Expressing and Organizing Specification Patterns with TILCO-X

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: 31/08/2006

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 in learning and usage of formal methods and tools. The introduction and usage of patterns has a double impact. They are examples of covering classical problems with formal methods in many different notations, so that the user can shortening the time to understand if a the formal method can be used for covering his purpose and how. In addition, they are used for shortening the specification time, reusing and composing different patterns for covering the specification, thus producing more understandable specifications referring to commonly known patterns. For these reasons, the interests and the usage of patterns are growing, thus a number of proposals for patterns and pattern classification/organisation have been presented in literature. In this paper, the analysis of the state of the art for real-time specification patters is presented in order to organise them in a systematic and uniform manners, providing the missing models. The proposed organization is based on the effective behavioural relationships among the patterns. In addition, during the discussion the patterns have been formalised in TILCO-X (an extended version of TILCO temporal interval logic), offering in this manner an additional formalism to be compared with those already publicly acessible in the literature for exposing patterns.

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 are in many cases modelled by using suitable specification techniques, which allow the verification and validation of the specified behaviour. For their specification, a set of formalisms for defining relationships expressing temporal constraints among events are used – for example: properties of invariance, ordering among events, periodicity, liveness and safety conditions, etc.

To cope with the above problems, in many cases, formal models have been used as requirements analysis and/or specification techniques. The selected methods/formalisms are in many cases 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 the requirements analysis and specification of real time systems, those 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 reasonable efficiency only by accurately trained people.
Recently the presentation of specification patterns for temporal logics has improved their usability. They can be used for training and guiding analysts and developers to express requirements and specifications directly in a formal language [Dwyer et al, 1999], thus shortening the specification time and producing specifications which are more understandable since refer to common well-known design patterns.
In [Dwyer et al, 1999], a set of specification patterns have 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. Those specification patterns were mainly focussed on formalising specification properties such as occurrence and ordering of events. 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. Thus, similarly organised requirements may be formalised by using the same specification pattern.

In the literature, a wide work has been performed to identify many kind of pattern to better formalise and shortening the 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 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 the patterns that helps the user to understand the patterns. Patterns with real time properties have been also discussed in [Gruhn and Laue, 2005] considering the pattern structure (e.g., scope, behaviour and events or occurrences) and presenting patterns models/mappings by means of timed automata.

In [Konrad and Cheng, 2005], in order to allow the specification of real time patterns, the temporal logics selected were in most cases quantitative extensions of the logics used in [Dwyer et al, 1999]. On the other hand, observing the state of the art of temporal logics, only a few of them present a metric of time; that is the possibility of expressing temporal constraints in a quantitative manner [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].

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 former in [Bellini et al, 2001], [Mattolini and Nesi, 2001] and [Bellini, Nesi and Rogai, 2006]. All of them have a metric of time and therefore they can be profitably 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 interesting operators that make the specification quite compacts. Therefore, we think very interesting to present TILCO specification of the proposed patterns.

TILCO has been designed for the specification of real-time systems, and 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 eventually and henceforth to time intervals [Mattolini and Nesi, 2001]. TILCO allows defining expressions of ordering relationships among events, delays, time-outs, periodicity, liveness and safety conditions, etc. The TILCO logics and theory have been further developed so as to create more powerful formalisms; such as TILCO-X [Bellini and Nesi, 2001A] and CTILCO [Bellini and Nesi, 2001B]. TILCO-X extends TILCO introducing the new syntax and semantics of major TILCO (dynamic intervals and bounded happen). This extension allows both removing *since* and *until* TILCO operators and writing simple predicates, including counting of events that may occur in intervals. CTILCO supports process composition and decomposition by allowing the specification of communicating TILCO processes.

In this paper, the analysis of the state of the art for real-time specification patterns is presented in order to organise them in a systematic and uniform manner. The proposed organization is based on their classification and considers behavioural relationships among the patterns. These relationships are discussed and proved along the paper. 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, so that the user can shortening the time to understand if a the formal model can be used for modelling the cases under specification. In addition, 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 exploiting specification patterns is mainly conceived as collecting and providing organised properties ready to be used.
During the presentation of pattern organisation and analysis of their relationships, the pattern mappings have been formalised in TILCO-X [Bellini and Nesi, 2001] (an extended version of TILCO temporal interval logic [Mattolini and Nesi, 2001]), offering in this manner an additional formalism to be compared with those already used in the literature for exposing patterns. With TILCO-X is possible to formalise all the patterns proposed in [Dwyer et al, 1999] and in [Konrad and Cheng, 2006]. Moreover, in some cases, the specification provided resulted quite compact and concise due to the presence of some specific operators in TILCO-

X with respect to other specification models; namely: the uniform management of past and future, dynamic internal operator, banded happen operators. For this reason a short overview of TILCO-X is also reported in the paper. During the formalisation of patterns in TILCO-X a particular attention has been given in separating the specification of the pattern scope with respect to the description of the pattern behaviour. In this manner, we think that the patterns proposed result to be more re-usable. A comparison of the patterns produced in TILCO-X with respect to those accessible from the literature and produced in other formalisms is presented in Appendix 1.

