficult to handle, especially in performing proofs and in implementing them by means of real programming language. It is intuitive that counters, temporal translation of them, and linear inequalities between them are more manageable. In this Section we provide evidence of this fact by means of some examples.

Example 4. Consider the following example, where an event s causes an event t with temporal bounds l_t and u_t, and t causes an event r with temporal bounds l_r and u_r



The proof that s causes an event r with temporal bounds $l_t + l_r$ and $u_t + u_r$ is immediate. From the relations for counter $\#t$ and $\#r$

$$past(\#s, l_t) \leq \#t \leq past(\#s, u_t)$$
$$past(\#t, l_r) \leq \#r \leq past(\#t, u_r)$$

we can derive the following relation:

$$past(\#s, l_t + l_r) \leq \#r \leq past(\#s, u_t + u_r)$$

Example 5. For the ordering-delivering system presented in Example 3, we want to prove that all the ordered items are shipped after 7 time units:

$$\#PD + \#ED \geq past(\#GO, 7)$$

Proof. Exploiting theorem 3, translating of 7 time units in the past, we can state that $\#ED + past(\#PD, 2) \geq past(\#GO, 7)$. Thanks to the counter property stating that a counter can only increase, we can assert that $\#PD \geq past(\#PD, 2)$. Combining the two relations, we can prove the property above. □

Example 6. The technique presented in this paper can be applied to prove refinement rules for TPNs. A method based on temporal logic for proving that a TPN is a correct implementation of another TPN (specification) and a set of correct refinement rules are presented in [6]. Using counters the same proofs can be more easily conducted and understood.

For example, in Figure 6 we show a simple TPN and a refinement of its, that we want to prove to be correct, i.e. we want to prove that the refinement preserves the cause effect relationship between s and t, i.e. the following relation between firings of transitions t' and s' holds:

Theorem 4. $dist(\#t', d) \leq \#s' \leq dist(\#t', D)$

Proof. For the refined TPN, we can write (see Section 5.1):

$$dist(\#t', d_2) \leq \#r_1 + \#r_2 \leq dist(\#t', D_2) \quad (1)$$

For r_1 and r_2 , exploiting theorem 3, we can write:
$futr(\#r_1, D_1) + futr(\#r_2, D_1) \geq \#s'$ and $futr(\#r_1, d_1) + futr(\#r_2, d_1) \leq \#s'$

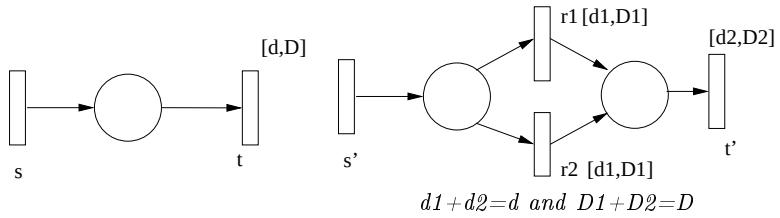


FIGURE 6. A TPN and its refinement

This can be rewritten:

$$dist(\#r_1 + \#r_2, d_1) \leq \#s' \leq dist(\#r_1 + \#r_2, D_1) \tag{2}$$

Translating (1) by D_1 and d_1, we obtain

(3)
$$dist(\#r_1+\#r_2, D_1) \leq dist(\#t', D_1+D_2) \text{ , and } dist(\#t', d_2+d_1) \leq dist(\#r_1+\#r_2, d_2+d_1)$$

Combining (2) and (3), we have proved Theorem 4 that *t'* is unique effect of *s'* in $[d_1 + d_2, D_1 + D_2]$ □

Furthermore, specifications based on counters are readily implementable, since counters are trivially computable by means of increments of integer-valued program variables.

