# Expressing and Organizing Real Time Specification Patterns via Temporal Logics

P. BELLINI, P. NESI, D. ROGAI

DISIT-DSI, Distributed Systems and Internet Technology Laboratory

Department of Systems and Informatics, University of Florence

Via S. Marta, 3 - 50139 Florence, Italy

Tel: +39-055-4796567, Fax: +39-055-4796363

nesi@dsi.unifi.it, http://www.disit.dsi.unifi.it

Date: 14/05/2008

**Abstract**

Formal specification models provide support for the formal verification and validation of the system behaviour. This advantage is typically paid in terms of effort and time spent in learning and using formal methods and tools. The introduction and usage of patterns have a double impact. They stand for examples on how to cover classical problems with formal methods in many different notations, so that the user can shorten the time to understand if a formal method can be used to meet his purpose and how it can be used. Furthermore, they are used for shortening the specification time, by reusing and composing different patterns to cover the specification, thus producing more understandable specifications which refer to commonly known patterns. For these reasons, both interests in and usage of patterns are growing and a higher number of proposals for patterns and pattern classification/organisation has appeared in literature. This paper reports a review of the state of the art for real-time specification patterns, so as to organise them in a unified way, while providing some new patterns which complete the unified model. The proposed organization is based on some relationships among patterns as demonstrated in the paper. During the presentation the patterns have been formalised in TILCO-X, whereas in appendix a list of patterns with formalizations in several different logics such as TILCO, LTL, CTL, GIL, QRE, MTL, TCTL, RTGIL, is provided disguised as links to the locations where such formalizations can be recovered and/or are directly reported, if found not accessible in literature; this allows the reader to have a detailed view of all the classified patterns, including the ones already added. Furthermore, an example has been proposed to highlight the usefulness of the new identified patterns completing the unified model.

**Keywords:** patterns, real time specification pattern, formal methods, temporal logic, TILCO.

## 1 Introduction

Many applications must meet some temporal constraints in order to avoid critical and/or degenerative conditions; examples are in the area of avionics, robotics, process control, patient monitoring, etc. These applications are frequently considered real-time systems and in many cases they are modelled by using suitable specification techniques, which allow the verification and validation of the specified behaviour. For their specification, a set of formalisms is used to define relationships expressing temporal constraints among events – for example: properties of invariance, ordering among events, periodicity, liveness and safety conditions, etc.

In many cases, in order to cope with the above problems, formal models have been used as requirements analysis and/or specification techniques. The selected methods/formalisms are

often formal enough to verify and validate the specification with respect to system requirements by using theorem provers and/or model-checking techniques.
Among the scenarios of formal methods for requirement analysis and specification of real time systems, the ones based on temporal logics play a relevant role [Bellini, Mattolini and Nesi, 2000], [Alur and Henzinger, 1992]. The real-time system specification by using temporal logics is a time consuming work which can be performed with a sensible efficiency only by accurately trained people.
The presentation of specification patterns for temporal logics has recently improved its usability. Such patterns can be used for training and guiding analysts and developers to express requirements and specifications straight in a formal language [Dwyer et al, 1999], thus shortening the specification time needed to produce specifications which are more understandable, since they refer to common well-known design patterns.
In [Dwyer et al, 1999], a set of specification patterns has been proposed by using LTL (Linear Time Temporal Logic) [Manna and Pnueli, 1992], and CTL (Computational Tree Logic) [Clarke, Emerson and Sistla, 1986] temporal logics. Such specification patterns were mainly focussed on formalising specification properties such as occurrence and event ordering. The identification of those patterns has been produced by analysing a large set of typical specifications to extrapolate the typical recurrent formal structures in the requirements and in the specifications, i.e., the patterns. Therefore, similarly organised requirements may be formalised by using the same specification pattern.

In literature, a wide work has been performed to identify many kinds of pattern to formalise better and to reduce the time for analysis and designed processes. See for instance: analysis patterns [Fowler, 1997], architectural patterns [Shaw, 1996], [Douglass, 2003], design patterns [Gamma et al., 1994]. A different classification has been discussed in [Konrad and Cheng, 2004].

In the area of real time systems, more recently, Konrad and Cheng [Konrad and Cheng, 2005], [Konrad and Cheng, 2006], have proposed some real time specification patterns with the aim of extending the early defined patterns in [Dwyer et al, 1999], with some more specific patterns including quantitative temporal constraints. In [Konrad and Cheng, 2005], a set of real-time specification patterns expressing concepts of duration, periodic and real-time ordering has been presented, by using the temporal logics formalizations in MTL (Metric Temporal Logic) [Koymans, 1990], [Koymans, 1992], TCTL (Timed CTL) [Alur, 1991] and RTGIL (Real-Time Graphical Interval Logic) [Moser et al., 1997]. Konrad and Cheng, in their works, have also presented a classification of patterns based on the structured English formalization of patterns, which helps the user to understand the patterns. Patterns with real time properties have been also discussed in [Gruhn and Laue, 2005], [Gruhn and Laue, 2005b], while considering the pattern structure (e.g., scope, behaviour and events or occurrences) and presenting pattern models by means of timed automata.

In [Konrad and Cheng, 2005], in order to allow the specification of real time patterns, the selected temporal logics were in most cases quantitative extensions of the logics used in [Dwyer et al, 1999]. On the other hand, while observing the state of the art of temporal logics, only few of them have a metric of time; that is the possibility of expressing temporal constraints in a quantitative manner as described in the general survey about the temporal logics reported in [Bellini, Mattolini and Nesi, 2000]. Therefore, the patterns presented in [Dwyer et al, 1999] can be regarded as qualitative patterns with respect to the quantitative patterns proposed in [Konrad and Cheng, 2005]. Qualitative patterns are those in which the distance among events is not measured in terms of time units. Thus they can be specified also by means of temporal logics that do not present a metric for time such as LTL, CTL, GIL, etc. On the other hand, quantitative patterns are those that contain specific time bounds (temporal constraints expressed in quantitative manner, for example 5 time unitis) that are typically of real time systems. These patterns can be only expressed by using temporal logics that provide a metric of time such as TCTL, RTGIL, MTL, TILCO, etc. In many cases the latter type of logics are an evolution of the former type for logics: e.g., CTL and TCTL, GIL and RTGIL, etc.

Among the temporal logics endowed of a metric of time, we can classify the above mentioned MTL, TCTL, and RTGIL temporal logics, but also TILCO (Temporal Internal Logic with Compositional Operator) [Mattolini and Nesi, 2001], and TRIO [Felder and Morzenti, 1994]. They have been discussed and partially compared with the formers in [Bellini et al, 2000], [Mattolini and Nesi, 2001] and [Bellini, Nesi and Rogai, 2006]. All of them have a metric of time and therefore they can be used for specifying both qualitative and quantitative temporal constraints. In this category, MTL, TILCO and TRIO are first order temporal logics. Please note that, other temporal logics produce specifications structurally similar to the above logics or have similar operators [Bellini, Nesi, and Rogai, 2006], [Mattolini and Nesi, 2001]. Among the above mentioned temporal logics for quantitative reasoning, TILCO and RTGIL are the only interval logics, and TILCO presents operators making the specification quite compact. TILCO has been designed for the specification of real-time systems and it extends the FOL (First Order Logic) with a set of temporal operators. TILCO can be regarded as a generalization of the classical temporal logics operators possibly and henceforth to time intervals [Mattolini and Nesi, 2001]. TILCO allows the definition of expressions of ordering relationships among events, delays, time-outs, periodicity, liveness and safety conditions, etc. TILCO-X extended TILCO theory and logic by introducing operators for Dynamic Intervals and Bounded Happen [Bellini and Nesi, 2001]. They allowed to generalize *since* and *until* operators and to write simpler predicates including counting of events that may occur in intervals.

## 1.1 Paper organisation, contributions and related works

The usage of patterns has a double impact. They are an occasion to provide examples of the usage of formal methods in many different notations with respect to the same cases, on such grounds the user can shorten the time to understand if a formal model can be used for modelling the cases under specification. Besides, they can be used for shortening the specification time, reusing and composing different patterns for the specification of more complex problems and thus for producing more understandable specifications referring to other users at the commonly known patterns. For these reasons, the practice of specification pattern exploitation is mainly conceived as collecting and providing organised properties ready to be used.

This paper reports a review of the state of the art for real-time specification patterns, so as to organise the latter in a systematic and unified way, while providing some new patterns which complete the unified model. The analysis of the state of the art patterns has been performed with the aim of deriving a unified view of the results proposed in the above mentioned literature mainly considering [Dwyer et al, 1999] for qualitative patterns and [Konrad and Cheng, 2006] for quantitative patterns. In Section 3, an overview of qualitative and quantitative specification patterns is presented.

The analysing of the state of the art and the relationships among patterns proposed in it have persuaded us to produce a unified organisation in which a number of new patters have been placed as reported in this paper. The organisation proposed in [Dwyer et al, 1999] presented only qualitative patterns, while that proposed in [Konrad and Cheng, 2006], arranged some qualitative and quantitative patterns according to their description as “structure English”. On the other hand, the organization proposed in this paper is based on the effective behavioural relationships and properties of patterns. The identified relationships are discussed and formally proved along the paper. The study and the formalization of the proposed behavioural organisation allowed us to identify new and renovated patterns such as: Time-Constrained Precedence and Time Constrained Response that complete the unified model.
In Section 4, the proposed patterns organisation is reported together with relationships among patterns and the arrangement of the new proposed patterns. Section 5 presents both qualitative and quantitative patterns expressed in TILCO-X together with their relationships and related demonstrations. The presentation of patterns also includes a comparison among the different specifications in different logics, that allow to compare the different specification models.
Furthermore, during the pattern formalisation and formal proofs, a particular attention has been given in separating the specification of the pattern scope from the description of the pattern behaviour in order to make them re-usable, and the pattern structure more modular. A discussion about patterns’ scope is offered in Section 6. In [Konrad and Cheng, 2006], other scopes have been presented with respect to those proposed in [Dwyer et al, 1999]. In this paper, a generalization of the concept of scope is introduced.
Thus, the proposed pattern organisation and method for their formalisation can be used to reduce the time to understand if a formal model can be used for modelling the cases under specification.

During the article, the patterns have been analysed and formalised in TILCO-X [Bellini and Nesi, 2001] (an extended version of TILCO temporal interval logic [Mattolini and Nesi, 2001]). For this reason a short overview of TILCO-X is also reported in Section 2, together with its major operators and formalisms, which are used for reasoning on patterns and their scopes. On the other hand, in Appendix 1 the list of patterns with their corresponding formalizations in several logics TILCO, LTL, CTL, GIL, QRE, MTL, TCTL, RTGIL are offered. When their formalisations have been recovered from the literature the link has been reported, while in the other cases, the patterns have been directly specified by the authors in the several logics. This has been performed to offer at the reader a detailed view of the whole classified patterns, including those that have been added. Please note that not all patterns can be formalised in all logics. With TILCO-X, TCTL, RTGIL and MTL it is possible to formalise all the patterns proposed in [Dwyer et al, 1999] and in [Konrad and Cheng, 2006] and those proposed in this paper. Furthermore, in some cases, the TILCO-X specifications resulted quite simple thanks to the presence of specific operators.

In addition, a case study is proposed in Section 7 to give evidence about the usefulness of the new identified patterns, and of the unified model and organisation.

## 2 TILCO-X overview

TILCO-X is a logic language which can be used to specify temporal constraints in either a qualitative or a quantitative way; the meaning of a TILCO-X formula is given with respect to the current time. Time is discrete and linear and the temporal domain is the set of integers. The minimum time interval corresponds to one instant, the current time instant is represented by 0 and positive (negative) numbers represent future (past) time instants. The basic entity in TILCO-X is a temporal interval, the boundaries of which can be either included or excluded by using the usual notation with squared, ("[", "]") or round ("(", ")") brackets, respectively. TILCO-X has to be considered for the specification of synchronous systems, meaning that each system state update increments the system's clock by 1 time unit. The basic TILCO-X temporal operators are:

- "$@$", universal quantification over a temporal interval: $A@[2,44]$ means that A will be true from 2 and 44 time units, with respect to the evaluation time instant;
- "$?$", existential quantification over a temporal interval; $A?[2,44]$ means that A will be true at least for one time unit from 2 and 44 time units, with respect to the evaluation time instant;

Interval can be also defined as a single time instant. In this case, a compressed notation can be used, e.g., $A@[-3,-3] \equiv A@-3$.

