TILCO-X,
an Extension of TILCO Temporal Logic

P. Bellini, P. Nesi
Department of Systems and Informatics, University of Florence
Via S. Marta 3, 50139 Firenze, Italy, tel.: +39-055-4796523, fax.:+39-055-4796363
email: nesi@ingfi1.ing.unifi.it, www: http://www.dsi.unifi.it/~nesi

No Institute Given

Abstract. Formal techniques for the specification of real-time systems must be capable of describing temporal constraints among events and actions: properties of invariance, precedence, periodicity, repeated occurrences, liveness and safety conditions, etc. This paper describes an evolution of the Temporal Interval Logic called TILCO. TILCO is a generalisation of classical temporal logics based on the operators *eventually* and *henceforth* and allows both qualitative and quantitative specification of time relationships. TILCO is based on time intervals and can concisely express temporal constraints with time bounds. TILCO is strongly concise and efficient for the specification of real-time systems. In this paper, an extension of TILCO, called TILCO-X is presented. TILCO-X proposes two new operators that increase conciseness and readability of specifications allowing to describe (i) ordering of events without distinction between past and future, and (ii) predicates depending on the number of occurrences of events in intervals. TILCO-X is capable of describing specifications by using a lower number of quantifications and of nesting levels between temporal operators. This paper defines the semantics of TILCO-X and related examples that show the power of the model proposed.
Index terms: formal specification language, first order logic, temporal interval logic, verification and validation, real-time systems, temporal operators.

1 Introduction

For the specification of real-time system behavior many factors have to be considered. Typically, their specification includes the definition of a set of relationships expressing the temporal constraints among events and actions [1], [2]: properties of invariance, precedence among events, periodicity, liveness and safety conditions, etc. To this end, many logical languages integrating constructs for temporal reasoning – temporal logics – have been proposed; e.g., [3], [4], [5], [6], [7]. These languages are a good support as abstract approaches for requirement specification and real-time system analysis [8], [1].

Several different temporal logics with different degrees of expressiveness have been proposed. Some of them are based on propositional logic — e.g., PTL [3], TPTL [9], RTTL [10], ITL [11] — and adopt the $\diamond$ and $\Box$ operators. Others are based on first or higher order logic — e.g., TRIO [12], MTL [13], interval temporal logic [14], TILCO [7]. Propositional logic is decidable, while FOL has a greater expressive power but it is intrinsically undecidable; however, some restrictions can be applied to make the theory both decidable and executable [15]. Higher order logics have an even greater expressive power, but are more difficult to manipulate automatically than simpler logics. For these reasons, the most diffuse temporal logics are based on FOL. Consequently, most of the temporal logics can be translated into FOL. Their definition is very useful since temporal logics constrain the users to write formulæ whose validity and satisfiability can be more easily checked, leading to specifications that can be verified or automatically validated.

Logic-based languages for modelling temporal constraints can be based on *time points* (e.g., [16], [17]), or on *time intervals* (e.g., IL [5], ITL [11], [18], interval temporal logic in [14], EIL [19], RTIL [20] GIL [21]). Interval logic formulæ have typically a higher level of abstraction.

These logics usually have specific operators to express the relationships between intervals (*meet*, *before*, *after* [22]), operators for combining intervals (e.g., the *chop* operator [17]), or operators that specify the interval constituting the context of temporal formulæ [19]. Interval logics are typically more concise than point-based temporal logics for the specification of real-time systems.

The time structure can be linear or branched; however, only a linear structure can be suitably used for real system specification. A branched future can be unsuitable for specification languages, since for branched models the system can have more than one possible evolution and the correct evolution cannot be unequivocally determined in advance.

According to [7], in which TILCO (Temporal Interval Logic with Compositional Operators) temporal interval logic has been presented by the authors, none of the temporal logics presented in the past few years is completely satisfactory for real-time system specification. In fact, most of them have no metric for time, thus allowing only specification of qualitative temporal requirements — e.g., [3], [23], [5]. In the literature, only a few examples of quantitative temporal logics exist. In these cases, an operator expressing the distance between time points is usually defined. Most of the approaches including the metric for time are based on propositional logic instead of FOL, and are therefore not expressive enough to describe realistic systems - e.g., [24], EIL [19], RTIL [20], TPTL [9], [15]. Those first-order temporal logics that provide a metric for time usually allow quantification over the temporal domain — e.g., RTL [4], MTL [25], TRIO [12] — whereas a prohibition of this kind of quantification has been shown to be a necessary condition for the existence of feasible automated verification mechanisms [26]. All these temporal logics are based on time points rather than on intervals, and provide a sharp distinction between past and future. TILCO does not allow the quantification over time and present unique operators for specifying the events and action from past to future.

The authors defined TILCO temporal logic for covering the above-mentioned problems, with a special emphasis on temporal logic expressiveness and conciseness for the specification of real-time systems [7]. TILCO extends FOL with a set of temporal operators, and is a generalisation of the classical temporal logics based on the application of the operators *eventually* and *henceforth* to time intervals. TILCO has a metric for time, the time is discrete, linear and no explicit temporal quantification is allowed. Thus, TILCO allows specification of both qualitative and quantitative relationships about events and facts and provides specific compositional operators among time intervals. In TILCO, the same formalism used for system specification is employed for describing high-level properties that should be satisfied by the system itself. These must be proven on the bases of the specification in the system validation phase. Since TILCO operators quantify over intervals, instead of using time points, TILCO is more concise in expressing temporal constraints with time bounds, as is needed in specifying real-time systems. In fact, TILCO can be effectively used to express invariant, precedence among events, periodicity, liveness and safety conditions, etc., and these properties can be formally verified by automatic theorem-proving techniques. To this end, a formalisation of TILCO has been implemented in the theorem prover Isabelle/HOL [27], [28]. Using this formalisation, a set of fundamental theorems has been proven and a set of tactics has been built for supporting the semi-automatic demonstration of properties of TILCO specifications. Causal TILCO specifications are also executable by using a modified version of the Tableaux algorithm.