The paper is organised as follows. Section 2 presents an overview of TILCO-X temporal logic with its major operators and formalisms, this will allow us to use the TILCO-X for reasoning on patterns and scopes. In Section 3, an overview of qualitative and quantitative specification patterns is presented. In Section 4, a discussion on organisation of the patterns and on the scopes is reported. Section 5 presents both qualitative and quantitative patterns specified in TILCO-X together with their relationships and related demonstrations. A discussion on patterns' scope is offered in Section 6. Conclusions are drawn in Section 7. Appendix 1 reports all the patterns considered and references to specifications of the same patterns that can be accessed from the literature.

2 TILCO-X overview

TILCO 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 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 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 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 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;

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. The first version of TILCO [Mattolini and Nesi, 2001] adopted since and until operators for the same purpose. These operators make a strong distinction between past and future and, subsequently, their adoption often makes the specification complex and difficult to read. The adoption of a unique operator, as in TILCO-X, for defining ordering relationships among events reduces in many cases the need of adopting nested since and until operators.
Specifying the occurrence of one event with respect to a number of occurrences of another event is a situation that arises 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. Specifications with these constraints are quite complex to understand and difficult to be realized with classical temporal operators [Bellini, Mattolini and Nesi, 2000]. This is the reason why some temporal logics present specific operators for this purpose. These constraints can be specified in FOL and therefore also in first order temporal logic; the specification turns out to be complex with respect to the complexity of the concept being under specification and it involves quantifications.

TILCO-X temporal logic has been defined by extending TILCO so as to enhance its readability and conciseness, especially when it comes to express order relationships, thus canceling the distinction between past and future and generalizing the Since and Until concepts. To this end, operators called Dynamic Intervals and Bounded Happen have been included in TILCO-X. They can be combined to allow defining complex real-time constraints by using a small number of operators.

Please note that the semantics and the deductive system of those operators and therefore of TILCO-X logic are reported in [Bellini and Nesi, 2001A]. TILCO-X presents a different semantics and deductive system with respect to TILCO. This has been mandatory, since the new operators cannot be defined in terms of the previous TILCO operators. On the contrary, since and until operators can be defined in terms of the new TILCO-X operators, as presented in the next section.

2.1 TILCO-X Dynamic Intervals

Dynamic Intervals have been introduced to avoid any needs 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 allows the combination of ordering and quantitative relationships. These capabilities were introduced in TILCO-X by allowing one to write temporal intervals, not only as constant integer sets, and also by using formula as an interval bound. 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 an 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 in which predicate A has to be true.

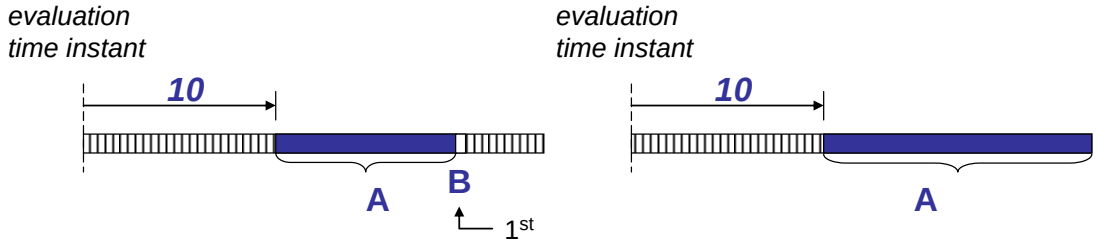

Figure 1: Example of Dynamic Interval: $A@[10,+B)$

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.

Noted that, 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.

The following specification is a typical case in which a strong distinction between past and future has to be performed to adopt since and until operators: since the last occurrence of C and until the first occurrence of D, for every occurrence of A there will be an occurrence of B at the same time. In TILCO, it can be formalized as follows:

$$(\text{since}\, C\, (A \rightarrow B)) \wedge (A \rightarrow B) \wedge (\text{until}\, D\, (A \rightarrow B))$$

With TILCO-X, writing 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. The following TILCO-X formula provides another such example:

$$A\,?[+B,+C]$$

This formula states that *A* happens between the next occurrence of *B* and the next occurrence of *C*. The interval boundaries are included; therefore, A may happen even at the same time instant as *B* or *C*. Without the new construct the same formula would have been written using two nested until operators.