Many temporal logics adopt *since* and *until* operators to specify dependencies among events. These operators make a strong distinction between past and future and, subsequently, their adoption often makes the specification more complex to be read. The adoption of a unique operator, as in TILCO-X, for the definition of ordering relationships among events, reduces in many cases the need for the adoption of nested since and until operators. Specifying the occurrence of one event with respect to a number of occurrences of another event is a situation arising quite often (operators for event counting are needed). For instance, *A* has to start after the arrival of *5* messages on channel *B* within interval *I*. To this end, TILCO-X includes operators called Dynamic Intervals and Bounded Happen.
Please note that the semantics and the deductive system of those operators and therefore of TILCO-X logic are reported in [Bellini and Nesi, 2001].

### 2.1 TILCO-X Dynamic Intervals

Dynamic Intervals allow to reduce the need of distinguishing between past and future for ordering relationships and to avoid the nesting of *since* and *until* operators in many cases. They reduce the number of quantifications and allow the combination of ordering and quantitative relationships. Thus in TILCO-X the temporal intervals are not only constant integer sets, but also dynamic interval bound defined as predicates. For example, TILCO-X formula $A@[10,+B)$ states that $A$ is true from 10 time units in the future until $B$ is true for the first time, where $+B$ identifies the first future instant in which $B$ is true (from the evaluation time instant), if such instant does not exist, $A$ is forever true in interval $[10,+\infty)$. These two conditions are represented in Figure 1, where blue bars depict the defined interval where predicate $A$ has to be true.

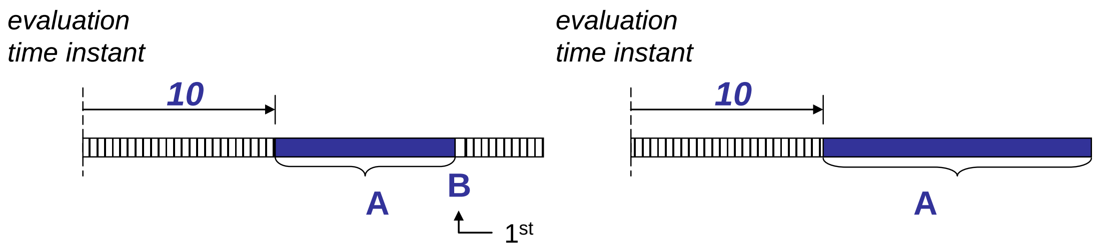


**Figure 1: Example of Dynamic Interval:** $A@[10,+B)$

In a similar way, an interval bound can be located in the past; for example, formula *A@(−B,0]* states that A has been true since the last time instant in which $B$ was true until the current instant. Where $-B$ identifies the last instant where $B$ was true. So that, we have an implicit operator + and - for referring to events in the future and past, respectively, with respect to the evaluation time instant.
With TILCO-X, to write intervals which start in the past and end in the future becomes possible; therefore, the above TILCO specification is greatly simplified:

$$(A \rightarrow B)@(-C,+D)$$

This TILCO-X formula can be read as: $A \Rightarrow B$ is true from the last occurrence of $C$ in the past and the first occurrence of $D$ in the future, with respect to the evaluation time instant.
In many cases, the definition of intervals with dynamic bounds (identified by the validity of a generic formula) is of great help in avoiding the adoption of nesting temporal quantifiers. Another example can be $A?[+B,+C]$ for which A happen at least one in interval [+B,+C].

Note that, the classical weak until and since operators can be defined with the following formulas:

$$\text{until}\, A\, B \equiv B@(0,+A)$$

$$\text{since}\, A\, B \equiv B@(-A,0)$$

The adoption of the Dynamic Interval operator allows writing expressions when events have to refer to time intervals defined in terms of other events.

### 2.2 TILCO-X Bounded Happen

Bounded Happen operator has been defined to increase constraint readability which includes the dependency on the counting of occurrences. Sometimes a constraint implies counting the number of event occurrences or in general how many times a formula is true in a given time interval. Bounded Happen operator is used to state that a formula is true in an interval from a minimum to a maximum number of times. For example, TILCO-X formula $A?_2[1,15)$ states that $A$ is true twice or more times in interval $[1,15)$. While $A?^3[1,15)$ states that $A$ is true up to three times in interval $[1, 15)$. By combining such operators, it can be stated that a formula $A$ has to be true in the interval from a minimum to a maximum number of times; with the following example: $A?_2^3[1, 15)$.

Bounded Happen operator can be used with the Dynamic Interval operator. The formula $A?_2^3[0,+B)$ states that $A$ happens two or three times from now until $B$ happens (see Figure 2):

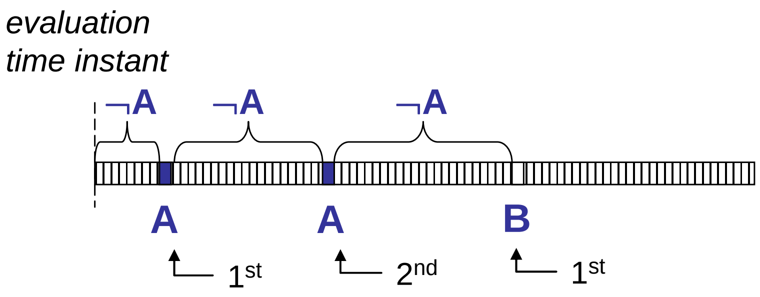


**Figure 2 – Example of Bounded Happen:** $A?_2^3[0,+B)$

## 3 Overview of Specification Patterns

In [Dwyer et al, 1999], a classification of specification pattern has been proposed. In [Alavi et al]. details about qualitative specification patterns are published and cover all the typical situations which a developer could have to deal with, when trying to define reactive systems.

Patterns are typically formalised considering the:

- **Pattern**: the pattern itself which is the property, the behaviour that has to be specified with the chosen formalism (formal model in this case). In this paper, a specification of a pattern with a given logic formalism is also called *mapping* as in [Dwyer et al, 1999]. So that, in Appendix I, the mappings for the considered patterns are reported for a certain number of different temporal logics;
- **Scope**: the extent of the program execution over which the pattern behaviour must hold. The scope is determined by specifying a starting and an ending state/event for the pattern: the scope consists of all states/events beginning with the starting state/event and up to and not including the ending state/event. Also the Scope has to be formalised with the chosen formalism. To this end, the person who is going to formalize the pattern can be more or less interested in making evident the distinction among the scope and the pattern itself, in order to have the so called "separation of concern".

Both pattern and scope refer to the occurrence of events/states that could be substituted with more complex predicates in the aim of creating more complex specifications/models, let's say for "composition of patterns".

### *3.1 Patterns scope*

In [Dwyer et al, 1999], five basic kinds of scopes have been proposed, as shown in Figure 3:

- **global** – the property has to hold for the entire program execution;
- **before *R*** – the property has to hold up to the occurrence of state/event *R*;

- **after *Q*** – the property has to hold after the occurrence of state/event *Q*;
- **between *Q* and *R*** – the property has to hold in every interval having state/event *Q* on left and state/event *R* on right; please note that multiple overlapped intervals having the same end point are included in the scope, see Figure 3, for the this scope and interval covering *Q*-*Q*-R sequence;
- **after *Q* until *R*** – the property has to hold in every interval having state/event *Q* on left and state/event *R* on right or no ending event; this means that this property holds even when the interval is not closed by *R*.

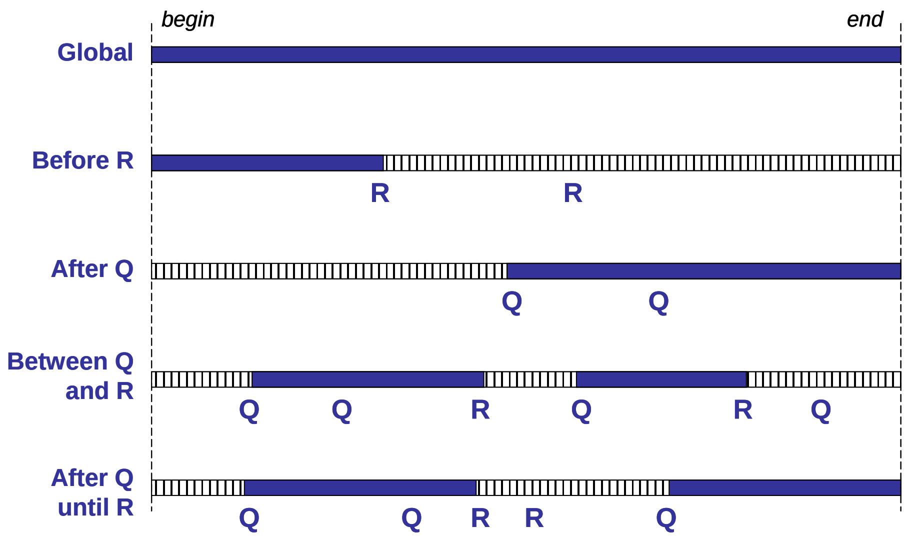


**Figure 3 – Pattern Scopes**

Experience points out that in most cases requirements are specified as properties of: (i) the whole program execution, or of (ii) the specific segments of the program execution. Therefore, a pattern system for properties allows someone to specify the system behaviour as to the specific status fragment/condition of the program execution [Dwyer et al, 1999].

In [Konrad and Cheng, 2006], other scopes have been presented:

- **in the presence of *F*** – a property has to hold only in an interval where *F* occurs at least once;
- **in the absence of *F*** – a property has to hold only in an interval where *F* never occurs;
- **from when *F* never holds** – a property has to hold only from the state/event where *F* is going to stay false forever.

Explicit scope operators do not appear in most specification formalisms; interval logics can be considered an exception, if the operators to define the interval are used for modelling the scope. Generally, it could be also possible to use scopes which are open on left and/or right. A discussion about these additional scopes and other aspects related to the scopes are reported in Section 7.

### *3.2 Patterns models*

In [Dwyer et al, 1999], patterns are classified as:

- **Occurrence Patterns** are used to express properties related to the existence or to the lack of existence of certain states/events in the pattern scope. They have been classified in four subtypes:
    - **Absence**, also known as *never happen.* The event will never occur within the scope;
    - **Universality***,* also known as *henceforth*. The event will always occur within the scope;
    - **Existence**, also known as *eventually.* The event may occur at least one time within the scope;
    - **Bounded Existence**. The event has to occur a fixed number of times within the scope. Variations of this pattern may be defined replacing the fixed counting of events with "*at least*" or an "*at most*" construct.
- **Order Patterns** are used to express requirements related to pairs of states/events during - defined scopes. There are two order-related patterns:
    - **Precedence**. *P* event has always to precede *Q* event within the scope.
    - **Response**, also known as Follows, Leads-To. *P* event has always to be followed by *Q* event within the scope.
    - **Chain Precedence**. A sequence of *Pi* events has always to precede a sequence of *Qi* events within the scope. It can be regarded as a generalisation of the Precedence pattern.
    - **Chain Response**. A sequence of *Pi* events has always to be followed by a sequence of *Qi* events within the scope. It can be regarded as a generalisation of the Response pattern.

In the above classification, the Chain Precedence and Chain Response patterns can be considered as specific cases (a specialization) of Precedence and Response patterns, respectively; since the occurrence of a sequence or of a chain of events can be regarded as the occurrence of the single event (chain or sequence) and in the patterns, the event *P* may be interpreted in that manner. For this reason, in the following they have not been reproduced in TILCO-X.

In [Dwyer et al, 1999], the proposed patterns are defined in terms of what happens in the future and never in terms of what has occurred in the past. The decision of Dwyer et al. about presenting only patterns referring to the future may be due to the used formalisms, and related limitations. In the following, for the presentation of TILCO-X–based pattern, both approaches have been offered in some cases, see Appendix 1, since TILCO-X allows reasoning in a uniform manner in both past and future.

In [Konrad and Cheng, 2006], a set of real time patterns has been proposed considering MTL, RTGIL and TCTL temporal logics. They have been classified as:

- **Duration Patterns** are used to express requirements related to the duration of a condition with respect to quantitative value. There are two basic patterns:
    - **Minimum Duration.** When *P* becomes true, it remains in that condition at least for a minimum time duration *t*;

    - **Maximum Duration.** When *P* becomes true, it remains in that condition at most for the maximum time duration *t*;
- **Periodic Patterns** are used to express requirements related to definition of periodic events/states. There is one related pattern:
    - **Bounded Recurrence** (called Time-Constrained Recurrence in the classification proposed herein)**.** Limits the period which a given occurrence has to happen within. *P* occurs every *t* time instants;
- **Real Time Order Patterns** are used to express requirements related to formalising patterns where the time duration among event occurrences is limited. There are two basic patterns:
    - **Bounded Response** (a specific cases also included in the Time-Constrained Response in the classification proposed in this paper)**.** Limits the maximum time duration from the event/state where a formula is true until another formula becomes true;
    - **Bounded Invariance** (called Time-Constrained Invariance in the classification proposed in this paper)**.** Limits the minimum time duration from the event/state where a formula is true once another formula is true.