TILCO has been compared in [7] against several other first order temporal logics with metric of time – e.g., MTL [25], TRIO [12]. TRIO and MTL are both based on points and present a sharp distinction between past and future. MTL and TRIO have distinct operators for past and future (e.g., MTL: **G**, **H**; TRIO: **Past()** and **Futr()**). In contrast, TILCO presents a uniform model for time from past to future and unique operators for stating facts and events along the time axis [7]. These features make TILCO specifications more concise than those in TRIO and MTL.

TILCO, TRIO, MTL and many other temporal logics adopt **since** and **until** operators to specify dependencies between events. These operators make a strong distinction between past and future and, thus, their adoption frequently makes the specification complex and hard to read. This has been considered a limitation for specification conciseness.

In the specification of real time systems, the needs of specifying the occurrence of one event from the repeated occurrence of another is quite frequent. For instance, *A has to start after the arrival of 5 messages on channel B within interval I.* Specifications with these constrains are quite complex to understand and difficult to realise by using classical interval operators. To this end, some temporal logics present specific operators for this purpose – e.g., in RTL a special operator is capable of recovering at which instant the particular occurrence of an event happens [4]. These constraints can be specified in FOL and thus also in first order temporal logic but the specification results to be complex to read with respect to the complexity of the concepts described. TRIO, MTL, TILCO, do not present such temporal operators.

In this paper, TILCO-X, TILCO eXtension, is proposed. TILCO-X has been defined extending TILCO by integrating two new families of operators to cope with the problems generated by the presence of **since**, **until** operators and for making possible the specification including the counting of events. These operators called *Dynamic Intervals* and *Bounded Happen*, respectively can be combined for defining very complex real-time constraints in a concise manner. For this reason, TILCO-X has to be considered a strong improvement with respect to TILCO. TILCO-X presents a different semantics with respect to TILCO but it continues to be sound. TILCO-X has been formalised by using the theorem prover Isabelle/HOL. TILCO-X can be used to verify the completeness and consistency of specifications, as well as to validate system behaviour against its requirements and general properties.

This paper is organised as follows. In Section 2, a short overview of TILCO as presented in [7] is reported. Section 3 presents TILCO-X with some examples. In Section 4 the syntax and semantics of TILCO-X are reported. Section 6 reports a specification example to highlight the features of the extensions introduced. Conclusions are drawn in Section 7.

2 Overview of TILCO

TILCO extends FOL in order to create a logic language capable of specifying the relationships between events and time, as well as the transformations on the data domain [7]. It can be used to specify temporal constraints among events in either a qualitative or quantitative manner. Therefore, the boundaries of an interval, which specify the length of intervals and actions, can be expressed relative to other events (i.e., in a qualitative manner) or with an absolute measure (i.e., in a quantitative manner). This allows definition of expressions of ordering relationships among events or delays and time-outs. These features are mandatory for specifying the behaviour of real-time systems. In addition, the TILCO deductive approach is sound, and thus consistent. It forces the user to write formulæ without using direct quantifications over the temporal domain, thus avoiding the writing overly intricate or difficult to understand specifications [15].

TILCO includes the concepts of typed variables and constants; it provides a set of basic types and allows the definition of new types. Predefined types are: **nat** for natural numbers, **int** for integer numbers, **bool** for Booleans, **char** for text characters, and **string** for character strings. The usual arithmetic operators: $+$, $-$, $*$, $/$, mod , $\sim$ (change sign), are defined for integers and natural numbers. String manipulation functions are defined for strings. Comparative operators:

$=$, $<$, $>$, $\geq$, $\leq$, $\neq$, can be used with integers, naturals, characters and strings, and can be overloaded for user-defined types.

A system specification in TILCO is a tuple

$$\{\mathcal{U}, \mathcal{T}, \mathcal{F}, \mathcal{P}, \mathcal{V}, \mathcal{W}, \mathcal{C}, \mathcal{J}\},$$

where $\mathcal{U}$ is a set of TILCO formulæ, $\mathcal{T}$ a set of type definitions, $\mathcal{F}$ a set of functions, $\mathcal{P}$ a set of predicates, $\mathcal{V}$ a set of typed time-dependent variables, $\mathcal{W}$ a set of typed time-independent variables, $\mathcal{C}$ a set of typed constants (also called time invariant parameters), and $\mathcal{J}$ is a set of integer intervals. $\mathcal{U}$ specifies the rules defining the behavior of the specified system. $\mathcal{T}$ defines the types used in the specification. Functions and predicates have their usual meaning and are used to manipulate predefined and user-defined data-types. Time-dependent variables are employed for modelling system inputs (read-only), outputs, and auxiliary variables (read/write) of the system under specification. Time-dependent variables can assume any value in their corresponding domain. Time-independent variables are used to build parametric formulæ that operate on structured data types (i.e., arrays, lists, etc.) through quantification. Constants are used for modelling system parameters. Integer intervals, which are connected sets of integers, are used for specifying quantitative temporal relationships.

A system is specified in TILCO according to the following rules:

- a system presents: input and output ports to communicate with the external environment, and auxiliary variables for defining the internal state;
- inputs, outputs and auxiliary variables can assume only one value at each time instant. Each of them is defined by a unique name;
- an input is a typed variable whose value can change due to external events;
- an output is a typed variable which can be forced to assume a value by some predicates through an assignment. This leads to a change in the external environment;
- an auxiliary variable can be forced to a value by an assignment and it can be read as an input variable;
- a system is described to be a set of formulæ which define its behaviour and the data transformation.

2.1 TILCO Operators

TILCO's *temporal operators* have been added to FOL by leaving the evaluation time implicit. The meaning of a TILCO formula is given with respect to the current time. Time is discrete and linear, and the temporal domain is $\mathbb{Z}$, the set of integers; the minimum time interval corresponds to 1 time unit. The current time instant is represented by 0, whereas positive (negative) number represent future (past) time instants. TILCO formulæ can be time dependent or independent; the latter are those that do not present any TILCO *temporal operator*, and are comprised only of time-independent subformulæ. A time independent formula can be regarded as a constraint that must be satisfied in each time instant.