$$B\,?(0,+\infty) \wedge (\text{until}\, B\, \neg C) \wedge \text{until}(B \wedge (A \vee \neg(\text{until}(C \wedge \neg A)\, \neg A)))(\neg B)$$

B must happen in the future, otherwise the interval $[+B,+C]$ is empty and *A* cannot happen in an empty interval. Explicit temporal quantifications are not allowed in TILCO language, as demonstrated in [Alur and Henzinger, 1990]. On such grounds they have been excluded as a necessary condition to have automated verification mechanisms [Mattolini and Nesi, 2001]. For these reasons, in the early version of TILCO, it was complex to specify certain constraints without the adoption of a direct temporal quantification. This problem has been solved with TILCO-X that allows writing complex ordering relationships among events, especially those combining order and quantitative relationships.

2.2 TILCO-X Bounded Happen

Bounded Happen has been defined to increase constraint readability which includes the dependency on the counting of occurrences. Sometimes a constraint implies counting the number of an event occurrences or in general how many times a formula is true in a given time interval.

Bounded Happen can be 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 TILCO-X formula

$$A?^3[1,15)$$

states that *A* is true up to three times in interval $[1,15)$.

By combining such operators, it can be stated that: a formula has to be true in the interval from a minimum to a maximum number of times; as it is shown with the following example:

$$A?_2^3[1,15)$$

Bounded happen can be used with the *Dynamic Interval operator*. The following formula states that *A* happens two or three times from now until *B* happens (see Figure 2):

$$A?_2^3[0,+B)$$

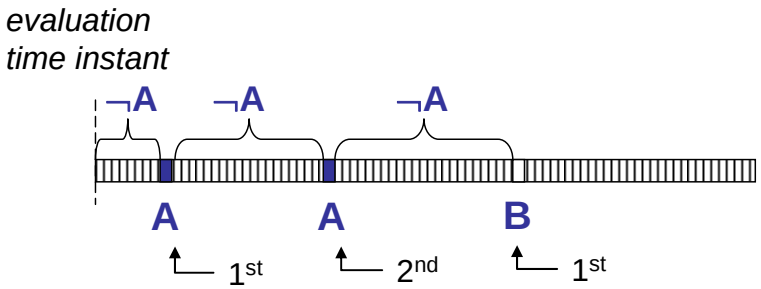

Figure 2 – Example of Bounded Happen: $A?_2^3[0,+B)$

3 Overview of Specification Patterns

In [Dwyer et al, 1999], a classification of specification pattern has been proposed. Then, in [SAnToS] details about qualitative specification patterns are published covering all the typical situations in which a developer may be 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, modelled, mapped with the chosen formalism (formal model in this case);
- **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. On this aim, who is going to formalize the pattern can be more or less interested in making evident the distinction among, 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 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 mapping, 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*.

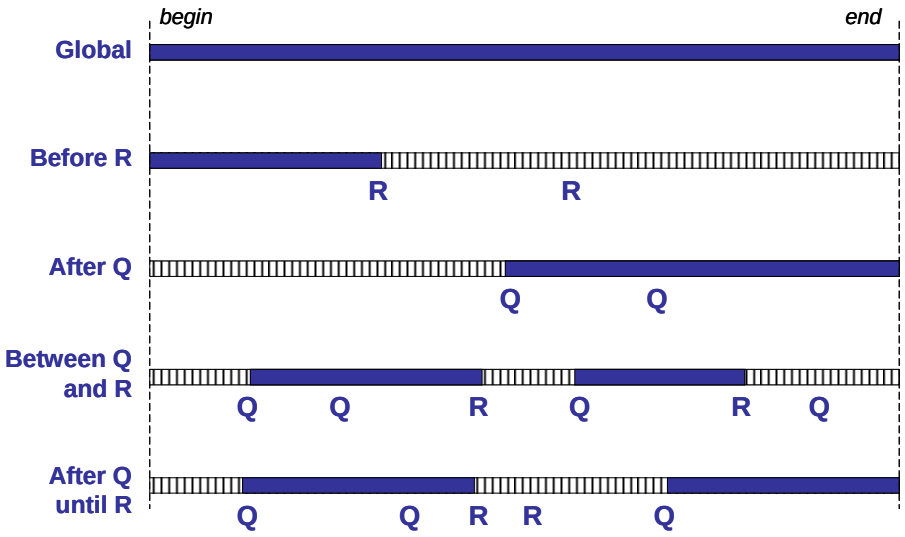

Figure 3 – Pattern Scopes

The experience strongly indicates that in most cases requirements are specified as properties of (i) the whole program execution or of (ii) specific segments of the program execution. Thus a pattern system for properties allows one to specify the system behaviour regarding 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 in which *F* occurs at least once;
- **in the absence of F** – a property has to hold only in an interval in which *F* never occurs;
- **from when F never holds** – a property has to hold only from the state/event in which *F* is going to stay false forever.