Please note that, in the patterns proposed in [Konrad and Cheng, 2005], the word bounded is used for describing a bound in time, while in [Dwyer et al, 1999] the same word is used to refer to a limit in the number of event occurrences. For this reasons, in order to avoid confusion and provide a unified model, some of the patterns presented in [Konrad and Cheng, 2005] have been renamed in this paper as reported above, mainly by substituting "*Bounded*" with "*Time-Constrained*".

By analysing the relationships among all the above mentioned patterns, several similarities have been identified, which persuaded us to produce a unified organisation as reported and discussed in the next section.

## 4 Specification Patterns Organization

In this section, the unified organization of patterns is proposed by considering both qualitative and quantitative patterns. This approach required a reorganization of the existing pattern catalogues and an extension of the concept of *scope*. Our choice was to put in strict relationship real-time patterns with those already existent which are not based on the availability of a metric of time. That is the possibility of expressing temporal constraints in a quantitative manner such as it is possible with MTL, TILCO, and other logics [Bellini, Mattolini and Nesi, 2000] , [Konrad and Cheng, 2005].

In the proposed unified organisation of patterns, unlike [Konrad and Cheng, 2006], no radical distinction has been performed from qualitative and quantitative (also called real time) patterns in our pattern organisation. The organisation proposed in [Konrad and Cheng, 2006] kept the [Dwyer et al, 1999] hierarchy and added an additional hierarchy for the real time patterns. In their turn, for their purpose, they have been organised grouping together patterns which share a common root in terms of pattern description as "structure English". Our organisation is based on a different purpose as described below.

Figure 4 shows the proposed unified organisation that put together qualitative and real time patters. At the first layer of the organisation, the pattern categories are a unified view with respect to those proposed in [Dwyer et al, 1999] and [Konrad and Cheng, 2006]. After the first layer, the patterns are grouped according to these categories. The categories distinguish only which kind of constraint the pattern is applying to the predicates:

- **Occurrence**: properties which express if a given predicate has to occur, always, never, periodically or for a given amount of times. It has been defined in [Dwyer et al, 1999].
- **Duration**: properties that, even though not imposing the occurrence, require a predicate to hold for a given duration. It has been defined as a real-time type category in [Konrad and Cheng, 2006].
- **Order**: properties that put in relationship more predicates, by ordering them. It has been defined in [Dwyer et al, 1999].

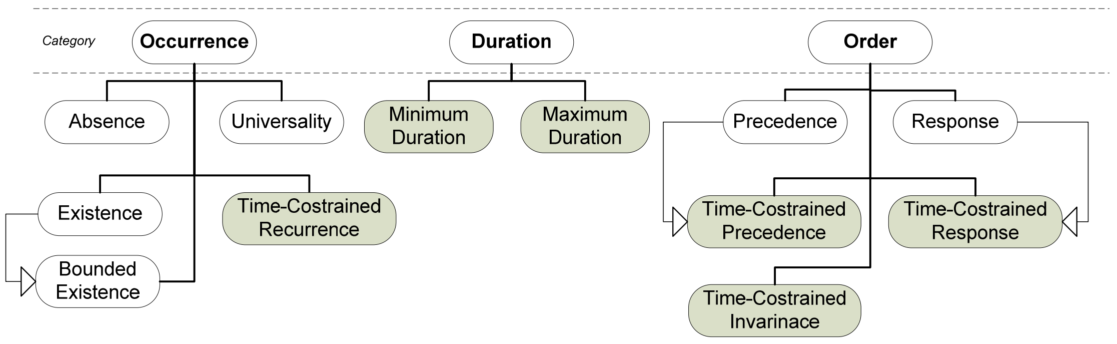


**Figure 4 – Proposed unified organisation of pattern and relationships, in grey the real time patterns, in white qualitative patterns and main categories.**

Both qualitative and quantitative patterns are organized in these categories. Qualitative patterns in the proposed unified organization correspond to those presented by [Dwyer et al, 1999]. Quantitative patterns (real time) have been marked in grey.

Therefore, the following remarks can be made:

- the Chain Precedence and Chain Response patterns of [Dwyer et al, 1999] have not been reported in the proposed organisation, since a chain of events can be considered as an event itself, which is also stated in [Gruhn and Laue, 2005]. Thus, they should be classified in the Order category respectively as a specific case of Precedence and Response patterns;
- the Duration Patterns corresponds to those presented in [Konrad and Cheng, 2006];
- the category of Periodic patterns proposed in [Konrad and Cheng, 2006] with only one pattern (Bounded Recurrence and called **Time-Constrained Recurrence** in this paper) has not been used, because the single pattern Time-Constrained Recurrence can be better classified as an occurrence pattern – i.e., periodic occurrences, which is demonstrated in the following section;
- the category of Real Time Order proposed in [Konrad and Cheng, 2006] with two patterns Bounded Response (a specific case of **Time-Constrained Response** in this paper) and Bounded Invariance (called **Time-Constrained Invariance** in this paper) has

been merged with the Order category. In fact, there are strong relevant relationships among them, as reported below.

- the Time-Constrained Precedence is a *new pattern* identified to complete the unified model. It consists in requiring a cause occurred in the past, in order to accept the present effect, similarly to Precedence pattern. Besides, in this pattern, time constraints can be specified as lower and upper bounds of the time window, located in the past, where the cause is expected. It is dual with respect to the Time-Constrained Response pattern. Please note that, the needed properties expressing time constraints in the past can be transformed in “pure future” form. On the other hand, they are more naturally expressed referring to the past, and for these reasons the authors deem of great value, in term of readability, to let a past property be formulated in its “past form”. With some temporal logics, which do not support past operators, a “pure future” transformation could be required when formulating a mapping of this pattern.
- the Time-Constrained Response pattern used in this paper is more general with respect to the Bounded Response pattern proposed in [Konrad and Cheng, 2006] since it is able to specify a lower and a upper bound for the expected effect. The one proposed in this paper presents a complete model for restricting the response to a given time window. This pattern can also produce simpler properties by setting bounds to extreme values as explained in the next paragraph. On the contrary, a single-sided constraint model would require to use two different properties in order to limit the occurrence of the response in a given time window (i.e. “*S responds to P after at most $k_{max}$ time units*”, and “*if P holds that not S holds for at least $k_{min}$ time units*”).

According to the performed analysis (reported in the following section), some relationships among the above mentioned patterns have been identified and depicted in Figure 4 using blank arrows. In particular, as better described in the following section, so co-called *“behavioural generalisation”* has been identified between some of them. In the diagram, the generalization is depicted by using a blank arrow. The general concept is that un-timed properties can be obtained by relaxing time constraints from timed properties.
For example,

- the Time-Constrained Response pattern, which models properties like “*S responds to P between $k_{\min}$ and $k_{\max}$*” can be used to obtain a Response Pattern by imposing simply qualitative time bounds as $k_{\min} = 0$ and $k_{\max} = \infty$. Therefore Time-Constrained Response with $k_{\min} = 0$ and $k_{\max} = \infty$ can replace Response wherever in a system without changing the system behaviour. Conceptually the Time-Constrained Response is more general since it includes the case of Response pattern, while it can be considered a subtype of the latter;
- along the same line of reasoning, the Precedence pattern can be regarded as a special case of the Time-Constrained Precedence. In fact, in this case, the equivalence can be obtained by imposing $k_{\min} = 0$ and $k_{\max} = -\infty$, because the time window is located in the past;
- Existence pattern can be regarded as a specific case of the Bounded Existence pattern, where the occurrence count is not limited, therefore one occurrence or more occurrences of P are accepted.

The above reported approach and comments have led to the proposed unified organization of patterns with respect to those reported by [Konrad and Cheng, 2006], [Dwyer et al, 1999]. According to the above description of the unified organisation some of the patterns have not been formalised in this section, while they are formalised in details in the next section together their specification.

## 5 Temporal Logic Specification of Patterns

In this section, the proposed unified organisation of patterns reported in the previous Section is discussed and supported while giving the evidence of the relationships among the patterns that confirm the validity of the unified organisation. The formalism used in the presentation of the patterns is TILCO-X, which resulted quite effective in the formalisation of many complex structures. TILCO-X can be used to formalize both qualitative and quantitative real-time patterns. On the other hand, the same demonstrations reported in terms of TILCO-X can be replicated in other logics mentioned before such as: RTGIL, MTL, TCTL, etc.

According to the previous discussion, in the literature a certain number of qualitative and quantitative patterns have been presented by using several different temporal logics such as: RTGIL, MTL, TCTL, LTL, GIL, etc. (see [Konrad and Cheng, 2005] and [Dwyer et al, 1999]). In all these cases, the patterns have been presented by:

- Referring to a point in which the process starts, nothing has happened before;
- Considering the pattern behaviour from the process start to the infinite;
- Describing the actions towards the future, fixing a point and stating what is going to happen in the next status or state evolution.

By using TILCO-X for the pattern specification, we noticed some differences that make some of the specification mappings more intuitive and somehow different with respect to the ones presented for other logics in [Konrad and Cheng, 2005], [Dwyer et al, 1999]. The main differences are based on the fact that in TILCO-X:

- it is possible to specify formulas in the past and in future in a uniform manner [Bellini et al., 2006];
- a specific process start is missing; while one can be defined by means of $(start : A) \equiv process\_start \rightarrow A$ thus *process_start* is the given time instant from which any property has to be satisfied;
- once the start has been defined, it is possible to define a rule imposing the validity of the formula from the process start to the time limit (e.g., infinite) $(rule : A) \equiv start\ \ A @ [0, +\infty)$.

In fact, *start* identifies an expression which has to be verified only on the initial time instant, while *rule* imposes the expression to be verified on the entire time domain. Therefore, patterns are typically presented in the form of *start* or of *rule* depending on the needs.

This formalisation for the specification of patterns can be used to formalize the pattern in all the considered temporal logics, and allows to obtain a more evident distinction from pattern and the scope. For this purpose, comparison has been offered among the specification methods used for producing the pattern specifications in different temporal logics.

Therefore, in Appendix 1, all the patterns discussed in the proposed unified organization are reported for all the scopes. The formalisation of the patterns has been offered in TILCO-X and in other formalisms (the above mentioned temporal logics, RTGIL, MTL, TCTL, LTL, GIL, etc.). When those specifications have been found accessible in other documents or web sites, citations and/or links have been provided; when the they have been missing, a specification has been provided for completeness.

In the next Section, a description of the patterns according to the proposed unified organization described in the previous Section is reported (please refer to Appendix 1 whenever an exhaustive view of all formulas for a pattern is needed). The description focuses on presenting, demonstrating and stressing the main relationships among patterns. In some cases, such relationships are of behavioural specialization as shown in the sequel.
Please note that the following section also includes the new and renovated patterns such as: Time-Constrained Precedence and Time Constrained Response Invariance that complete the unified model.

### *5.1 Occurrence specification patterns*

As stated in Section 4, the category of the Occurrence patterns includes: Absence, Universality, Existence, Bounded existence and Time-Constrained.

The Absence Specification Pattern aims at describing a portion of a system's execution that is free of certain events or states. As it can be noted by observing the Absence (Occurrence) pattern reported below, the scopes are modelled through dynamic intervals. Therefore, the TILCO-X mapping seems to be very concise for every occurrence pattern on each scope.

**Pattern Name and Classification**
Absence: Occurrence Specification Pattern

**Temporal Logic Mappings**
TILCO-X

| | |
|---|---|
| Globally: | $start : \neg P\ @[0,+\infty)$ |
| Before R: | $start : R\ ?(0,+\infty) \rightarrow \neg P\ @[0,+R)$ |
| After Q: | $start : \neg P\ @[+Q,+\infty)$ |
| Between Q and R: | $rule : R\ ?(0,+\infty) \rightarrow \neg P\ @[+Q,+R)$ |
| After Q until R: | $rule : \neg P\ @[+Q,+R)$ |

The definition of interval-based operators like “$@$”, “$?$”, and “$?_{\min}^{\max}$” allows to reuse the pattern mappings for all the other Occurrence specification mappings.
Please note that in TILCO, in order to specify all the other Occurrence patters on the five scopes presented in Section 3.1 it  is only needed to replace the operators on the left, while keeping the scopes independently modelled through intervals,

- Globally: $[0,+\infty)$
- Before R: $[0,+R)$
- After Q: $[Q,+\infty)$
- Between Q and R: $[+Q,+R)$
- After Q until R: $[+Q,+R)$

For example, all the occurrence patterns for the "*After Q until R*" scope are.