The basic temporal entity in TILCO is the interval. Intervals can be quantitatively expressed by using the notation with round, "(", ")", or squared, "[", "]", brackets for excluding and including interval boundaries, respectively. Time instants are regarded as special cases that are represented as closed intervals composed of a single point (e.g., $[a, a]$). Symbols $+\infty$ and $-\infty$ can be used as interval boundaries, if the extreme is open, to denote infinite intervals — i.e., $[a, +\infty)$ represents set $\{x \in \mathbb{Z} | a \leq x\}$. Thus, TILCO allows both the specification of *facts* in intervals and *events* in time instants. Classical operators of temporal logic (i.e., eventually, $\diamond$, and henceforth, $\Box$) can be easily obtained by using TILCO operators with infinite intervals. TILCO can be regarded as a generalisation of most of the interval logics presented in the literature in the past [7] — with the addition of a metric to measure time.

The basic TILCO *temporal operators* are:

- "**@**", bounded universal temporal quantification over an interval;
- "**?**", bounded existential temporal quantification over an interval;
- **until**, to express that either a predicate will always be true in the future, or it will be true until another predicate will become true;
- **since**, to express that either a predicate has always been true in the past, or it has been true since another predicate has become true.

Operators "**@**" and "**?**" are called temporal quantifiers. $A\,@\,i$ is true if formula A is true in every instant in the interval i, with respect to the current time instant. Therefore, if t is the current time instant, $(A\,@\,i)^{(t)} \equiv \forall x \in i.A^{(x+t)}$ holds. In particular, $A\,@\,[t_1,t_2)$ evaluated in t means:

$$\forall x \in [t_1,t_2).A^{(x+t)}.$$

Obviously t_1 and t_2 can be either positive or negative, and, thus the interval can be in the past or in the future. If the lower bound of an interval is greater than the upper bound, then the interval is null (i.e., it is equal to the empty set). Operators "**@**" and "**?**" correspond, in the temporal domain, to FOL quantifiers $\forall$ and $\exists$, respectively; hence, they are related by a duality relationship analogous to that between $\forall$ and $\exists$. "**@**" and "**?**" are used to express delays, time-outs and any other temporal constraint that requires a specific quantitative bound. Concerning the other *temporal operators*, **until** A B (evaluated in t) is true if B will always be true in the future with respect to t, or if B will be true in the interval $(t, x+t)$ with $x > 0$ and A will be true in $x+t$. This definition of **until** does not require the occurrence of A in the future, so the **until** operator corresponds to the *weak until* operator defined in PTL [23]. The operators **until** and **since** express the same concept for future and past, respectively; they are related by a relationship of *temporal duality.* **until** and **since** can be effectively used to express ordering relationships among events without the need of specifying any numeric constraint.

2.2 Some TILCO Examples

Tab. 1 provides few examples of TILCO formulæ, where t stands for a positive integer number. To provide a clearer view of TILCO's expressiveness the formulæ are accompanied by an explanation of their meaning.

$A\,@\,[t_1,t_1],(-t_2,t_3]$	A is true in t_1, and in $(-t_2,t_3]$
$A\,?\,[t_1,t_1],(t_2,t_3]$	A is true in t_1, and is true at least once in $(t_2,t_3]$
$A\,?\,[0,t_1]\,@\,[0,+\infty)$	A will become true within t_1 for each time instant in the future (response)
$(A \Rightarrow B)\,?\,[0,t]$	if A is true within t, then also B will be true at the same time
$(A \Rightarrow B\,?\,i)\,@\,j$	A leads to an assertion of B in i for each time instant of j
$(A \Rightarrow B\,@\,i)\,@\,j$	A leads to the assertion of B in the whole interval i for each time instant of j
$(A \Rightarrow B\,@\,i)\,?\,j$	A leads to the assertion of B in the whole interval i in at least a time instant of j

Table 1. Examples of TILCO formulæ.

2.3 Comments

- Each TILCO formula used in a system specification must be closed, thus each time independent variable in a formula must be quantified. If a TILCO formula is open, it is replaced by its universal closure (i.e., an external universal quantifier is introduced for each of the time independent variables which are not quantified). According to the syntax definition, each quantified variable must be time independent, otherwise (i) it would be possible to write higher order formulæ and (ii) time could not be left implicit because the meaning of the formula would change during system evolution.
- In a TILCO specification, predicates and functions with typed parameters can also be defined. Predicates are functions that return a value of type **bool**. Functions and predicates are used to define operations and relationships over predefined and user-defined types. Functions and predicates are incrementally defined by using predefined functions and predicates over the basic data types and type constructors. The body of each predicate must be specified by means of a TILCO formula, in which the only non-quantified variables are the predicate parameters. Predicates are only instruments used to simplify the writing of formulæ; hence, more complex temporal expressions and formulæ can be hidden in predicates.
- The classical *henceforth* operator, $\Box$, can be expressed in terms of TILCO operator "$@$": $A \mathbin{@} [0, +\infty)$, which means that A will be true forever from the current time instant. Analogously, the *eventually* operator, $\Diamond$, can be expressed by $A \mathbin{?} [0, +\infty)$.
- TILCO is also characterized by its compositional operators that work with intervals: *comma* ",", which corresponds to $\wedge$, and *semicolon* ";", which corresponds to $\vee$, between intervals. Compositional operators "," and ";" assume different meanings if they are associated with operators "$@$" or "$?$". Other operators among intervals, such as intersection, "$\cap$", and union, "$\cup$", could be defined by considering time intervals as sets. However, the introduction of $\cup$ is problematic because the set of intervals is not closed over this operation.
- Once system behaviour is specified by means of a set of TILCO formulæ, the specification can be validated to verify whether it corresponds to the system requirements. In TILCO, system validation is performed by proving that high-level properties (e.g., safety, liveness, etc.) are satisfied by the TILCO specification of the system. These properties can be expressed by means of other TILCO formulæ, thus TILCO is used to specify both the system and its high-level properties. Therefore, the classical safety conditions, such as $A \mathbin{@} i$ (where A is a positive property), and $\neg B \mathbin{@} i$ where B is a negative condition) must be satisfied by the system specification, where the interval i can be extended to the *specification temporal domain*, as well as to only a part of it. Moreover, liveness conditions, such as $A \mathbin{?} i$ (A will be satisfied within i) or deadlock-free conditions, such as $(\neg A \mathbin{?} i) \mathbin{@} j$, can also be specified. If during the validation of a TILCO specification it is found that a desired property (constituting a system requirement) cannot be deduced from the system specification given in terms of TILCO formulæ, then the specification is incomplete. If that property must be satisfied by the system, a new TILCO formula should be added to the system specification, provided that this formula does not contradict any other formula contained in the specification. This formula may itself be the desired property or a formula that completes the system specification in order to prove the desired property, thus allowing the incremental system specification.