Explicit scope operators are not present in most specification formalisms; interval logics can be considered an exception, if the operators to define the interval is used for modelling the scope. Generally, it could be also possible to use scopes which are open on left and/or rights. A discussion on 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 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 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 by 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 regarded as specific cases 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 intended in that manner. For this reason in the following they have not been reproduced in TILCO-X.

In [Dwyer et al, 1999], proposed patterns are defined in terms of what happen in the future and never on what has been occurred in the past. The decision of presenting only patterns referring to the future may be due to the formalisms used. In the presentation of TILCO-X pattern in some cases both approaches have been offered, 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 have 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 in this paper)**.** Limits the period within which a given occurrence has to happen. *P* occurs every *t* time instants;
- **Real Time Order Patterns** are used to express requirements related to formalising patterns in which the time duration among events 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 in which a formula is true until another formula become true;
    - **Bounded Invariance** (called Time-Constrained Activation in the classification proposed in this paper)**.** Limits the minimum time duration from the event/state in which a formula is true once another formula is true.

Please note that, in the set of patterns proposed in [Konrad and Cheng, 2005] the term bounded is used for describing a bound in time, while in [Dwyer et al, 1999] the same terms is used to refer to a limit in the number of event occurrences. For this reasons, in order to avoid confusion, 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*". In addition, we performed another change in the naming proposed in [Konrad and Cheng, 2005], specifically changing "*Invariance*" with "*Activation*", this will be more clear when the related pattern will be presented.

Analysing the relationships among all the above mentioned patterns several similarities have been identified that convinced us to produce an integrated classification as reported and discussed in the next section.

4 Specification Patterns Organization

In this section, the above mentioned organization of patterns is presented considering both qualitative and quantitative patterns. This approach required a reorganization of the existing pattern catalogue, and an extension of the *scope* concept. It has been chosen to put in strict relation real-time patterns with those already present which do not consider a metric of time.

Differently from [Konrad and Cheng, 2006], in our pattern organisation no radical distinction has been performed from qualitative and quantitative (also called real time) patterns. The organisation proposed in [Konrad and Cheng, 2006] maintained the [Dwyer et al, 1999] hierarchy and added an additional hierarchy for the real time patterns. In turn, for their

purpose, they have been organised grouping together those that may have a common root in terms of pattern description as "structure English". Our organisation is based on a different purpose as described in the following.

In Figure 4, the proposed organisation is presented. At the first layer, the pattern categories are considered as in [Dwyer et al, 1999] and [Konrad and Cheng, 2006]. After that layer, the patterns are grouped according to the categories and present some relationships among them highlighted with empty arrows. The proposed organisation has been created after to have analysed and identified the relationships among patterns and discovered some "behavioural generalisation" among some of them as discussed in the rest of the paper.

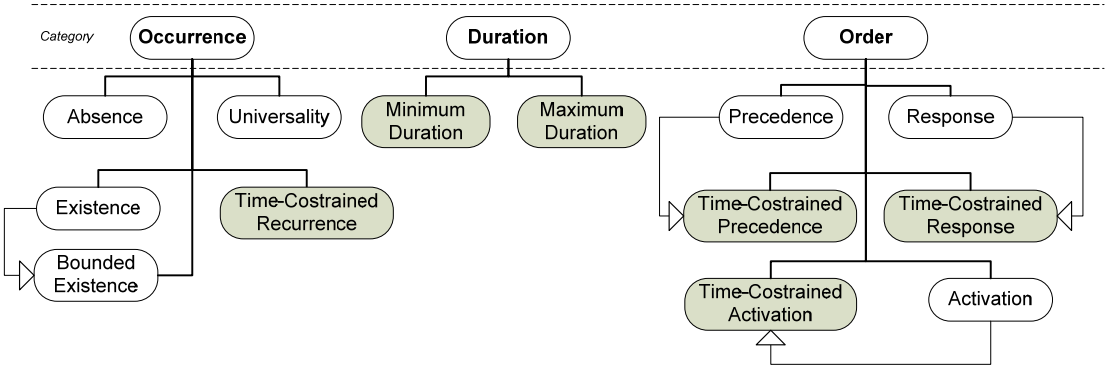

Figure 4 – Pattern hierarchy and relations

The "behavioural generalization" has been used to model the fact that the un-timed properties can be obtained by relaxing the time constraints from the timed properties. For example, 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 simply imposing qualitative time bounds as $k_{min} = 0$ and $k_{max} = \infty$. In the diagram, the generalization is depicted by using a white arrow to represent the "*is a*" relationship.