- Universality: $rule : P\,@[+Q,+R)$
- Absence: $rule : \neg P\,@[+Q,+R)$
- Existence $rule : true\,?[+Q,+R) \rightarrow P\,?[+Q,+R)$
- Bounded Existence: $rule : true\,?[+Q,+R) \rightarrow P\,?_{\min}^{\max}[+Q,+R)$
- Time-Constrained Recurrence $rule : (true\,?[k,+R) \rightarrow P\,?[0,k))@[+Q,+R)$

Please note that formulas share the same structure. The same structure applies for all the other scopes. TILCO-X operators model the occurrence patterns in a quite simple way, while leaving to the intervals the definition of the pattern scope and thus keeping separate the two concepts into the specification.

The only difference is in the semantic of "$@$" and "$?$" when the specified time interval is empty (i.e., $R$ happens before $Q$ with respect to the evaluation instant). In that case the "$@$" operator is *vacuously true*, while "$?$" operator on an empty interval is evaluated as false [Mattolini and Nesi, 2001]. For example, in the Existence Pattern, formula $true\,?[+Q,+R)$ states that, with respect to the evaluation time instant, a non-empty interval [+Q,+R] will occur in the future (Q and R are predicates).

According to the definition of the scope in [Konrad and Cheng, 2006], the model accepts the presence of multiple $Q$ instances in the interval and it is valid in all of them from $Q$ to $R$. If the scope needs to be restricted to start from the first $Q$, the $Q$ in the scope should be substituted by: $(Q \wedge \neg Q\,@(-R,0))$. This rewriting can be applied also to some other patterns, for the same purpose. Please note that the "past" semantic of dynamic interval allows the identification of the first $Q$ in a time interval which begins with $Q$ and ends with $R$ with a quite simple formula; this would be much more complex using only future operators.

Among the Occurrence patterns, a relationship of behavioural specialization has been identified. A model of the Bounded Existence pattern is indeed a model of the Existence pattern in the corresponding scopes: if $P$ exists in a limited number of times from a minimum to a maximum, it surely occurs at least once. TILCO-X semantics maps this concept with this substitution:

$$P?_1^{\infty}\, i \equiv P?i$$

Where: *i* is any time interval (dynamic or not) [Mattolini and Nesi, 2001]. This relationship is also confirmed among the pattern mappings in LTL or CTL proposed by [Dwyer et al, 1999]. Please note that, in TILCO-X, the specification of the Bounded Existence pattern turns out to be quite simple with respect to the specifications performed in formalisms which do not have operators for modelling/counting the occurrences (e.g., LTL).

When examining other temporal logic mappings, the following remarks can be summed up:

- The LTL mapping for "*Universality of P*" in "*Between Q and R*" scope is expressed as `[](Q & !R -> (!R W (P & !R)))` while the "*Occurrence of P*" over the same scope is expressed as `[]((Q & !R & <>R) -> (P U R))`. Please note that, on the right side of both formulae expressions appear different: since no interval operators are present, the right bound event *R* of the scope is used to impose the occurrence of *P* ("*not R holds until (P and not R) holds*"). In TILCO-X expressions, the use of "?" operator allows to distinguish parts of the formula which depend on the scope or on the pattern itself.
- The GIL mapping for "*Occurrence of P*" in "*Between Q and R*" scope and for "*Universality of P*" in "*Between Q until R*" are shown in Figure 5 (respectively "a" and "b"). Please note that, an "or" condition is added in order to express the situation where *R* never holds in the future. In TILCO-X, this addition is not applied, since the interval semantics can model unbounded intervals. In fact with $P @[+Q,+R)$ the "weak until" semantics is applied and if *R* never holds after *Q*, the formula is equivalent to $P @[+Q,+\infty)$.

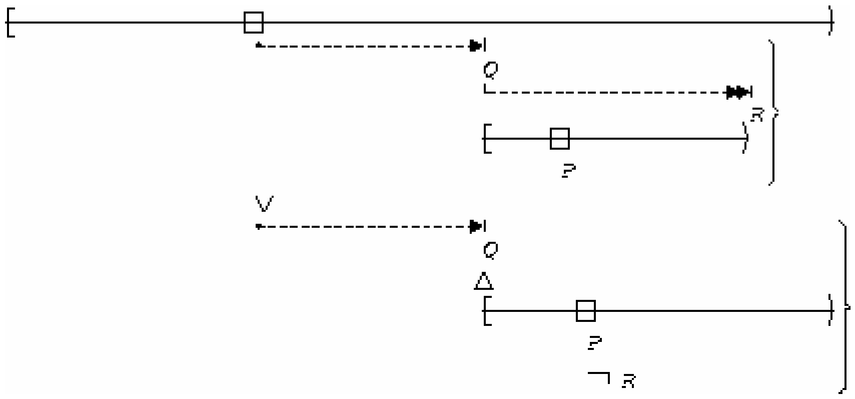


**Figure 5 – GIL mapping for Pattern: Universality, Scope: *Between Q until R***

### 5.2 *Duration Specification Patterns*

As stated in Section 4, the category of the Duration patterns includes: minimum duration and maximum duration of events. The minimum duration describes a condition in which "*once P becomes true, it holds for at least k time instants*", while the maximum duration states that "*once P becomes true, it holds for at most k time instants*".

When observing and comparing those patterns as reported in Appendix 1, it can be remarked that scopes are highlighted and therefore the same pattern specification and scope can be managed independently.

In the following patterns, the specification segment $(\neg P @ -1 \wedge P)$ identifies the occurrence of a false-true transition of *P*.

The duration constraints can be imposed with quantitative intervals. For example, considering both patterns in the same scope "*After Q*".

- Minimum duration $start : ((\neg P\ @{-1} \wedge P) \rightarrow P\ @(0,k))@[+Q,+\infty)$
- Maximum duration $start : ((\neg P\ @{-1} \wedge P) \rightarrow \neg P\ ?(0,k))@[+Q,+\infty)$

The first property is dual with respect to the second since

$$\neg(P\ @(0,k)) \Leftrightarrow \neg P\ ?(0,k).$$

### 5.3 Order Specification Patterns

Order specification patterns include: Precedence, Response, Time-Constrained Precedence, Timed-Constrained Response and Time-Constrained Invariance.

#### 5.3.1 Precedence pattern

The Precedence Specification Pattern is used to describe relationships between a pair of events/states where the occurrence of the first is a necessary pre-condition for the occurrence of the second. It can be expressed as follows: an occurrence of the second is enabled by an occurrence of the first. Precedence properties occur quite commonly in specifications of concurrent systems.
The precedence property is intuitively a "past-based" formula; the following example depicts two different mappings of "*S precedes P*" on "*Between Q and R*".

- with past mapping $rule : R\ ?(0,+\infty) \rightarrow (P \rightarrow S\ ?[-Q,0))@[+Q,+R)$
- pure future mapping $rule : Q \wedge \neg R \wedge R\ ?(0,+\infty) \rightarrow \neg P\ @[0,+(S \wedge R))$

Please note that the past formula allows keeping scope and pattern specifications independent each another; the future formula uses the dynamic interval with the conjunction of S and R (scope boundary). Furthermore the past formula is more readable, since it is still easy to recognize that "*if P occurs, then S has occurred before*".
The use of past in the intervals is a fundamental feature in order to obtain such an intuitive mapping of the Precedence concept. The use of past keeps intact the actual "aim" of the expressed property, which is to verify a condition regarding the past with respect to the occurrence of P.
The model for pattern "*S precedes P between* $k_{\min}$ *and* $k_{\max}$" is more general and includes the case of "*S precedes P*" when $k_{\min}$ is the evaluation time instant (i.e., 0, zero) and $k_{\max}$ is the left bound of the scope. This is presented in Section 5.3.5.

Temporal logic mappings were produced for Precedence pattern in LTL, GIL etc. For these temporal logics the past operators are not present in their basic form (PLTL has been defined including past operators for LTL). In fact, the expression for "*S precedes P*" are twisted in order to obtain a pure future formula: "*not P holds until S holds*". Therefore, LTL mapping for Precedence pattern on "*After Q*" scope is expressed as `[]!Q | <>(Q & (!P W S))`.
TILCO-X, by using time interval, that can be located in the past, is using $P \rightarrow S\ ?[-Q,0)$ to impose that *S* has to occur after the beginning of the scope, but before the current time instant.

### 5.3.2 Response pattern

The Response Specification Pattern is used to describe cause-effect relationships between a pair of events/states. An occurrence of the first, the cause, must be followed by an occurrence of the second, the effect.

In a similar way as it occurs with Precedence, Response pattern is quite commonly used in specifications of concurrent systems. Note that a Response property is like a converse of a Precedence property. Precedence says that some cause precedes each effect, and Response says that some effect follows each cause. They are not equivalent, because a Response allows effects to occur without causes (Precedence similarly allows causes to occur without subsequent effects).
The mappings with TILCO-X of Response pattern preserve the same structure of Precedence, while using dynamic interval with future bounds. In fact, “*S responds to P*” in “*Between Q and R*” can be expressed as:

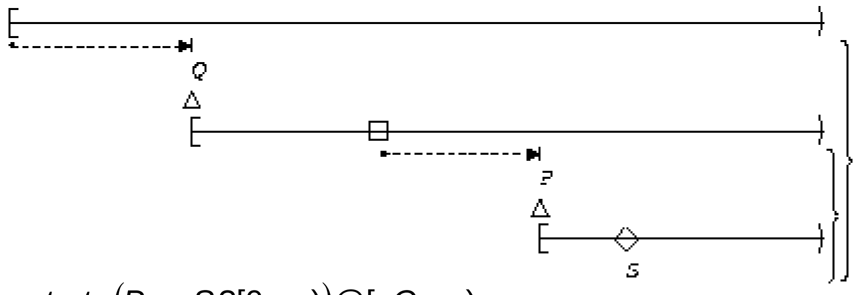
$$\mathit{rule}: R\,?(0,+\infty) \rightarrow \left(P \rightarrow S\,?[0,+R)\right)@[+Q,+R).$$

In this case, the interval in which *S* has to occur is between the occurrence of *P* and the end of the scope, while, in the corresponding Precedence mapping, the interval is between the start of the scope and the occurrence of *P*.
Alike to Precedence, the model for pattern “*S responds to P between* $k_{\min}$ *and* $k_{\max}$” is more general and includes the case of “*S responds to P*” when $k_{\min}$ is the evaluation time instant and $k_{\max}$ is the right bound of the scope. This is demonstrated in Section 5.3.6.

As to expressing this pattern in LTL, GIL and TILCO-X, similar expressions are obtained when the scope is not so restrictive. For example, in the scope “*After Q*”, the tree mappings are presented in Figure 6: a) LTL, b) GIL and c) TILCO-X.

a) `[](Q -> [](P -> <>S))`

b)



c) $\mathit{start}: \left(P \rightarrow S\,?[0,+\infty)\right)@[+Q,+\infty)$

**Figure 6 – Response Pattern on “*After Q*” scope**

Differences arise when the scope is an interval like “*Between Q and R*”, where LTL, without interval-based operators, has to formulate as follows: `[]((Q & !R & <>R) -> (P -> (!R U (S & !R))) U R)` where it is required that “*R does not hold before S*”. The latter formula requires more operators and operands than those created with intervals, because the property is based on a scope which is an interval. The LTL formula needs to apply the operator on *R* (“*not R until …*”), since the formula aims at verifying the behaviour of *S*.

### 5.3.3 Consideration on Order Specification Patterns

Please note that interval-based logic can model the scopes in a readable manner and it can keep scopes apart from the specification of pattern behaviour, since the former can be regarded as an interval. Generally speaking, the Precedence, Response Specification patterns can be expressed by defining "start of the scope" and "end of the scope"; these events/states need to be expressed as if considered at any time instant inside the scope. The definitions are reported in the Table below.

| **Scope** | **Beginning of scope** | | **End of scope** |
|---|---|---|---|
| | w.r.t. process_start | w.r.t time instants inside scope | |
| Parameter predicates | $scope\_beg$ | $scope\_beg\_in$ | $scope\_end$ |
| Globally | $0$ | $-process\_start$ | $+\infty$ |
| Before R | $0$ | $-process\_start$ | $+R$ |
| After Q | $+Q$ | $-Q$ | $+\infty$ |
| Between Q and R | $+Q$ | $-Q$ | $+R$ *(must exist)* |
| After Q until R | $+Q$ | $-Q$ | $+R$ |

In the second row of the table, parameters predicates have been reported in order to indicate general scope bounds in expressing TILCO-X mapping, regardless of the scope.