3 TILCO-X, TILCO eXtension

TILCO, MTL and TRIO are first order temporal logics for real-time system specifications. They have a metric of time and thus can be profitably used for specifying qualitative and quantitative temporal constraints. Many other logics produce specifications structurally similar to TRIO and MTL or have similar operators; while, other logics can be difficult to use in comparison to TRIO, MTL and TILCO, since they are based on elementary operators that lead to the production of overly complex specifications.

For the specification of real-time systems, it is strongly relevant the conciseness, readability and understandability of the temporal logic used. This depends on the expressiveness of the logic, on the number of operators, on the structure of the formulas (nesting levels, number of calls to special functions/parameterised predicates, presence of quantifiers), the needs of user defined special operators, and the need of adopting temporal quantifications.

A formal proof of TILCO's conciseness with respect to other temporal logics would be difficult, mainly due to the lack of a formal definition of conciseness or readability. Moreover, an examination of the elementary specifications for TILCO, TRIO and MTL, has demonstrated that the typical specifications produced in TILCO use fewer distinct operators than in TRIO and MTL [7]. In addition, some specifications can be written in TRIO and/or MTL only by using nested operators, while simple direct operators are used in TILCO.

TILCO specifications are based upon four fundamental operators. A greater number of operators are present in TRIO/MTL-like formulas. Thus, complexity increases and conciseness decreases for both TRIO and MTL; and leads to a higher cognitive complexity (or comprehensibility complexity) as in programming language (demonstrated by the validation of several cognitive metrics [29], [30], [31], [32], [33]).

TRIO and MTL are both based on points and present a sharp distinction between past and future. MTL and TRIO have distinct operators for past and future (e.g., MTL: **G**, **H**; TRIO: **Past()** and **Futr()**). In contrast, TILCO is an interval temporal logic with a uniform model for time from past to future. TILCO specifications can describe facts and events without to use different operators for past and future [7].

In the analysis of TILCO and several other temporal logics performed by the authors [34], two specific fields of improvement have been identified for the specification of real time systems:

- TRIO, MTL and many other temporal logics adopt **since** and **until** operators to specify dependencies between events. Also TILCO [7] adopts **since** and **until** operators for the same purpose. These operators make a strong distinction between past and future and, thus, their adoption frequently makes the specification complex and hard to read. The adoption of a unique operator for defining ordering relationships between events reduces in several cases the needs of the adoption of nested **since** and **until** operators. This removes the gap between TILCO temporal operators for dealing with events and facts and the needs of **since** and **until** operators for specifying relationships between events.
- The needs of specifying the occurrence of one event from the repeated occurrence of another is quite frequent (operators for events counting are needed). For instance, *A has to start after the arrival of 5 messages on channel B within interval I.* Specifications with these constrains are quite complex to understand and difficult to realise by using classical temporal operators such as those proposed in TILCO, IL, TRIO, MTL, RTL, etc. This is the reason for which some temporal logics present specific operators for this purpose. In RTL, a special operator capable of recovering *at which instant the particular occurrence of an event happens* has been proposed [4]. These constraints can be specified in FOL and thus also in first order temporal logic but the specification results to be strongly complex with respect to the complexity of the concept under specification and involves several quantifications.

These facts have been considered a limitation for specification conciseness and readability and thus for the adoption of logical languages for the specification of real-time systems. Therefore, TILCO-X temporal logic has been defined by extending TILCO to enhance its readability and conciseness, especially for the expression of order relations. To this end, the operators called *Dynamic Intervals* and *Bounded Happen* have been defined. They can be combined allowing the definition of very powerful real-time constraints in a strongly concise manner.

3.1 Dynamic Intervals

Dynamic Intervals have been introduced to: (i) avoid the needs of distinguishing between past and future for ordering relationships, (ii) avoid in several cases the nesting of **since** and **until** operators, (iii) reduce the number of quantifications, (iv) allow the combination of order and quantitative relationships.

These capabilities have introduced in TILCO-X by making possible to write temporal intervals not only as constant integer sets, but also by using a formula as an interval bound.

For example the following 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 true forever in interval $[10,+\infty)$. These two conditions are represented in Fig. 1.

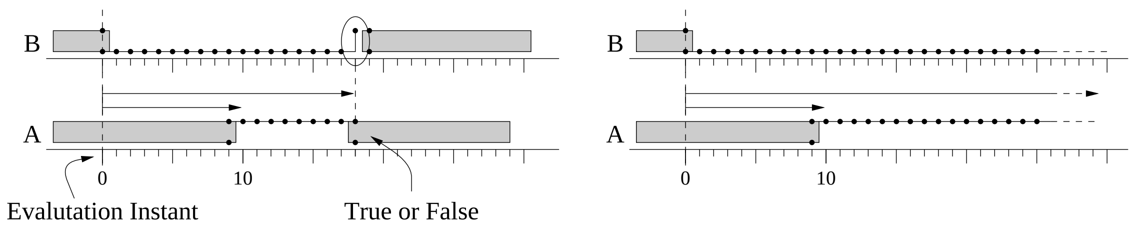

Fig. 1. Example of Dynamic Interval: $A\,@\,[10,+B)$

In a similar way, an interval bound can be located in the past; for example, formula

$$A\,@\,(-B,0]$$

states that A *is true since the last time instant in which* B *is true until the current instant.* Where $-B$ identifies the last instant where B is true.