This generalization has brought a unified hierarchy with respect to what has been defined by [Konrad and Cheng, 2006]. In fact, the categories only distinguish which kind of constraint the pattern is applying to the predicates:

- **Occurrence**: properties which express if a given predicate has to occurs, always, never, periodically or for a given amount of times. It has been defined in [Dwyer et al, 1999].
- **Duration**: properties that without imposing the occurrence, requires 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 relation more predicates, by ordering them. It has been defined in [Dwyer et al, 1999].

In these categories, both qualitative and quantitative patterns are organized while the latter (real time) have been marked in grey to put them in evidence.
The category of Periodic patterns proposed [Konrad and Cheng, 2006] with only one pattern (Bounded Recurrence and called Time-Constrained Recurrence in this paper) has not been

used since the single pattern Time-Constrained Recurrence has been classified as a occurrence pattern – i.e., periodic occurrences.
The category of Real Time Order proposed [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 Activation in this paper) has been fused with that or Order. In effect there exist strong behavioral relationships among them as demonstrated in the following.
In the above hierarchy, Precedence Chain and Response Chain have not been presented, since a chain of events can be considered as an event itself as stated before and in [Gruhn and Laue, 2005].

5 Specification Patterns with TILCO-X

In this section, the patterns organisation reported in the previous Section is discussed presenting the evidence of the relationships among the patterns. 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.

According to the previous discussion, 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], [Dwyer et al, 1999]). In all these cases, the patterns have been presented by:

- Referring to a point in which the process start, nothing happen 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 mapping, we have noticed some differences that make some of the mappings more intuitive and in some how different with respect to those presented for other logics. 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 an 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 defined the start, it is possible to define a rule that impose the validity of the formula from the process start to the time limit (e.g., infinite) $(rule : A) \equiv start\ A@[0,+\infty)$

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

Therefore, one of the contributions of the paper is also to present the TILCO-X formalisation for specifying patterns that can be compared with those presented in other formalisms by following the Appendix 1. In Appendix 1, all patterns discussed in this paper are reported in TILCO-X for all the scopes and for each of them references to other documents or web sites containing the same pattern mappings in other formalisms are reported.

In the following, a description of the patterns considered by following the organization described in the previous Section is reported (please refer to Appendix 1 when an exhaustive view of all formulas for a pattern is needed). The description is focused on presenting and stressing the main relationships among the patterns. Those relationships are in some cases of behavioural specialization as shown in the sequel.

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 Recurrence (called Bounded Recurrence in [Konrad and Cheng, 2006]).

The Absence Specification Pattern aims to describe a portion of a system's execution that is free of certain events or states. Thus, as can be noted observing the Absence (Occurrence) pattern reported below, the scopes are modelled by dynamic intervals. Thus, the TILCO-X mapping appears 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:	$rule : R\,?(0,+\infty) \rightarrow \neg P \,@\left[+Q,+R\right)$
After Q until R:	$rule : \neg P \,@\left[+Q,+R\right)$

The definition of interval-based operators like "$@$", "$?$", and "$?_{\min}^{\max}$" allows to reuse the pattern mappings for all the other Occurrence specification mappings.
In fact to map all the other Occurrence patters on the on the five scopes it is only needed to replace the operators on the left while the scopes remain independently modelled by intervals,

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

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 structuring applies for all the other scopes. TILCO-X operators model in a quite simple manner the occurrence patterns, while leaving to the intervals the definition of the pattern scope, 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.

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, for the same purpose, also to some other patterns. Please note that the "past" semantic of dynamic interval allows identifying the first *Q* of a *Q-R* sequence with a quite simple formula, this would be more complex with only future operators.

Among the Occurrence patters, a relationship of behavioural specialization has been identified. In fact, a model of the Bounded Existence pattern is also 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 patter results to be quite simple with respect to the specifications performed in formalisms that does not present operators for modelling/counting the occurrences (e.g., LTL).

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*".

Observing and comparing those patterns as reported in Appendix 1, it can be noted that the scopes are put in evidence and thus 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 (similar to the Bounded Response in [Konrad and Cheng, 2006]). Activation and Time-Constrained Activation (called Bounded Invariance in [Konrad and Cheng, 2006]).

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 an occurrence of the second. We say that 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; in the following example two different mappings of "*S precedes P*" on "*Between Q and R*" are presented.

- 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 form allows keeping independent scope and pattern intent; the future form uses the dynamic interval with the conjunction of S and R (scope boundary). Furthermore the past form is more readable since it is still recognizable 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.

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.

Similarly to 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:

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

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 begin of the scope and the occurrence of *P*.
Similarly 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.