***Precedes:***
The general expression of "*S precedes P*" for the first three scopes can be written in terms of parameter predicates (defined as reported in the above table) as

$$start : (P \rightarrow S?[scope\_beg\_in, 0))@[scope\_beg, scope\_end).$$

The other two scopes define potentially an infinite set of intervals, therefore it is not possible to obtain expression which are evaluated only at start time instant, while the need to use a "rule" is self-evident, so as to detect any scope realization (i.e., whenever an interval between *Q* and *R* takes place).
As depicted in the Universality Pattern (see Appendix 1), to assert a property *P* at any time instant after Q until R means to impose $P@[+Q,+R)$ on a single time instant just before an interval that begins with *Q* and end with *R;* when using "rule", the desired expression is obtained and *P* is asserted in all the intervals which are delimited by *Q* and *R* along the time axis.
For such reasons, the general expression for "*S precedes P*" is still valid for "*After Q until R*", while being written with "rule" statement as

$$rule : (P \rightarrow S?[scope\_beg\_in, 0))@[scope\_beg, scope\_end)$$

and the expression of "*Between Q and R*" only adds the scope existence (i.e. R must happen):

$$rule : \exists scope\_end \rightarrow \\ (P \rightarrow S?[scope\_beg\_in, 0))@[scope\_beg, scope\_end)$$

***Responds:***
Similarly, the general expression of "*S responds to P*" for the first three scopes ("*Globally*", "*Before R*" and "*After Q*") can be written as

$$start : (P \rightarrow S\,?[0, scope\_end))@[scope\_beg, scope\_end).$$

The "*After Q until R*" and "*Between Q and R*" scopes can be respectively written as

$$rule : (P \rightarrow S\,?[0, scope\_end))@[scope\_beg, scope\_end)$$

and

$$rule : \exists scope\_end \rightarrow (P \rightarrow S\,?[0, scope\_end))@[scope\_beg, scope\_end)$$

Some of the pattern mappings could accept simpler expressions. Therefore the result of maintaining the same clear structure for all the mappings, while distinguishing among scopes and pattern intents, has been considered of great value. This could help in reusing/extending these mappings to adapt easily their formulae to specific behaviours. In Appendix 1, the alternative and simpler formalizations are also reported, even if only for some patterns.

In LTL and GIL temporal logic, the above described structure cannot include all the mappings related to Order patterns, due to the lack of past operators and of "*weak-until based*" interval bounds. Furthermore in LTL and GIL mappings on scope such as "*Between Q and R*" and "*After Q until R*" start with "*It is always the case that, if Q holds and R will hold in the future then…*". In TILCO-X, this form could be used, but to let in a clear form the interval "*Q-R*" has been considered of great value for the reader. Two versions of TILCO-X mapping for "*S responds to P*" in *After Q until R* scope are reported:

$rule : (P \rightarrow S\,?[0,+R))@[+Q,+R)$ or $rule : Q \rightarrow ((P \rightarrow S\,?[0,+R))@[0,+R))$

## 5.3.4 Time-Constrained Precedence Pattern

The above generalization suggested how to generalize Order pattern to add real-time quantification of the event relationships. Since TILCO-X enables specification of time intervals with both qualitative (i.e., events) and quantitative (i.e., time durations) manners, the use of dynamic interval allows to introduce metric of time (e.g., formalizing distances among events in terms of time units) for Order Patterns, while maintaining a comprehensible structure.

For example, for the Time-Constrained Precedence Pattern, few examples for some scopes are as follows:

- Globally $$start : \left( P \rightarrow \left( \begin{pmatrix} true\,?[-k_{\max}, -process\_start) \wedge \\ S\,?[-process\_start, -k_{\min}) \end{pmatrix} \vee \begin{pmatrix} true\,?[-process\_start, -k_{\max}] \wedge \\ S\,?[-k_{\max}, -k_{\min}) \end{pmatrix} \right) \right) @[0,+\infty)$$
- After Q until R $$rule : \left( P \rightarrow \begin{pmatrix} (true\,?[-k_{\max}, -Q) \wedge S\,?[-Q, -k_{\min})) \vee \\ (true\,?[-Q, -k_{\max}] \wedge S\,?[-k_{\max}, -k_{\min})) \end{pmatrix} \right) @[+Q,+R)$$

The augmented expression, introduced to impose a real-time property to the occurrence of S, is made complex to distinguish, when the left bound of the scope has occurred before the $k_{\max}$ time instants in the past. In Figure 7, two different conditions are presented, please note that *S* must precede *P* after the scope boundary, if it happens within the requested time duration.

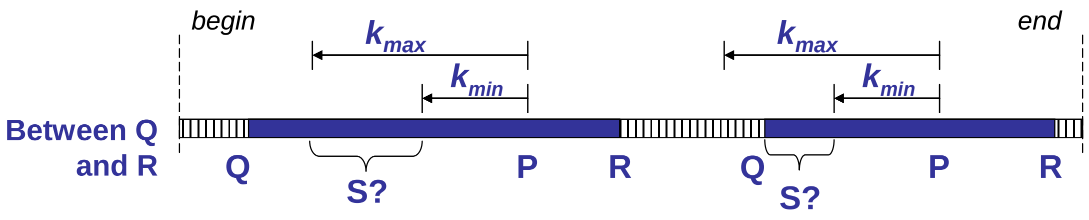


**Figure 7 – Scope boundaries and time durations**

It has to be highlighted that all the mappings of these patterns have been realized by reusing the same formula structure (see Appendix 1), which is created on the basis of the start and the end of each scope:

$$start:\left(P\rightarrow\left(\begin{pmatrix}true?[-k_{\max},scope\_beg\_in)\wedge\\ S?[scope\_beg\_in,-k_{\min})\end{pmatrix}\vee\begin{pmatrix}true?[scope\_beg\_in,-k_{\max})\wedge\\ S?[-k_{\max},-k_{\min})\end{pmatrix}\right)\right)@[scope\_beg,scope\_end)$$

If *process_start* definition also implies that all the examined predicates are false before such instant:

$$(\neg P\wedge\neg Q\wedge\neg R\wedge\neg S)@(-\infty,process\_start).$$

the above formalizations can be simplified. Some examples are given below:

- *"Globally, S precedes P between* $k_{\min}$ *and* $k_{\max}$*"*:
  $start:(P\rightarrow S?[-k_{\max},-k_{\min}))@[0,+\infty)$
- *"Before R, S precedes P between* $k_{\min}$ *and* $k_{\max}$*"*:
  $start:(P\rightarrow S?[-k_{\max},-k_{\min}))@[0,+R)$

Generally, replacing "quantitative" time constants with "qualitative" scope bounds, the Order Patterns as defined by [Dwyer et al, 1999] are obtained. Thus the latter are a special case of Time-Constrained version. What follows is a demonstration of the fact that Time-Constrained Precedence generalizes Precedence. It can be proved that to express "*After Q until R, S precedes P*" is equivalent to express "*After Q until R, S precedes P between* $k_{\min}$ *and* $k_{\max}$" where $k_{\min}=0$ and $-k_{\max}=-Q$ (the left side of the scope).

Therefore, the real-time TILCO-X mapping can be rewritten as:

$$rule:\left(P\rightarrow\begin{pmatrix}(true?[-Q,-Q)\wedge S?[-Q,0))\vee\\ (true?[-Q,-Q]\wedge S?[-Q,0))\end{pmatrix}\right)@[+Q,+R)$$

and according to the dynamic interval semantics of TILCO-X, it can be stated that

$$true\,?[-Q,-Q) = false \text{ (empty interval)}$$

$$true\,?[-Q,-Q] = true \text{ (non-empty interval)}$$

Therefore, the Time-Constrained Precedence mapping can be simplified to

$$rule : (P \rightarrow S\,?[-Q,0))@[0,+R)$$

Which is exactly the same expression of the Precedence Pattern mapping on scope "*After Q until R*".

The MTL and TCTL mappings for Time-Constrained Precedence pattern have been created by the authors of this paper and are reported in Appendix 1, for example in MTL as to "*Before R*" scope, the result is as follows:

$$\text{<>R->([]}_{<k_m-k_{min}}\text{!S \& []}_{<k_{max}}\text{!R->[]}_{=k_{max}}\text{ !P)UR.}$$

Please note that the subtraction of the quantitative bounds is present in order to impose the desired behaviour in the future: "*if not S holds for a period then not P holds at a given time instant in the future*".

### 5.3.5 Time-Constrained Response Pattern

Time-Constrained Response Pattern can be defined in a similar way to Time-Constrained Precedence Pattern. In this case, the right bound of the scope has to be evaluated with respect to $k_{\max}$. The Time-Constrained Response Pattern can be regarded as a more general version of the Bounded Response pattern proposed by [Konrad and Chen, 2006]. In fact, it present both temporal bounds

Some example of TILCO-X mappings are given below, while the complete set of pattern mappings is presented in Appendix 1:

- Before R

$$start : R\,?(0,+\infty) \rightarrow \left( P \rightarrow \begin{pmatrix} (true\,?[+R,k_{max}) \wedge S\,?[k_{min},+R)) \vee \\ (true\,?[k_{max},+R] \wedge S\,?[k_{min},k_{max})) \end{pmatrix} \right) @[0,+R)$$

- After Q: $start : (P \rightarrow S\,?[k_{min},k_{max}))@[+Q,+\infty)$

Please note for the "*After Q*" scope the formula seems simpler, since for this scope there is no need to have a right bound.

Even in this case the Time-Constrained Response Pattern is a generalization of the corresponding "un-constrained" Response Pattern .The demonstration is taken by proving that to express "*Before R, S responds to P*" is equivalent to express "*Before R, S responds to P between $k_{\min}$ and $k_{\max}$*" where $k_{min} = 0$ and $k_{max} = +R$ . Therefore, the TILCO-X mapping of this pattern can be rewritten as:

$$start : R?(0,+\infty) \rightarrow \left( P \rightarrow \begin{pmatrix} (true?[+R,+R) \wedge S?[0,+R)) \vee \\ (true?[+R,+R] \wedge S?[0,+R)) \end{pmatrix} \right) @[0,+R)$$

and according to the dynamic interval semantics of TILCO-X, it can be stated that:

$$true?[+R,+R) = false;$$

$$true?[+R,+R] = true.$$

Therefore, Time-Constrained Response maps exactly Response in the scope "*Before R*".

$$start : R?(0,+\infty) \rightarrow (P \rightarrow S?[0,+R))@[0,+R)$$

In the specification of real-time constrains, the use of "*between* $k_{\min}$ *and* $k_{\max}$" is a generalization with respect to the Bounded Response defined in [Konrad and Cheng, 2006], where one-bound constraint has been used. This Pattern can be obtained by replacing one of the quantitative boundaries ($k_{\min}$, $k_{\max}$) with a qualitative one, which can be "now", "start of the scope" or "end of the scope".

Observing MTL and TILCO-X about Time-Constrained Response pattern, by using only a single-sided constraint (i.e., the version expressed in [Konrad and Cheng, 2006]) the MTL formula for "*Before R, S responds to P after at most $k_{max}$ time instants*" is
`<>R->((P-> (!R U`$_{\leq k_{max}}$`(S & !R))) U R`
The "*bounded until*" operator "$U_{\leq k}$" can include the hybrid semantics of quantitative and qualitative constraint: the case in which end of the scope occurs before $k_{max}$ time instants is covered as well as the case in which end of the scope occurs after.
On the other hand, the equivalent TILCO-X formula is

$$start : R?(0,+\infty) \rightarrow \left( P \rightarrow \begin{pmatrix} (true?[+R,k_{\max}) \wedge S?[0,+R)) \vee \\ (true?[k_{\max},+R] \wedge S?[0,k_{\max})) \end{pmatrix} \right) @[0,+R).$$

In TILCO-X, a distinction needs to be taken $true?[+R,k_{\max})$, then two different sub-formulas, one for the scope end and one for the amount of time instants, have been used. In the double-sided, version of quantitative constraints, this semantic gap is recovered since the TILCO-X formula remains the same while the MTL version has to be doubled.