It should be noted that, **until** and **since** operators can be defined by means of the following formulas:

$$\textbf{until}\ A\ B \equiv B\,@\,(0,+A)$$
$$\textbf{since}\ A\ B \equiv B\,@\,(-A,0).$$

In the following, the above mentioned applications of the Dynamic Interval solution are presented and discussed.

Distinction between past and future

The following specification is a typical case in which a strong distinction between past and future has to be performed for adopting of **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 follow, in MTL and TRIO structurally similar formulas can be obtained:

$$(\textbf{since}\ C\ (A \Rightarrow B)) \wedge (A \Rightarrow B) \wedge (\textbf{until}\, D\ (A \Rightarrow B))$$

With TILCO-X, it is possible to write intervals starting from the past and ending in the future; thus, the above specification results to be strongly 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. Another example in TILCO-X is the following formula specifying that *A or B happened in the last ten instants or will happen until C and D are true*:

$$(A \vee B)\,\textbf{?}\,[-10, +(C \wedge D))]$$

This example shows, how it is possible to write specifications in which the interval has a fixed lower bound in the past and a dynamic upper bound in the future. This last example can be written in TILCO in the following way:

$$(A \vee B)\,\textbf{?}\,[-10, 0] \vee \neg\, \textbf{until}(C \wedge D)(\neg(A \vee B))$$

Nesting levels

The definition of intervals with dynamic bounds (identified by the validity of a generic formula) avoids in many cases the adoption of nesting temporal quantifiers. Thus, TILCO-X produces more concise formulas that result to be much more readable.

An example could be the following TILCO-X formula

$$A\,@\,[+B, +\infty)$$

stating that *after the next occurrence of B, A is always true*. The same behaviour can be specified in TILCO by using nested **until** and @ operators such as in:

$$\boxed{\textbf{until}\,(B \wedge \boxed{A\,@\,[0, +\infty)})\ (\neg B)}$$

The box around formulas highlights the nesting levels of the temporal operators.

A more complex example can be the following TILCO-X formula

$$A\,\textbf{?}\,[+B, +C]$$

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 has to be written by using two nested **until** operators.

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

......TBR........ B must happen in the future because otherwise the interval $[+B,+C]$ is empty and A cannot happen on an empty interval.

Reduction of the number of quantifications Temporal quantifications are not allowed by TILCO language since their prohibition has been shown to be a necessary condition for the existence of feasible automated verification mechanisms [26], [7]. Therefore, in TILCO is very complex to specify certain constraints without the adoption of a direct temporal quantification. For example:

After every occurrence of event S, a signal A has to be true after the first occurrence of event E (if it happens) and A remains true until a certain deadline d (value relative to event S)

Fig. 2 reports a representation of the constraint proposed which is complex to be specified without temporal quantification.

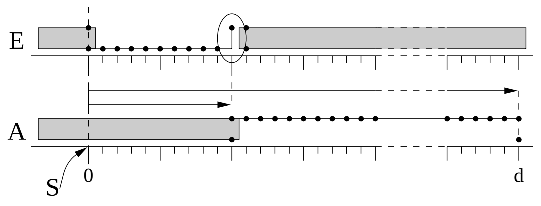

Fig. 2.

The specification complexity is due to the fact that, the above constraint is comprised of two parts: one relating S to next occurrence of E; and the second, relating S to the end of the temporal interval d (that is constant). In order, to specify the second part an @ operator could be used. The identified time instant in which E will occurs has to be considered as the starting bounds for its interval. This dependency, between the two parts, can be only defined by using a quantification.

A way to specify this requirement in TILCO is to introduce a clock variable "Ck" with the following property:

$$((\mathrm{Ck}=0\wedge\neg S)\,@\,(-\infty,0]\wedge(\exists\,v.\ \mathrm{Ck}=v\ \Rightarrow\ (\mathrm{Ck}=v+1)\,@\,[1,1])\,@\,(0,+\infty))?(-\infty,+\infty)$$

stating that an instant (the initial) exists before Ck is zero and S does not happen, and that Ck is incremented by one at every time instant, after the initial time instant.

Using Ck, the above reported constraint can be written as follows in TILCO:

$$(1)\qquad S\wedge E\,\mathbf{?}\,(0,d)\Rightarrow(\exists v.\ \mathrm{Ck}=v\wedge\mathbf{until}\,(E\wedge\mathbf{until}\,(\mathrm{Ck}\geq v+d)\ A)\ (\neg E)$$

A little bit more complex formulas, but structurally similar, can be obtained by using MTL and TRIO. This writing modality for constraint specification should be avoided since it produces less readable specifications. This factor is much more relevant for the specification of complex systems.

In temporal logics supporting time quantification, such as TRIO, the above reported constraint can be rewritten as follows:

$$\forall t\, t'.\ S \wedge \mathbf{Futr}(E,t) \wedge (0 < t < d) \wedge (t < t' < d) \longrightarrow \mathbf{Futr}(A,t')$$

A structurally similar formula can be obtained in MTL.

In TILCO-X, the above constraint is simply stated by the following formula:

$$S \Rightarrow A\,@\,(+E,d)$$

that is much more simple than both TRIO and TILCO versions.

Therefore, it has been shown that the adoption of new TILCO-X operators reduces the number of quantifications and operators, thus, increasing the readability and coinciseness of formulas.

Combination of order and quantitative relationships Using TILCO-X is easy to write ordering relations between events, especially those that combine order and quantitative relationships. For example, the following TILCO-X formula states that *A happens after B within 100 time units*:

$$A\,\mathbf{?}\,(+B,100]$$

In Fig. 3, the visual representation of this condition is reported.

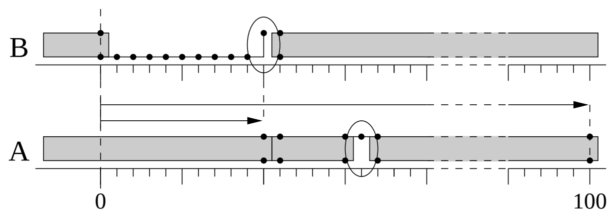

Fig. 3. Example of Dynamic Interval: $A\,\mathbf{?}\,(+B,100]$

The above formula is in some measure *similar* to TILCO formula

$$A\,\mathbf{?}\,(0,100] \wedge \neg\,\mathbf{until}\,A\,(\neg B)$$

These two formulas are not equivalent (the second implies the first); because according to TILCO-X formula, A may be true or not before B is true. This is not possible for the TILCO formula, as it has been depicted in Fig. 4.