5.3.3 Activation Pattern

Another pattern for imposing activation property (where an event triggers another to hold) can be defined as Activation. It is related to the Response in the sense that *S* has not only to occur, but to hold until the end of the scope.
Even in this case, TILCO-X operators allow maintaining a well-defined structure. For example, "*P activates S*" on "*Between Q and R*" can be written as

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

Please note that only the TILCO-X operator in the pattern mapping has been changed with respect to Response Pattern. This remarks the powerful of using interval-based operators for modeling Existence or Universality.
The model for pattern "*P activates S at least for k*" is more general and includes the case of "*P activates P*" when *k* is right bound of the scope. This is shown in Section 5.3.7.

5.3.4 Consideration on Order Specification Patterns

Please note that interval-based logic is capable of modelling the scopes in a readable manner, and keeping them separate from the specification of pattern behaviour, since the scope can be viewed as an interval. In fact, in a general way all the Precedence, Response and Activation Specification patterns can be generally expressed by defining "beginning of the scope" and "end of the scope", these event/states need to be expressed like viewed at any time instant inside the scope. The definitions are reported in the following Table.

Scope	**Beginning of scope**		**End of scope**
	w.r.t. process_start	w.r.t time instants inside scope	
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	$+Q$	$-Q$	$+R$
	$scope\_beg$	$scope\_beg\_in$	$scope\_end$

In the last row of the table, some predicates are defined in order to generally indicate scope bounds in expressing TILCO-X mapping independently from the scope.

Precedes:

The general expression of "*S precedes P*" for the first three scopes can be written as

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

The other two scopes potentially define an infinite set of intervals, thus is not possible to obtain expression which are evaluated only at start time instant, while the need of using a "rule" is evident in order to detect any scope realization (i.e. when a *Q*-*R* sequence occurs).

As depicted in Universality Pattern (see Appendix 1), to assert a property P at any time instant after Q until R, it is needed to impose $P@[+Q,+R)$ at a single time instant just before a Q-R sequence; using "rule" the desired expression is obtained and A is asserted in all the Q-R sequences along the time axis.

Thus, the general expression for "*S precedes P*" is still valid for "*After Q until R*", while is 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 existence of the scope (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

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

Activates:

Activation pattern can be represented like Response for the first three scopes as

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

and for "*After Q until R*" and "*Between Q and R*" scopes 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. Thus the result of maintaining the same clear structure for all the mappings, distinguish among scopes and pattern intents it has been considered of great value. This could help in reusing/extending these mappings to easily adapt their formulae to specific behavior. In Appendix 1, for some Patterns, the alternative and simpler formalizations are also reported.

5.3.5 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), use of dynamic interval allow to introduce metric of time for Order Patterns, still maintaining a comprehensible structure.

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

- 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

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

The augmented expression introduced for imposing 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 5, two different conditions are presented, please note

that *S* must precede *P* after the scope boundary if it happens within the requested time duration.

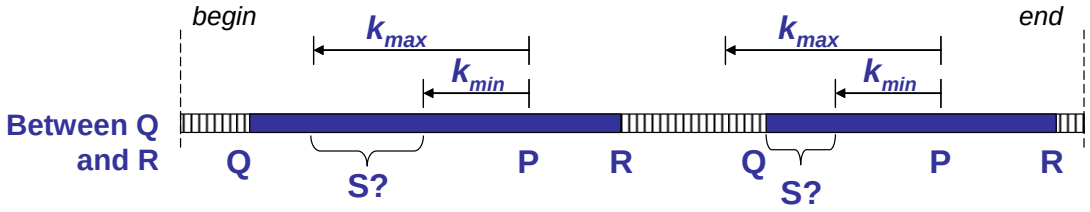

Figure 5 – Scope boundaries and time durations

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 beginning and the end of each scope:

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

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

- "*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 as a special case of Time-Constrained version. In the following, a demonstration of the fact that Time-Constrained Precedence generalizes Precedence is provided. It can be proved that expressing "*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:

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

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

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

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

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

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

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

5.3.6 Time-Constrained Response Pattern

Time-Constrained Response Pattern can be defined in a similar manner to Time-Constrained Precedence Pattern. In this case the right bound of the scope has to be evaluated with respect to k_{max}.

Some example of TILCO-X mappings are given in the following, 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 appears simpler since for this scope there is not needs 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 expressing "*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:

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

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

Thus, the Time-Constrained Response exactly maps 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, that can be "now", "beginning of the scope" or "end of the scope".

5.3.7 Time-Constrained Activation Pattern

Also Activation Pattern is related to the corresponding real-time version: Time-Constrained Activation. 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, like [Konrad and Cheng, 2006], the scope end cannot interrupt the time length in which *S* holds (see Figure 6). The formula $true[k,+R]$ is placed to state that "*R occurs after at least k time instants*".