### 5.3.6 Time-Constrained Invariance Pattern

Also Invariance Pattern is related to the corresponding real-time version: Time-Constrained Invariance. For example:

- After Q: $start : (P \rightarrow S@[0,k))@[+Q,+\infty)$
- Between Q and R $rule : R?(0,+\infty) \rightarrow (P \rightarrow true[k,+R] \wedge S@[0,k))@[+Q,+R)$

Please note that, as in [Konrad and Cheng, 2006], the scope end cannot interrupt the time length where *S* holds (see Figure 8). The formula $true[k,+R]$ is placed to state that "*R occurs after at least k time instants*".

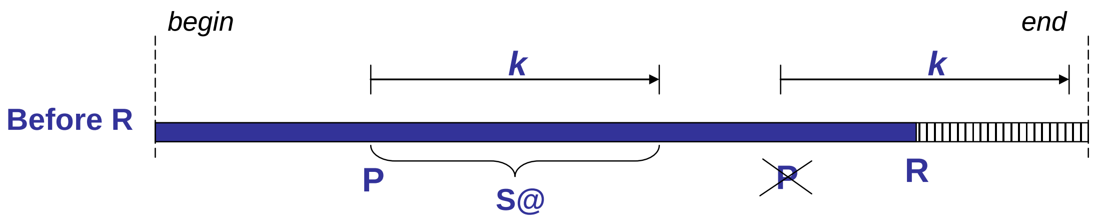


**Figure 8 – Examples of Time-Constrained Invariance property**

In MTL, the Time-Constrained Invariance pattern in "Before R" is represented as `<>R ->(P-> []`$_{<k}$`(S & !R)) U R)`. Please note that, similarly to the Response pattern, a property is imposed to *R* "*to not hold for at least k time instant*". In TILCO-X formula, which has been presented above, the case where scope end occurs within k time instants is considered by using $true[k,+R]$. This expression allows imposing that an interval of *k* time instants has to exist since now to the end of the scope and *S* holds in this interval.

## 6 Discussion on Pattern Scopes

As mentioned above, in [Konrad and Cheng, 2006] other scopes have been presented with respect to those proposed in [Dwyer et al, 1999]. These additional scopes may be specified according to the following constructs, where *P* is modelled as Universality Pattern, In order to obtain other patterns of the Occurrence category, it is needed to change the temporal operator on the time interval.

**Scope: *in the presence of F*** – a property has to hold only in an interval where *F* occurs at least once.

$$start : F?[0,+\infty) \rightarrow P@[0,+\infty)$$

**Scope: *in the absence of F*** – a property has to hold only in an interval where *F* never occurs:

$$start : \neg F?[0,+\infty) \rightarrow P@[0,+\infty)$$

**Scope: *from when F never holds*** – a property has to hold only from the state/event where *F* is going to stay false for ever:

$$start : P@[+(\neg F?[0,+\infty)),+\infty)$$

Real-time constraints can also extend scopes as defined by [Gruhn and Laue, 2005]. In fact scope boundaries can be easily generalized as a given amount of time before or after a qualitative event. The scope "*After Q*" can be extended as "*After k time instants after/before Q*". This can be useful to model "*the airbag system is ready, after 10 seconds the car engine has started*". The extension is a generalization, since the present scopes as defined by [Dwyer et al, 1999] are still modelled by applying $k = 0$. In Figure 9, an example of real-time scope "*After Q + k*" is depicted.

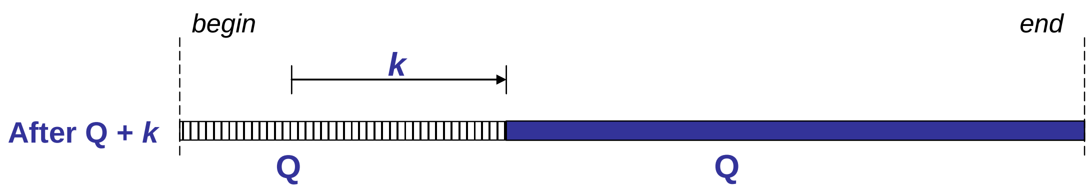


**Figure 9 – Real-time scope *After Q + k***

TILCO-X allows modelling simply the real-time extension of scope. Let us write TILCO-X specification for Universality Pattern on "*After Q*" scope and on "*After Q + k*".
"*After Q, P holds*" can be imposed by asserting $P@[+Q,+\infty)$ at process start. Similarly, when "*After k time instants after Q, P holds*" is imposed, the previous formula can be changed in $P@[+(Q@-k),+\infty)$ or $P@[k,+\infty)@+Q$.

## 7 Case study: Crossroad Traffic-light controller

In this section, a case study is presented aiming at giving evidence about the application of the above mentioned patterns with quantitative bounds. The case study is introduced presenting the main requirements and on the basis of their analysis a pattern based specification has been performed.

### *7.1 Requirements: Crossroad Traffic-light controller*

The Crossroad Traffic-Light controller has to manage two roads with 4 traffic-lights (one for each direction). The traffic-lights can be modelled as if two, since the opposite ones may be coupled because they follow the same behaviour. Besides, the considered Crossroad Traffic-Light controller has to provide: (i) a switch on order to disable (re-enable) the traffic-lights for maintenance reasons; (ii) two emergency signals (one for each road) to request the turn on of the green light for the corresponding road as soon as possible (e.g., for rescue or for police).

An informal description of the system can be summarized as:

1. Normally, the car flow alternates between the two roads and the nominal cycle of a traffic light follows the sequence green-yellow-red-red-yellow. As soon as the green light is shown to a road, the red light must be shown to the other. Yellow light and red-yellow configuration are warning messages that respectively communicate "*attention, a red light is about to arrive*" and "*be ready, a green light is about to arrive*". The normal sequence respects some time constants for changing the lights from one status to another.
2. In the disabled mode, the yellow lights only start blinking on both roads. Disable mode activation waits for the completion of the normal sequence until red is shown on both roads. To enable the system will start the sequence from the same situation.
3. in case of emergency from road A, the light sequence changes as soon as possible into green light for road A, while respecting safety conditions (i.e., minimum time of green and yellow lights for the opposite road). The green light is shown as long as the emergency status on road A holds, then the normal cycle is resumed. Emergency on road A has higher priority than road B and both ones have higher priority than enable/disable switch.

This case study requires a relevant usage of specification patterns for expressing the system real-time properties. For example the most trivial safeness condition to be satisfied is expressed by the first requirement: "*Globally, it is never the case that green1 and green2 holds*". Therefore the Absence Pattern is used in the Global scope.

### *7.2 Properties: Crossroad Traffic-light controller*

In this section, a set of properties of the systems under analysis is reported. For each of them, the textual description/requirement is reported together with the TILCO-X specification by using the above mentioned patterns. Furthermore, for each pattern the name and the scope are reported as well, so as to make the pattern usage more self-evident.

When focusing on real-time behaviour, several requirements can be highlighted They have been associated with behavioural description. In the following expressions, (i) the enable/disable switch is monitored with two Boolean variables representing events: *on, off*; (ii) emergency conditions are represented by $eme_A$ and $eme_B$; (iii) the status of lights is simply represented by six Boolean variables named $green_A$, $yellow_A$, etc. Moreover, Boolean function *up(x)* is true when *x* changes from *false* to *true*.

The main part of the specification consists in formalizing the light sequence which controls the car traffic on both roads. This sequence exposes real-time requirements in terms of time constants that rule the sequence: if yellow light is shown on a road, than red light cannot be lit until some reasonable time is elapsed.

| "*Globally, if up($yellow_A$) holds, not $red_A$ holds for at least YELLOWMIN*" | |
|---|---|
| $start : ((\neg yellow_A @ -1 \wedge yellow_A) \to \neg red_A @ (0, YELLOWMIN)) @ [0, +\infty)$ | |
| Pattern: *Time-Constrained Invariance* | Scope: *Globally* |

The same specification can be applied to green lights, which have to hold for a given time. Furthermore, liveness property can be added, so as to ensure the light sequence is followed in no emergency conditions. The following properties express bounds for green light duration.

| "*Globally, once $green_A$ becomes true, it holds for at least GREENMIN*" | |
|---|---|
| $start : ((\neg green_A @ -1 \wedge green_A) \to green_A @ (0, GREENMIN)) @ [0, +\infty)$ | |
| Pattern: *Minimum Duration* | Scope: *Globally* |

| "*In the absence of $eme_B$, once $green_B$ becomes true, it holds for at most GREENMAX*" | |
|---|---|
| $start : \neg eme_B @ [0, +\infty) \to$ $((\neg green_B @ -1 \wedge green_B) \to \neg green_B ? (0, GREENMAX)) @ [0, +\infty)$ | |
| Pattern: *Maximum Duration* | Scope: *In the absence of F* |

Another interesting property is as follows: if a green light on a road occurs, the same event occurred before in the opposite road. The real-time expression highlights a safety condition regarding how fast the light sequence is.

| "*After on until off, $green_A$ precedes $green_B$ between SAFESWITCH and ∞*" |
|---|

| | |
|---|---|
| $rule : \left( green_B \rightarrow \left( \begin{pmatrix} true\,?[-\infty,-on) \wedge \\ green_A\,?[-on,-SAFESWITCH) \end{pmatrix} \vee \begin{pmatrix} true\,?[-on,-\infty] \wedge \\ green_A\,?[-\infty,-SAFESWITCH) \end{pmatrix} \right) \right) @[+on,+off)$ <br> $rule : (green_B \rightarrow green_A\,?[-on,-SAFESWITCH))@[+on,+off)$ | |
| Pattern: *Time-Constrained Precedence* | Scope: *After Q until R* |

Please note that the Time-constrained Precedence is used in a specific case where one of the bounds is undefined (-∞). The expression is subsequently simplified and a hybrid interval is obtained. The second statement describes what it is expected by the traffic-light in the disable mode.

The most simple property is expressed to verify whether the yellow light is really blinking at a given rate.

| | |
|---|---|
| "*After off until on, yellow$_A$ and yellow$_B$ holds at least every BLINK*" | |
| $rule : (true\,?[BLINK,+on) \rightarrow (yellow_A \wedge yellow_B)?[0,BLINK))@[+off,+on)$ | |
| Pattern: *Time-Constrained Recurrence* | Scope: *After Q until R* |

Please note that hybrid interval used with "*?*" and with a "*true*" Boolean expression means if such interval exists. In this case, on the left side of the implication, if *on* occurs after *BLINK* time instants, then both yellow lights will hold together before BLINK time instants.

Properties can verify the correct enabling and disabling of the traffic-light system.
The disable request can be executed immediately or after the proper sequence has been completed, so a time constant can be the upper limit of this waiting time. This property can be evaluated when the disable request is not overwritten by other commands.

| | |
|---|---|
| "*In the absence of eme$_A$ or eme$_B$, yellow$_A$ and yellow$_B$ responds to off between 0 and FULLCYCLE*" | |
| $start : \neg(eme_A \wedge eme_B)@[0,+\infty) \rightarrow$ <br> $(off \rightarrow (yellow_A \wedge yellow_B)?(0,FULLCYCLE))@[0,+\infty)$ | |
| Pattern: *Time-Constrained Response* | Scope: *In the absence of F* |

The property about enabling the traffic-light has to ensure that the traffic is stopped as the initial condition on both roads and for at least a given time.

| | |
|---|---|
| "*Globally, if on holds, red$_A$ and red$_B$ holds at least for STARTUP*" | |
| $start : (on \rightarrow (red_A \wedge red_B)@(0,STARTUP))@[0,+\infty)$ | |
| Pattern: *Time-Constrained Invariance* | Scope: *Globally* |

The presence of the emergency condition led to identify a number of properties that have to be specified on the traffic-light system. The first one is about how quickly the emergency request has to be fulfilled. In this case two constraints are active at the same time (safeness and speed).

| |
|---|
| "*After on, green$_A$ responds to (eme$_A$ and green$_B$) between SAFESWITCH and EMERGENCY*" |

| $start : (eme_A \wedge green_B \rightarrow green_A ?[SAFESWITCH, EMERGENCY)) @ [+on, +\infty)$ | |
|---|---|
| Pattern: *Time-Constrained Response* | Scope: *After Q* |

This example highlights how simple is to express properties with nesting interval operators and with the usage of hybrid intervals. Furthermore, Time-Constrained Precedence and Response are extremely reusable in their "generalized" form (between $k_{min}$ and $k_{max}$), since more specific cases are directly simplified towards more readable expressions. Since all the expressions are clearly structured, they can be changed by replacing sub-parts in a safe manner, with no risk of changing the time behaviour requirements.
The expression that is produced for "*After on, green$_A$ responds to (eme$_A$ and green$_B$) between SAFESWITCH and EMERGENCY*" would require a more complex form by using single-sided Time-Constrained Response as proposed in [Konrad and Cheng, 2006]. With their pattern, the above specified property has to be split in two: "*After on, green$_A$ responds to (eme$_A$ and green$_B$) after at most EMERGENCY time units*" and "*After on, if (eme$_A$ and green$_B$) hold, then not green$_A$ holds for at least SAFESWITCH time instants*". The used patterns are respectively Time-Constrained Response and Time-Constrained Invariance.