In order to have the equivalence, A has to be false until B is true, thus the correct TILCO formula is:

$$A\,\mathbf{?}\,(0,100] \wedge \neg\,\mathbf{until}\,A\,(\neg B) \Longleftrightarrow (\neg A)\,@\,(0,+B] \wedge A\,\mathbf{?}\,(+B,100]$$

This is a further example of the conciseness of TILCO-X with respect to TILCO. As demonstrated in [7], quite structurally similar formulas can be written for TRIO and MTL, but they result even less concise than TILCO formulas.

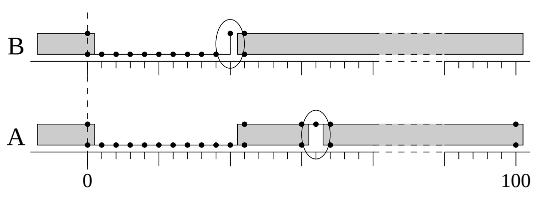

Fig. 4. A model for formula $A\,\mathbf{?}\,(0,100] \wedge \neg\,\mathbf{until}\,A\;(\neg B)$

3.2 Bounded Happen

Bounded Happen has been defined to increase the readability of constraints which includes the dependence from the counting of occurrences. Sometimes constraint implies the counting the number of occurrences of an event or in general the number of times that a formula is true in a given time interval. In TILCO, as well as in TRIO, MTL and other temporal logics, such a requirement can be specified by using a variable to count the number of times that a formula is true from an instant to another. This can be performed by using formula $(C_{\rightarrow}(A, n_A))$ stating that n_A is a variable that counts the occurrences of A from the evaluation time instant.

For example, to state that an event E occurs at most five times in $[2, 10)$, the following formula can be used:

$$C_{\rightarrow}(E, n_E)\;@\,[2,2] \wedge (n_E \leq 5)\;@\,[2,10)$$

where:

$$\begin{aligned} C_{\rightarrow}(A, n_A) \overset{\text{def}}{=} &(\neg A \rightarrow n_A = 0) \wedge \\ &(A \rightarrow n_A = 1) \wedge \\ &((\forall k.\; A \wedge (n_A = k)\;@\,[-1,-1] \rightarrow n_A = k+1) \wedge \\ &(\forall k.\; \neg A \wedge (n_A = k)\;@\,[-1,-1] \rightarrow n_A = k))\;@\,(0,+\infty) \end{aligned}$$

While, when E occurs at least three times in interval $[2, 10)$, the formula is:

$$C_{\rightarrow}(E, n_E)\;@\,[2,2] \wedge (n_E \geq 3)\,\mathbf{?}\,[2,10)$$

Therefore, a formula stating that *the above formula is true in every time instant* has to manage the variability of distinct counters that should be activated in each time instant:

$$(C_{\rightarrow}(E, n_E)\;@\,[2,2] \wedge (n_E \geq 3)\,\mathbf{?}\,[2,10))\;@\,[0,+\infty)$$

A different solution can be based on the adoption of a unique counter for the whole constraint and an existential quantification.

$$C_{\rightarrow}(E, n_E) \wedge (\exists k.(n_E = k)\;@\,[2,2] \wedge (n_E \geq k+3)\,\mathbf{?}\,[2,10))\;@\,[0,+\infty)$$

Bounded Happen operator has been introduced to specify the family of the above presented constraints in a concise manner. It 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:

$$A\,\mathbf{?}_2\,[1,15)$$

states that A is true two 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 $[1, 15)$.

With the combination of these 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 TILCO-X example:

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

Bounded happen can be used with *Dynamic Interval operator*. The following formula states that A happens two or three times until B happens:

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

The bounded happen may be used to state that a formula *becomes* true a limited number of times in an interval, this can be achieved with the derived operator up ($\uparrow$) defined as

$$\uparrow A \overset{\text{def}}{=} A \wedge (\neg A) \,@\, [-1, -1]$$

Therefore, formula

$$(\uparrow A) \,?^2_2\, [1, 15)$$

states that A becomes true exactly two times in $[1, 15)$. A possible model for this formula is reported in Fig. 5.

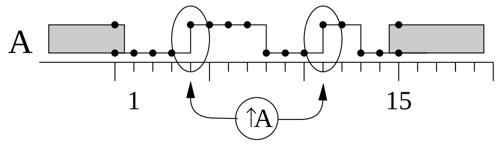

Fig. 5. Example of Bounded Happen: $(\uparrow A) \,?^2_2\, [1, 15)$

Additional interesting examples are:

$$(A \wedge B \,?^2_2\, (-100, 0)) \Rightarrow \neg A \,@\, (0, 10)$$

which states that *if A is true and in the last 100 time units there were two occurrences of B, then A will be false for 10 time units*; and

$$A \,?^1_3\, [0, +(B \wedge C)]$$

stating that A *will happen from one to three times until B and C are true.*

4 TILCO-X Syntax & Semantics

Given $\mathcal{F}$, $\mathcal{P}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{C}$ as defined for old TILCO, the syntax of TILCO-X formulæ is defined by the following BNF-like definitions:

$$\begin{array}{rcl}
\text{interval} & ::= & \text{open limit , limit close} \\
 & | & [\,\text{limit}\,] \\
 & | & \text{open limit} , +\infty) \\
 & | & (-\infty\,, \text{ limit close} \\
 & | & (-\infty\,, +\infty) \\
\text{limit} & ::= & i \quad \textit{for each } i \in \mathbb{Z} \\
 & | & +\text{formula} \mid -\text{ formula} \\
\text{open} & ::= & (\mid [\\
\text{close} & ::= &] \mid) \\
\text{interval\_list} & ::= & \text{interval} \\
 & | & \text{interval interval\_op interval} \\
\text{interval\_op} & ::= & , \mid ; \\
\text{variable} & ::= & w \quad \textit{for each } w \in \mathcal{W} \\
\text{term} & ::= & v \quad \textit{for each } v \in \mathcal{V} \\
 & | & \text{variable} \\
 & | & c \quad \textit{for each } c \in \mathcal{C} \\
 & | & f(\text{term\_list}) \quad \textit{for each} f \in \mathcal{F} \\
\text{term\_list} & ::= & \text{term} \\
 & | & \text{term, term\_list} \\
\text{atomic\_formula} & ::= & p(\text{term\_list}) \quad \textit{for each } p \in \mathcal{P} \\
\text{formula} & ::= & \top | \bot |\text{atomic\_formula} \\
 & | & \neg\text{formula} \\
 & | & \text{formula op formula} \\
 & | & v := \text{term} \quad \textit{for each } v \in \mathcal{V} \\
 & | & \text{quantifier variable. formula} \\
 & | & \text{formula temporal\_quantifier interval\_list} \\
 & | & (\text{formula}) \\
\text{op} & ::= & \vee | \wedge \mid \Rightarrow \mid \Leftrightarrow | {=}\!\triangleright\!\triangleright | {=}\!\triangleleft\!\triangleleft \\
\text{quantifier} & ::= & \forall | \exists | \exists ! \\
\text{temporal\_quantifier} & ::= & @ \mid ? \mid ?_m \mid ?^M \mid ?_m^M \quad \textit{for each } m, M \in \mathbb{N}
\end{array}$$

Before defining the semantics of TILCO-X, it is important to introduce the concept of *interpretation* of a TILCO-X formula. This concept is also used to define the validity and the satisfiability of TILCO-X formulæ and has been derived from the corresponding concept of TILCO [7].

Given a syntactically correct TILCO-X formula A, with $\{t_1, \ldots, t_h\}$ set of types used in A, $\{p_1, \ldots, p_k\}$ predicates, $\{f_1, \ldots, f_l\}$ functions, $\{v_1, \ldots, v_m\}$ time-dependent variables, $\{c_1, \ldots, c_q\}$ constants then an *interpretation* $\mathcal{I}$ is a tuple

$$(\{D_1, \ldots, D_h\}, \{R_1, \ldots, R_k\}, \{F_1, \ldots, F_l\}, \{V_1(t), \ldots, V_m(t)\}, \{C_1, \ldots, C_q\})$$

where:

- $\{D_1, \ldots, D_h\}$ assigns a domain D_i to each type t_i;
- $\{R_1, \ldots, R_k\}$ assigns an n-ary relation R_i over $D_{i_1} \times \ldots \times D_{i_n}$ to each n-ary predicate p_i with arguments of type $t_{i_1}, \ldots, t_{i_n}$;
- $\{F_1, \ldots, F_l\}$ assigns an n-ary function F_i over $D_{i_1} \times \ldots \times D_{i_n}$ to each n-ary function f_i with arguments of type $t_{i_1}, \ldots, t_{i_n}$;
- $\{V_1(t), \ldots, V_m(t)\}$ assigns a function of time $V_i(t) : \mathbb{Z} \rightarrow D_n$ to each time-dependent variable v_i of type t_n, specifying the history of that variable in every time instant (where t is the absolute time);
- $\{C_1, \ldots, C_q\}$ assigns a value $C_i \in D_n$ to each constant c_i of type t_n;

Given a TILCO-X formula A and an interpretation $\mathcal{I}$ for A, notation

$$\mathcal{I}, t \models A$$

expresses that $\mathcal{I}$ is a *model* for A evaluated in the time instant t.

To properly define the TILCO-X temporal operators (**@** and **?**) a function, to interpret an interval I, is needed:

$$[\![I]\!]_{\mathcal{I},t} \subseteq \mathbb{Z}$$

This represents the set of time instants corresponding to instant t where a formula defined over I has to be evaluated. The definition of this function, which makes TILCO-X strongly different with respect to TILCO, is reported later.

Moreover to formally define the *Bounded Happen* operator, a function $(\mathrm{N}_{\mathcal{I},t}(I, A))$ to count the number of time instants where formula A is true in an interval I is needed. Its definition is

$$\mathrm{N}_{\mathcal{I},t}(I, A) \overset{\text{def}}{=} |\{i \in I \mid \mathcal{I}, i + t \models A\}|$$

where $|\cdot|$ gives the number of elements in a set if the set is finite, or $+\infty$ if it is infinite.

The evaluation of $\mathcal{I}, t \models A$, stating the semantics of TILCO-X, is inductively defined on the structure of A by the following rules:

- $\mathcal{I}, t \models \top$;
- $\mathcal{I}, t \not\models \bot$;
- $\mathcal{I}, t \models \neg A$ iff $\mathcal{I}, t \not\models A$;
- $\mathcal{I}, t \models A_1 \wedge A_2$ iff $\mathcal{I}, t \models A_1$ and $\mathcal{I}, t \models A_2$;
- $\mathcal{I}, t \models A_1 \vee A_2$ iff either $\mathcal{I}, t \models A_1$ or $\mathcal{I}, t \models A_2$;
- $\mathcal{I}, t \models x := exp$ iff there exists a constant $k \in D_x$ such that $\mathcal{I}, t \models x = k$ and $\mathcal{I}, t - 1 \models exp = k$, where D_x is the domain assigned to the type of x by $\mathcal{I}$;
- $\mathcal{I}, t \models \forall x.A(x)$ iff, for each $y \in D_x$ it is true that $\mathcal{I}, t \models A(y)$, where D_x is the domain assigned to the type of x by $\mathcal{I}$;
- $\mathcal{I}, t \models \exists x.A(x)$ iff, there exists a $y \in D_x$ such that $\mathcal{I}, t \models A(y)$, where D_x is the domain assigned to the type of x by $\mathcal{I}$;