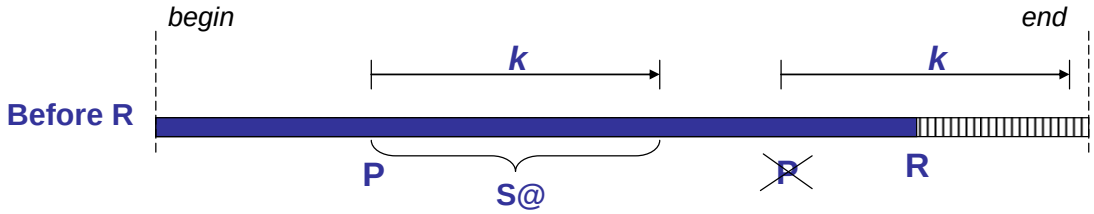

Figure 6 – Examples of Time-Costrained Activation property

The demonstration that this Pattern generalizes the Activation pattern is very similar to those presented for Precedence and Response patterns; thus, it has been left to the reader.

6 Discussion on Pattern Scopes

As above mentioned, in [Konrad and Cheng, 2006], other scopes have been presented. These additional scopes may be specified according to the following constructs, where *P* is modelled as Universality, while can be any other of the above mentioned.

Scope: *in the presence of F* – a property has to hold only in an interval in which *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 in which *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 in which *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 7, an example of real-time scope "*After Q + k*" is depicted.

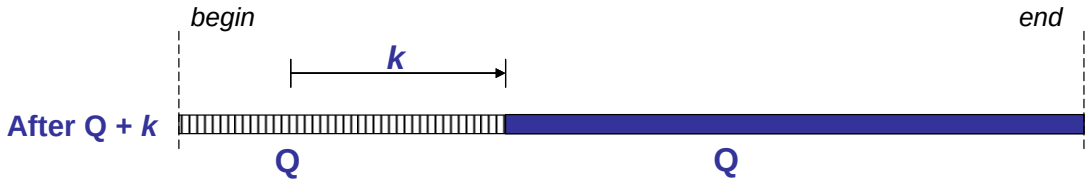

Figure 7 – Real-time scope *After Q + k*

TILCO-X allows to simply modelling the real-time extension of scope. Let us write TILCO-X mappings 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, to impose that "*After k time instants after Q, P holds*" the previous formula can be changed in $P@[+(Q@-k),+\infty)$ or $P@[k,+\infty)@+Q$.

7 Conclusions

In this paper, an analysis of the state of the art about specification patterns in formal logics has been presented. The identified patterns have been organized in a systematic and uniform manner. The proposed organization is based on their classification considering their nature and the behavioural relationships among the patterns. Discussions and demonstrations about the identified relationships among patterns have been reported.
The pattern classification proposed can be used provide organised examples of the usage of formal methods in many different notations with respect to the same cases, so that the user can reduce the time to understand if a the formal model can be used for modelling the cases under specification. In addition, it 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.
During the presentation of pattern organisation and analysis of their relationships, the pattern mappings have been formalised in TILCO-X (an extended version of TILCO temporal interval logic), offering in this manner an additional formalism to be compared with those already used in the literature for exposing patterns.
It has been shown that with TILCO-X is possible to formalise all the patterns proposed in [Dwyer et al, 1999] and in [Konrad and Cheng, 2006]. Moreover, in some cases, the specification provided resulted quite compact and concise due to the presence in TILCO-X of (i) a uniform management of past and future, (ii) dynamic interval operator, (iii) bounded happen operator, (iv) interval operator;
During the formalisation of patterns in TILCO-X a particular attention has been given in separating the specification of the pattern scope with respect to the description of the pattern behaviour. In this manner, we think that the patterns proposed result to be more re-usable. A comparison of the patterns produced in TILCO-X with respect to those accessible from the literature and produced in other formalisms is presented in Appendix 1.

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

It follows the complete list of Property Patterns. Please note that only the new material has been presented. Those Pattern Template parts which are missing are totally reused from what 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: $start : P?^{max}_{min}[0,+\infty)$

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

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

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

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

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

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

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

Between Q and R:

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

After Q until R:

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

MTL:

Globally: `[]([]`$_{<k_{max}-k_{min}}$`!S ->[]`$_{=k_{max}}$` !P)`

Before R: `<>R->([]`$_{<k_{max}-k_{min}}$`!S & []`$_{<k_{max}}$`!R->[]`$_{=k_{max}}$` !P)UR`

After Q: `[](Q->[]([]`$_{<k_{max}-k_{min}}$`!S ->[]`$_{=k_{max}}$` !P))`

Between Q and R: `[](Q&!R & <>R ->([]`$_{<k_{max}-k_{min}}$`!S & []`$_{<k_{max}}$`!R->[]`$_{=k_{max}}$` !P)UR)`

After Q until R: `[](Q&!R ->([]`$_{<k_{max}-k_{min}}$`!S & []`$_{<k_{max}}$`!R->[]`$_{=k_{max}}$` !P)WR)`