## 8 Conclusions

This paper reported a review of the state of the art for real-time specification patterns, so as to organise the latter in a more systematic manner while providing some new patterns that complete the unified model. The analysis of the state of the art patterns has been performed with the aim of deriving a unified organisation of the results proposed in the above mentioned literature. The proposed organization is based on the effective behavioural relationships among patterns. The identified relationships have been discussed and formally proved along the paper. The formalization of the behavioural organisation allowed us to identify new and renovated patterns such as: Time-Constrained Precedence and Time Constrained Response that complete the unified model.
The proposed behavioural organisation of patterns can be used to provide organised examples on the usage of formal methods in many different notations with respect to the same cases. Therefore, the user can reduce the time to understand if a formal model can be used for modelling the cases under specification, shortening the specification time, reusing and composing different patterns for the specification of more complex problems, thus producing specifications easier to be understood since referred to commonly known patterns.
Besides, during the formalisation of patterns a particular attention has been given in separating the specification of the pattern scope from the description of the pattern behaviour in order to make them re-usable. A more general model for the scope has been provided.
During the presentation the patterns have been formalised in TILCO-X, while in Appendix 1 a list of patterns with formalizations in several logics such as TILCO, LTL, CTL, GIL, QRE, MTL, TCTL, RTGIL is offered disguised as links to the locations which they can be recovered from or are directly reported, when they turned out to be not accessible from literature; this provides the reader with a detailed review of the whole classified patterns, including the ones which were added. In addition, an example has been proposed to give evidence about the usefulness of the new identified patterns that complete the unified model.

Examples of the pattern usage have been proposed to give evidence about the usefulness of the new identified patterns completing the unified model and to compare different specifications provided by authors and in the literature in different temporal logics.
In the end we have also shown that TILCO-X allows to formalise all the patterns proposed in [Dwyer et al, 1999] and in [Konrad and Cheng, 2006]. Furthermore, in some cases, the specifications resulted quite simple thanks to the presence of (i) a uniform management of past and future, (ii) Dynamic Interval operator, (iii) Bounded Happen operator, (iv) interval operator.

## Appendix 1
## (to be included as paper Appendix or made accessible as a WEB page)

What follows is the complete list of Property Patterns. New material only has been presented. The Pattern Template parts which are missing are totally reused from what has been presented in [Dwyer et al, 1999].

### Occurrence Patterns

**Pattern Name and Classification**
**Absence:** Occurrence Specification Pattern

**Temporal Logic Mappings**
**TILCO-X:**

| | |
|---|---|
| Globally: | $start : \neg P \, @[0,+\infty)$ |
| Before R: | $start : R \, ?(0,+\infty) \rightarrow \neg P \, @[0,+R)$ |
| After Q: | $start : \neg P \, @[+Q,+\infty)$ |
| Between Q and R: | $rule : R \, ?(0,+\infty) \rightarrow \neg P \, @[+Q,+R)$ |
| After Q until R: | $rule : \neg P \, @[+Q,+R)$ |

**LTL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/ltl.shtml
**CTL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/ctl.shtml
**GIL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/gil.shtml
**QRE:** http://patterns.projects.cis.ksu.edu/documentation/patterns/qre.shtml

**Pattern Name and Classification**
**Universality:** Occurrence Specification Pattern

**Temporal Logic Mappings**
**TILCO-X:**

| | |
|---|---|
| Globally: | $start : P \, @[0,+\infty)$ |
| Before R: | $start : R \, ?(0,+\infty) \rightarrow P \, @[0,+R)$ |
| After Q: | $start : P \, @[+Q,+\infty)$ |
| Between Q and R: | $rule : R \, ?(0,+\infty) \rightarrow P \, @[+Q,+R)$ |
| After Q until R: | $rule : P \, @[+Q,+R)$ |

**LTL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/ltl.shtml
**CTL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/ctl.shtml
**GIL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/gil.shtml
**QRE:** http://patterns.projects.cis.ksu.edu/documentation/patterns/qre.shtml

**Pattern Name and Classification**
**Existence:** Occurrence Specification Pattern

**Temporal Logic Mappings**
**TILCO-X:**

| | |
|---|---|
| Globally: | $start : P \, ?[0,+\infty)$ |
| Before R: | $start : R \, ?(0,+\infty) \rightarrow P \, ?[0,+R)$ |
| After Q: | $start : Q \, ?(0,+\infty) \rightarrow P \, ?[+Q,+\infty)$ |

Between Q and R: $rule : true\,?[+Q,+R) \wedge R\,?(0,+\infty) \rightarrow P\,?[+Q,+R)$

or $rule : Q \wedge \neg R \wedge R\,?(0,+\infty) \rightarrow P\,?[0,+R)$

After Q until R: $rule : true\,?[+Q,+R) \rightarrow P\,?[+Q,+R)$

or $rule : Q \wedge \neg R \rightarrow P\,?[0,+R)$

**LTL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/ltl.shtml
**CTL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/ctl.shtml
**GIL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/gil.shtml
**QRE:** http://patterns.projects.cis.ksu.edu/documentation/patterns/qre.shtml

**Pattern Name and Classification**
**Bounded Existence:** Occurrence Specification Pattern

**Temporal Logic Mappings**
**TILCO-X:**

Globally: $start : P\,?_{\min}^{\max}[0,+\infty)$

Before R: $start : R\,?(0,+\infty) \rightarrow P\,?_{\min}^{\max}[0,+R)$

After Q: $start : Q\,?(0,+\infty) \rightarrow P\,?_{\min}^{\max}[+Q,+\infty)$

Between Q and R: $rule : true\,?[+Q,+R) \wedge R\,?(0,+\infty) \rightarrow P\,?_{\min}^{\max}[+Q,+R)$

or $rule : Q \wedge \neg R \wedge R\,?(0,+\infty) \rightarrow P\,?_{\min}^{\max}[0,+R)$

After Q until R: $rule : true\,?[+Q,+R) \rightarrow P\,?_{\min}^{\max}[+Q,+R)$

or $rule : Q \wedge \neg R \rightarrow P\,?_{\min}^{\max}[0,+R)$

**LTL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/ltl.shtml
**CTL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/ctl.shtml
**GIL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/gil.shtml
**QRE:** http://patterns.projects.cis.ksu.edu/documentation/patterns/qre.shtml

**Relationships**
It can be considered as a generalization of Existence patterns, since the latter can be obtained by substituting *min* and *max* with 1 and ∞.

**Pattern Name and Classification**
**Time-Constrained Recurrence:** Real-Time Occurrence Specification Pattern
"*P holds at least every k*"

**Temporal Logic Mappings**
TILCO-X:

Globally: $start : P\,?[0,k)\,@[0,+\infty)$

Before R: $start : R\,?(0,+\infty) \rightarrow (true\,?[k,+R) \rightarrow P\,?[0,k))\,@[0,+R)$

After Q: $start : P\,?[0,k)\,@[Q,+\infty)$

Between Q and R: $rule : R\,@(0,+\infty) \rightarrow (true\,?[k,+R) \rightarrow P\,?[0,k))\,@[+Q,+R)$

After Q until R: $rule : (true\,?[k,+R) \rightarrow P\,?[0,k))\,@[+Q,+R)$

**MTL**: see Bounded Recurrence in [Konrad and Cheng, 2006]
**TCTL**: see Bounded Recurrence in [Konrad and Cheng, 2006]
**RTGIL**: see Bounded Recurrence in [Konrad and Cheng, 2006]

## Duration Patterns

**Pattern Name and Classification**
**Minimum Duration:** Real-Time Occurrence Specification Pattern
"*once P becomes true, it holds for at least k*"

**Temporal Logic Mappings**
**TILCO-X:**
Globally: $start : ((\neg P @ -1 \wedge P) \rightarrow P @(0,k))@[0,+\infty)$
Before R: $start : R ?(0,+\infty) \rightarrow ((\neg P @ -1 \wedge P) \rightarrow P @(0,k))@[0,+R)$
After Q: $start : ((\neg P @ -1 \wedge P) \rightarrow P @(0,k))@[Q,+\infty)$
Between Q and R: $rule : R @(0,+\infty) \rightarrow ((\neg P @ -1 \wedge P) \rightarrow P @(0,k))@[+Q,+R)$
After Q until R: $rule : ((\neg P @ -1 \wedge P) \rightarrow P @(0,k))@[+Q,+R)$
**MTL**: see [Konrad and Cheng, 2006]
**TCTL**: see [Konrad and Cheng, 2006]
**RTGIL**: see [Konrad and Cheng, 2006]

**Pattern Name and Classification**
**Maximum Duration:** Real-Time Occurrence Specification Pattern
"*once P becomes true, it holds for at most k*"

**Temporal Logic Mappings**
**TILCO-X:**
Globally: $start : ((\neg P @ -1 \wedge P) \rightarrow \neg P ?(0,k))@[0,+\infty)$
Before R: $start : R ?(0,+\infty) \rightarrow ((\neg P @ -1 \wedge P) \rightarrow \neg P ?(0,k))@[0,+R)$
After Q: $start : ((\neg P @ -1 \wedge P) \rightarrow \neg P ?(0,k))@[Q,+\infty)$
Between Q and R: $rule : R ?(0,+\infty) \rightarrow ((\neg P @ -1 \wedge P) \rightarrow \neg P ?(0,k))@[+Q,+R)$
After Q until R: $rule : ((\neg P @ -1 \wedge P) \rightarrow \neg P ?(0,k))@[+Q,+R)$
**MTL**: see [Konrad and Cheng, 2006]
**TCTL**: see [Konrad and Cheng, 2006]
**RTGIL**: see [Konrad and Cheng, 2006]

## Order Patterns

**Pattern Name and Classification**
**Precedence:** Order Specification Pattern "*S precedes P*"

**Temporal Logic Mappings**
**TILCO-X:**
Globally: $start : (P \rightarrow S ?[-process\_start, 0))@[0,+\infty)$
or $start : \neg P @[0,+S)$
Before R: $start : R ?(0,+\infty) \rightarrow (P \rightarrow S ?[-process\_start, 0))@[0,+R)$
or $start : \neg P @[0,+(S \wedge R))$
After Q: $start : (P \rightarrow S ?[-Q,0))@[+Q,+\infty)$
or $start : \neg P @[0,+S)@+Q$
Between Q and R: $rule : R ?(0,+\infty) \rightarrow (P \rightarrow S ?[-Q,0))@[+Q,+R)$
or $rule : Q \wedge \neg R \wedge R ?(0,+\infty) \rightarrow \neg P @[0,+(S \wedge R))$

After Q until R: $rule:(P \rightarrow S?[-Q,0))@[+Q,+R)$

or $rule: Q \wedge \neg R \rightarrow \neg P\ @[0,+(S \wedge R))$

**LTL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/ltl.shtml
**CTL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/ctl.shtml
**GIL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/gil.shtml
**QRE:** http://patterns.projects.cis.ksu.edu/documentation/patterns/qre.shtml

**Pattern Name and Classification**
**Time-Constrained Precedence**: Real-Time Order Specification Pattern
"*S precedes P between* $k_{min}$ *and* $k_{max}$"

**Temporal Logic Mappings**
**TILCO-X:**

Globally: $$start:\left(P \rightarrow \left(\begin{pmatrix} true?[-k_{max},-process\_start)\wedge \\ S?[-process\_start,-k_{min}) \end{pmatrix} \vee \begin{pmatrix} true?[-process\_start,-k_{max}]\wedge \\ S?[-k_{max},-k_{min}) \end{pmatrix}\right)\right)@[0,+\infty)$$

Before R:
$$start:R?(0,+\infty) \rightarrow \left(P \rightarrow \left(\begin{pmatrix} true?[-k_{max},-process\_start)\wedge \\ S?[-process\_start,-k_{min}) \end{pmatrix} \vee \begin{pmatrix} true?[-process\_start,-k_{max}]\wedge \\ S?[-k_{max},-k_{min}) \end{pmatrix}\right)\right)@[0,+R)$$