The translation of temporal relationships in terms of state variables as counters can be of great advantage during the implementation phase. Indeed, events states, counters, and temporal translation can be immediately implemented in terms of simple hardware devices or software fragments so that the detailed design, dimensioning, and implementation of monitoring and controlling systems can be made quite systematic and intuitive using predefined, parametric components. For a really toy example, consider the following, where a program written in the syntax of LEGO® Mindstorms® Quite C [1] like checks whether A is unique cause of B in [5,7] time units.

```
#define N 100
clock clockAB[N];
int countA, countB;
while (1) {    // forever
  if A
       // if A, reset the countA-th clock and count A
       clockAB[countA++ % N].reset;
  if B {
       // if B, take the countB-th clock and count B
       check = clockAB[countB++ % N]
       // check the delay is bound
       if !( 5 <= check && check <=7) then ERROR
  }
}
```

7. Conclusions

We have presented our position that states and events should not be considered as exclusive alternatives, but are best exploited in an integrated usage to facilitate modeling and analysis of reactive, embedded time critical systems. As an illustration of this point we have discussed temporal logic axiomatizations for events, state-like counter variables, and relations among them. By means of examples derived from case studies, we have shown that events and counters can be the basis for effective reasoning and for implementing the modeled systems in real programming languages.

References

[1] mindstorms.lego.com.

[2] M. Archer and C. Heitmeyer. TAME: A specialized specification and verification system for timed automata. In Azer Bestavros, editor, *Work In Progress (WIP) Proceedings of the 17th IEEE Real-Time Systems Symposium (RTSS'96)*, pages 3–6, Washington, DC, December 1996. The WIP Proceedings is available at urlhttp://www.cs.bu.edu/pub/ieee-rts/rtss96/wip/proceedings.

[3] T. Bolognesi and E. Börger. Abstract state processes. In E. Börger, A. Gargantini, and E. Riccobene, editors, *Abstract State machines - Advances in Theory and Applications (ASM2003)*, number 2589 in LNCS, 2003.

[4] P. Caspi and N. Halbwachs. A functional model for describing and reasoning about time behaviour of computing systems. *Acta Informatica*, 22:595–627, 1985.

[5] J. Derrick and E. Boiten. *Refinement in Z and Object-Z*. Springer, 2001.

[6] Miguel Felder, Angelo Gargantini, and Angelo Morzenti. A theory of implementation and refinement in timed Petri nets. *Theoretical Computer Science*, 202(1–2):127–161, 28 July 1998.

[7] 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.

[8] Angelo Gargantini and Angelo Morzenti. Automated deductive analysis of time critical systems based on methodical formal specification. Technical Report 50, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 1999.

[9] Angelo Gargantini and Angelo Morzenti. Automated deductive requirements analysis of critical systems. *ACM Transactions on Software Engineering and Methodology*, 10(3):255–307, July 2001.

[10] Carlo Ghezzi, Dino Mandrioli, and Angelo Morzenti. TRIO: A logic language for executable specifications of real-time systems. *The Journal of Systems and Software*, 12(2):107–123, May 1990.

[11] Ron Koymans. *Specifying Message Passing and Time-Critical Systems with Temporal Logic*. PhD thesis, Eindhoven University of Technology, Netherland, 1989.

[12] D. Mandrioli, A. Morzenti, M. Pezze', P. San Pietro, and S. Silva. A petri net and logic approach to the specification and verification of real-time systems. In C. Heitmeyer and D. Mandrioli, editors, *Formal Methods for Real-Time Computing*, volume 5 of *Trends in Software*, chapter 5. Wiley, 1996.

[13] P. M. Merlin and D. J. Farber. Recoverability of communication protocols: Implications of a theoretical study. *IEEE Transactions on Communication*, COM-24:1036–1043, September 1976.

[14] Michael Merritt, Francesmary Modugno, and Mark R. Tuttle. Time-constrained automata (extended abstract). In J. C. M. Baeten and J. F. Groote, editors, *CONCUR '91: 2nd International Conference on Concurrency Theory*, volume 527 of *Lecture Notes in Computer Science*, pages 408–423, Amsterdam, The Netherlands, 26–29 August 1991. Springer-Verlag.

[15] A. Morzenti, D. Mandrioli, and C. Ghezzi. A Model Parametric Real-Time Logic. *ACM Transactions on Programming Languages and Systems*, 14(4):521–573, 1992.

[16] Jin Yang, Aloysius K. Mok, and Farn Wang. Symbolic model checking for event-driven real-time systems. *ACM Transactions on Programming Languages and Systems*, 19(2):386–412, March 1997.