TCTL:

Globally: `AG(AG`$_{<k_{max}-k_{min}}$`!S ->AG`$_{=k_{max}}$` !P)`

Before R: `AFR->A[(AG`$_{<k_{max}-k_{min}}$`!S & AG`$_{<k_{max}}$`!R->AG`$_{=k_{max}}$` !P)UR]`

After Q: `AG(Q->AG(AG`$_{<k_{max}-k_{min}}$`!S ->AG`$_{=k_{max}}$` !P))`

Between Q and R:
```
AG(Q&!R& AFR ->
   A[(AG<kmax-kmin!S & AG<kmax!R->AG=kmax !P)UR])
```

After Q until R: `AG(Q&!R ->A[(AG`$_{<k_{max}-k_{min}}$`!S & AG`$_{<k_{max}}$`!R->AG`$_{=k_{max}}$` !P)WR])`

Relationships

It is a behavioural generalization of 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 constrains, 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", "beginning of the scope" or "end of the scope".

Pattern Name and Classification
Activation: Order Specification Pattern
"P activates S"

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:

Globally:	`[](P ->[]S)`
Before R:	`<>R -> ((P -> S U R) U R)`
After Q:	`[](Q -> [] (P -> []S) )`
Between Q and R:	`[](( Q & !R & <>R) -> ((P -> S U R) U R))`
After Q until R:	`[](( Q & !R ) -> ((P -> S W R) W R) )`

CTL:

Globally:	`AG (P -> AG S)`
Before R:	`AFR -> A[(P -> A[S U R]) U R]`
After Q:	`AG(Q -> AG(P -> AG S) )`
Between Q and R:	`AG(Q & !R & AFR -> A[(P -> A[S U R]) U R])`
After Q until R:	`AG(Q & !R -> A[(P -> A[S W R]) W R] )`

Pattern Name and Classification
Time-Constrained Activation: 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]
Relationships
It can be considered as a generalization of Activation patterns since the latter can be obtained from the former for *k* equal to +∞.

References

- 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, R., Techniques for automatic verification of real-time systems. PhD thesis, Stanford University, 1991.
- 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.
- 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.
- 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, P.; Giotti, A.; Nesi, P. Execution of temporal logic specifications Proc. of the 8th IEEE Int. Conf. on Engineering of Complex Computer Systems, ICECCS 2002, IEEE Press, GreenBelt, Maryland, pp.78-88, December 2002..
- Bellini, P.; Giotti, A.; Nesi, P.; Rogai D., TILCO Temporal Logic for Real-Time Systems Implementation in C++, Proc. of the 15th Int. Conf. on Software engineering and knowledge engineering, SEKE03, ACM press, San Francisco Bay, June 2003.
- Bellini, P.; Nesi, P. Communicating TILCO: a Model for Real-Time System Specification. Proc. of the 7th IEEE Int. Conf. on Engineering of Complex Computer Systems, ICECCS 2001, IEEE Press, Skovde, Sweden, pp.4-14, June 2001, (B)
- 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, (A)
- 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,B.P., Real-TimeDesignPatterns. Addison-Wesley, 2003.
- 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, 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, M., Analysis Patterns: Reusable Object Models.Addison-Wesley, 1997.
- Gamma, E., Helm, R., Johnson, R., and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.
- Gruhn, V., Laue, R., Patterns for timed property specification, Prof of the 3rd Int. Workshop on Quantitative Aspects of Programming Languages (QAPL 05), Edinburgh, Scotland, April 2005. 2005.
- Gruhn, V., Laue, R., Specification Patterns for Time-Related Properties, Proc. of the 12th International Symposium on Temporal Representation and Reasoning (TIME05), 2005.
- 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.
- 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, 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.
- Koymans, R. Specifying Message Passing and Time-Critical Systems with Temporal Logic.Lecture Notes in Computer Science 651, Springer-Verlag, 1992.
- Koymans, Specifying real-time properties with metrics temporal logic, Real Time Systems, Vol.2, n.4, pp.255-299, 1990.
- Manna, Z., and A. Pnueli. The temporal logic of reactive and concurrent systems. Springer-Verlag New York, Inc., 1992.

- Mattolini, R.; Nesi, P., “An interval logic for real-time system specification”, IEEE Transactions on Software Engineering, pp.208-227, 2001
- 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.
- SAnToS Laborary, Alavi, H.; Avrunin, G.; Corbett, J.; Dillon, L.; Dwyer M.; Pasareanu, C. Specification Patterns web site http://patterns.projects.cis.ksu.edu/
- Shaw, M., Some Patterns for Software Architecture, Pattern Languages of Program Design, Vol.2, pp.255-269, 1996.