After Q: $$start:\left(P \rightarrow \left(\begin{array}{l} (true?[-k_{max},-Q)\wedge S?[-Q,-k_{min}))\vee \\ (true?[-Q,-k_{max}]\wedge S?[-k_{max},-k_{min})) \end{array}\right)\right)@[+Q,+\infty)$$

Between Q and R:
$$rule:R?(0,+\infty) \rightarrow \left(P \rightarrow \left(\begin{array}{l} (true?[-k_{max},-Q)\wedge S?[-Q,-k_{min}))\vee \\ (true?[-Q,-k_{max}]\wedge S?[-k_{max},-k_{min})) \end{array}\right)\right)@[+Q,+R)$$

After Q until R:
$$rule:\left(P \rightarrow \left(\begin{array}{l} (true?[-k_{max},-Q)\wedge S?[-Q,-k_{min}))\vee \\ (true?[-Q,-k_{max}]\wedge S?[-k_{max},-k_{min})) \end{array}\right)\right)@[+Q,+R)$$

**MTL:**

| | |
|---|---|
| Globally: | $\texttt{[]([]}_{<k_{max}-k_{min}}\texttt{!S ->[]}_{=k_{max}}\texttt{ !P)}$ |
| Before R: | $\texttt{<>R->([]}_{<k_{max}-k_{min}}\texttt{!S \& []}_{<k_{max}}\texttt{!R->[]}_{=k_{max}}\texttt{ !P)UR}$ |
| After Q: | $\texttt{[](Q->[]([]}_{<k_{max}-k_{min}}\texttt{!S ->[]}_{=k_{max}}\texttt{ !P))}$ |
| Between Q and R: | $\texttt{[](Q\&!R \& <>R ->([]}_{<k_{max}-k_{min}}\texttt{!S \& []}_{<k_{max}}\texttt{!R->[]}_{=k_{max}}\texttt{ !P)UR)}$ |
| After Q until R: | $\texttt{[](Q\&!R ->([]}_{<k_{max}-k_{min}}\texttt{!S \& []}_{<k_{max}}\texttt{!R->[]}_{=k_{max}}\texttt{ !P)WR)}$ |

**TCTL:**

| | |
|---|---|
| Globally: | $\texttt{AG(AG}_{<k_{max}-k_{min}}\texttt{!S ->AG}_{=k_{max}}\texttt{ !P)}$ |
| Before R: | $\texttt{AFR->A[(AG}_{<k_{max}-k_{min}}\texttt{!S \& AG}_{<k_{max}}\texttt{!R->AG}_{=k_{max}}\texttt{ !P)UR]}$ |
| After Q: | $\texttt{AG(Q->AG(AG}_{<k_{max}-k_{min}}\texttt{!S ->AG}_{=k_{max}}\texttt{ !P))}$ |
| Between Q and R: | $\texttt{AG(Q\&!R\& AFR ->}$ $\texttt{A[(AG}_{<k_{max}-k_{min}}\texttt{!S \& AG}_{<k_{max}}\texttt{!R->AG}_{=k_{max}}\texttt{ !P)UR])}$ |
| After Q until R: | $\texttt{AG(Q\&!R ->A[(AG}_{<k_{max}-k_{min}}\texttt{!S \& AG}_{<k_{max}}\texttt{!R->AG}_{=k_{max}}\texttt{ !P)WR])}$ |

**Relationships**

It is a behavioural generalization of the Precedence pattern, the latter can be obtained from the former by using $k_{\min} = 0$ and $-k_{\max} = -Q$.

**Pattern Name and Classification**
**Response:** Order Specification Pattern
*"S responds to P"*

**Temporal Logic Mappings**
**TILCO-X:**

| | |
|---|---|
| Globally: | $start:(P \rightarrow S?[0,+\infty))@[0,+\infty)$ |
| Before R: | $start:R?(0,+\infty) \rightarrow (P \rightarrow S?[0,+R))@[0,+R)$ |
| After Q: | $start:(P \rightarrow S?[0,+\infty))@[+Q,+\infty)$ |
| Between Q and R: | $rule:R?(0,+\infty) \rightarrow (P \rightarrow S?[0,+R))@[+Q,+R)$ |
| After Q until R: | $rule:(P \rightarrow S?[0,+R))@[+Q,+R)$ |

**LTL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/ltl.shtml
**CTL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/ctl.shtml
**GIL:** http://patterns.projects.cis.ksu.edu/documentation/patterns/gil.shtml
**QRE:** http://patterns.projects.cis.ksu.edu/documentation/patterns/qre.shtml

**Pattern Name and Classification**
**Time-Constrained Response**: Real-Time Order Specification Pattern
"*S responds to P between* $k_{\min}$ *and* $k_{\max}$"

**Temporal Logic Mappings**
**TILCO-X:**
Globally: $start:(P \rightarrow S?[k_{\min},k_{\max}))@[0,+\infty)$

Before R: $start:R?(0,+\infty) \rightarrow \left(P \rightarrow \begin{pmatrix}(true?[+R,k_{\max}) \wedge S?[k_{\min},+R)) \vee \\ (true?[k_{\max},+R] \wedge S?[k_{\min},k_{\max})) \end{pmatrix}\right)@[0,+R)$

After Q: $start:(P \rightarrow S?[k_{\min},k_{\max}))@[+Q,+\infty)$

Between Q and R:

$rule:R?(0,+\infty) \rightarrow \left(P \rightarrow \begin{pmatrix}(true?[+R,k_{\max}) \wedge S?[k_{\min},+R)) \vee \\ (true?[k_{\max},+R] \wedge S?[k_{\min},k_{\max})) \end{pmatrix}\right)@[+Q,+R)$

After Q until R: $rule:\left(P \rightarrow \begin{pmatrix}(true?[+R,k_{\max}) \wedge S?[k_{\min},+R)) \vee \\ (true?[k_{\max},+R] \wedge S?[k_{\min},k_{\max})) \end{pmatrix}\right)@[+Q,+R)$

**MTL**: see Bounded Response in [Konrad and Cheng, 2006], that is a special case of this pattern;
**TCTL**: see Bounded Response in [Konrad and Cheng, 2006], that is a special case of this pattern;
**RTGIL**: see Bounded Response in [Konrad and Cheng, 2006], that is a special case of this pattern;
**Relationships**
In the specification of real-time constraints, the use of "between kmin and kmax" is a generalization with respect to the Bounded Response defined in [Konrad and Cheng, 2006], where one-bound constraint has been used. This Pattern can be obtained by replacing one of

the quantitative boundaries (kmin, kmax) with a qualitative one, that can be “now”, “start of the scope” or “end of the scope”.

**Pattern Name and Classification**
**Time-Constrained Invariance:** Real-Time Order Specification Pattern
“*P activates S holds for at least k*”

**Temporal Logic Mappings**
**TILCO-X:**

| | |
|---|---|
| Globally: | $start : (P \rightarrow S @[0,k))@[0,+\infty)$ |
| Before R: | $start : R?(0,+\infty) \rightarrow (P \rightarrow true[k,+R] \wedge S @[0,k))@[0,+R)$ |
| After Q: | $start : (P \rightarrow S @[0,k))@[+Q,+\infty)$ |
| Between Q and R: | $rule : R?(0,+\infty) \rightarrow (P \rightarrow true[k,+R] \wedge S @[0,k))@[+Q,+R)$ |
| After Q until R: | $rule : (P \rightarrow true[k,+R] \wedge S @[0,k))@[+Q,+R)$ |

**MTL**: see Bounded Invariance in [Konrad and Cheng, 2006]
**TCTL**: see Bounded Invariance in [Konrad and Cheng, 2006]
**RTGIL**: see Bounded Invariance in [Konrad and Cheng, 2006]

## References

[Alavi et al.] SAnToS Laborary, Alavi, H.; Avrunin, G.; Corbett, J.; Dillon, L.; Dwyer M.; Pasareanu, C. Specification Patterns web site http://patterns.projects.cis.ksu.edu/

[Alur and Henzinger, 1990], Alur, R.; Henzinger, T.A. Real-time logics: Complexity and expressiveness. Technical report, Dept. of Comp. Science and Medicine STAN-CS-90-1307, Stanford University, Stanford, California, USA, March, 1990.

[Alur and Henzinger, 1992], Alur, R., and T. A. Henzinger. Logics and models of real time: A survey. In J. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, Real Time: Theory in Practice, Lecture Notes in Computer Science 600, pp.74–106. Springer-Verlag, 1992.

[Alur, 1991], Alur, R., Techniques for automatic verification of real-time systems. PhD thesis, Stanford University, 1991.

[Bellini and Nesi, 2001], Bellini, P.; Nesi, P. TILCO-X an Extension of TILCO Temporal Logic. Proc. of the 7th IEEE Int. Conf. on Engineering of Complex Computer Systems, ICECCS 2001, IEEE Press, Skovde, Sweden pp.15-25, June 2001.

[Bellini, Mattolini and Nesi, 2000], Bellini, P., R. Mattolini, and P. Nesi. Temporal logics for real-time system specification. ACM Computing Surveys, vol.32, n.1, pp.12–42, 2000.

[Bellini, Nesi and Rogai, 2006], Bellini, P., Nesi, P., Rogai, D., Reply to Comments on "An Interval Logic for Real-Time System Specification', Reply to Comments on "An Interval Logic for Real-Time System Specification', IEEE Transactions on Software Engineering, Vol.32, n.6, pp.428-431, June 2006.

[Clarke, Emerson and Sistla, 1986] Clarke, E. M., E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems, Vol.2, pp.244–263, April 1986.

[Douglass, 2003], Douglass,B.P., Real-TimeDesignPatterns. Addison-Wesley, 2003.

[Dwyer et al, 1999], Dwyer, M.B.; Avrunin, G.S.; Corbett, J.C., Patterns in property Specifications for finite-state verification, Proc. of the 1999 IEEE International Conference on Software Engineering, pp.411-420, 1999.

[Felder and Morzenti, 1994], Felder, M.; Morzenti, A. Validating real-time systems by history-checking TRIO Specifications. ACM Transactions on Software Engineering and Methodology. 3-4 Oct. 1994, 308-339.

[Fowler, 1997], Fowler, M., Analysis Patterns: Reusable Object Models, Addison-Wesley, 1997.

[Gamma et al., 1994], Gamma, E., Helm, R., Johnson, R., and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

[Gruhn and Laue, 2005], Gruhn, V., Laue, R., Patterns for timed property specification, Proc. of the 3rd Int. Workshop on Quantitative Aspects of Programming Languages (QAPL 05), Edinburgh, Scotland, April 2005. 2005.

[Gruhn and Laue, 2005b], Gruhn, V., Laue, R., Specification Patterns for Time-Related Properties, Proc. of the 12th International Symposium on Temporal Representation and Reasoning (TIME05), 2005.

[Konrad and Cheng, 2004], Konrad, S., Cheng, B. H. C., and Campbell, L. A., Object analysis patterns for embedded systems. IEEE Transactions on Software Engineering, Vol.30, n.12, pp.970–992, December 2004.

[Konrad and Cheng, 2005], Konrad, S., Cheng, B. H. C., "Real-time specification patterns" Proceedings of the 27th International Conference on Software Engineering (ICSE05), St Louis, USA, May 2005.

[Konrad and Cheng, 2006], Konrad, S., Cheng, B. H. C., "Defining and Using Real-Time Specification Patterns for Embedded Systems", Technical Report of Michigan State University, MSU-CSE-04-37, Revision of March 2006.

[Koymans, 1990], Koymans, Specifying real-time properties with metrics temporal logic, Real Time Systems, Vol.2, n.4, pp.255-299, 1990.

[Koymans, 1992], Koymans, R. Specifying Message Passing and Time-Critical Systems with Temporal Logic.Lecture Notes in Computer Science 651, Springer-Verlag, 1992.

[Manna and Pnueli, 1992], Manna, Z., and A. Pnueli. The temporal logic of reactive and concurrent systems. Springer-Verlag New York, Inc., 1992.

[Mattolini and Nesi, 2001], Mattolini, R.; Nesi, P., "An interval logic for real-time system specification", IEEE Transactions on Software Engineering, pp.208-227, 2001

[Moser et al., 1997], Moser, L.E., Y. S. Ramakrishna, G. Kutty, P. M. Melliar-Smith, and L. K. Dillon. A graphical environment for the design of concurrent real-time systems. ACM Transactions on Software Engineering and Methodology, vol.6, n.1, pp.31–79, 1997.

[Shaw, 1996], Shaw, M., Some Patterns for Software Architecture, Pattern Languages of Program Design, Vol.2, pp.255-269, 1996.