- $\mathcal{I}, t \models \exists! x.A(x)$ iff, there exists one and only one $y \in D_x$ such that $\mathcal{I}, t \models A(y)$, where D_x is the domain assigned to the type of x by $\mathcal{I}$;
- $\mathcal{I}, t \models A \,@\, I$ iff, for each $s \in [\![I]\!]_{\mathcal{I},t}$, $\mathcal{I}, s+t \models A$ is true;
- $\mathcal{I}, t \models A \,?\, I$ iff, there exists an $s \in [\![I]\!]_{\mathcal{I},t}$ such that $\mathcal{I}, s+t \models A$;
- $\mathcal{I}, t \models A \,?_m I$ iff, $m \le \mathrm{N}_{\mathcal{I},t}([\![I]\!]_{\mathcal{I},t}, A)$;
- $\mathcal{I}, t \models A \,?^M I$ iff, $\mathrm{N}_{\mathcal{I},t}([\![I]\!]_{\mathcal{I},t}, A) \le M$;
- $\mathcal{I}, t \models A \,?_m^M I$ iff, $\mathcal{I}, t \models (A \,?_m I) \wedge (A \,?^M I)$;
- $\mathcal{I}, t \models p_i(e_1, \ldots, e_n)$, iff $(E_1, \ldots, E_n) \in R_i$, where R_i is the relation assigned by $\mathcal{I}$ to p_i and E_j, for each $j = 1, \ldots, n$, are the results of the expressions e_j when the values assigned by $\mathcal{I}$ are substituted for the constants and variables, and the variables are evaluated in t.

The semantics of predicates also includes that of functions, variables and constants.

In TILCO-X, the definition of $[\![I]\!]_{\mathcal{I},t}$ depends on two functions: $l^+_{\mathcal{I}}(A,t)$ and $l^-_{\mathcal{I}}(A,t)$. They are used to locate the next/previous time instant, corresponding to time instant t, where a formula A is true. These functions return $+\infty/-\infty$, if such an instant does not exist. Their formal definition is reported in the following:

$$l^+_{\mathcal{I}}(A,t) = \begin{cases} x & \text{if } 0 < x \text{ and } \mathcal{I}, x+t \models A \text{ and } \mathcal{I}, t \models (\neg A) \,@\, (0, x) \\ +\infty & \text{if } \mathcal{I}, t \models (\neg A) \,@\, (0, +\infty) \end{cases}$$

$$l^-_{\mathcal{I}}(A,t) = \begin{cases} -x & \text{if } 0 < x \text{ and } \mathcal{I}, -x+t \models A \text{ and } \mathcal{I}, t \models (\neg A) \,@\, (-x, 0) \\ -\infty & \text{if } \mathcal{I}, t \models (\neg A) \,@\, (-\infty, 0) \end{cases}$$

All the possible typologies of intervals that can be written in TILCO-X are reported in Figure 6, with their corresponding semantics. For example, to interval $[+A, b]$ is associated the set of integer values lower or equal to b and greater or equal to $l^+_{\mathcal{I}}(A,t)$ that represent the next time where formula A is true. If b is negative the set is empty.

Other TILCO-X operators are treated with the following definitions:

$$\begin{array}{rcl@{\qquad}rcl}
A_1 \Rightarrow A_2 & \overset{\text{def}}{=} & \neg A_1 \vee A_2 & A \,?\, I; J & \overset{\text{def}}{=} & (A \,?\, I) \vee (A \,?\, J) \\
A_1 \Leftrightarrow A_2 & \overset{\text{def}}{=} & A_1 \Rightarrow A_2 \wedge A_2 \Rightarrow A_1 & A \,?_m I, J & \overset{\text{def}}{=} & (A \,?_m I) \wedge (A \,?_m J) \\
A_1 {=}\!\rhd\!\!\rhd A_2 & \overset{\text{def}}{=} & A_1 \Rightarrow A_2 \,@\, [1,1] & A \,?_m I; J & \overset{\text{def}}{=} & (A \,?_m I) \vee (A \,?_m J) \\
A_1 {=}\!\lhd\!\!\lhd A_2 & \overset{\text{def}}{=} & A_1 \Rightarrow A_2 \,@\, [-1,-1] & A \,?^M I, J & \overset{\text{def}}{=} & (A \,?^M I) \wedge (A \,?^M J) \\
A \,@\, I, J & \overset{\text{def}}{=} & (A \,@\, I) \wedge (A \,@\, J) & A \,?^M I; J & \overset{\text{def}}{=} & (A \,?^M I) \vee (A \,?^M J) \\
A \,?\, I, J & \overset{\text{def}}{=} & (A \,?\, I) \wedge (A \,?\, J) & A \,?_m^M I, J & \overset{\text{def}}{=} & (A \,?_m^M I) \wedge (A \,?_m^M J) \\
A \,@\, I; J & \overset{\text{def}}{=} & (A \,@\, I) \vee (A \,@\, J) & A \,?_m^M I; J & \overset{\text{def}}{=} & (A \,?_m^M I) \vee (A \,?_m^M J)
\end{array}$$

Where the TILCO interval composition operators "," and ";" are extended to bounded happen in a way similar to regular happen.

In the case where the interval is null, it holds:

$$\begin{aligned} A \,@\, \emptyset &\equiv \top; \\ A \,?\, \emptyset &\equiv \bot. \end{aligned}$$

5 Deductive system

In this section, the deductive system used to prove properties in TILCO-X is introduced.

The FOL's deductive system in natural deduction style has been enhanced by adding rules for introduction and elimination of TILCO/TILCO-X temporal operators.

$$\llbracket\, [a, b]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a \le x \le b\} \tag{1}$$
$$\llbracket\, [+A, b]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) \le x \le b\} \tag{2}$$
$$\llbracket\, [-A, b]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) \le x \le b\} \tag{3}$$
$$\llbracket\, [a, +B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a \le x \le l^{+}_{\mathcal{I}}(B,t)\} \tag{4}$$
$$\llbracket\, [a, -B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a \le x \le l^{-}_{\mathcal{I}}(B,t)\} \tag{5}$$
$$\llbracket\, [+A, +B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) \le x \le l^{+}_{\mathcal{I}}(B,t)\} \tag{6}$$
$$\llbracket\, [-A, +B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) \le x \le l^{+}_{\mathcal{I}}(B,t)\} \tag{7}$$
$$\llbracket\, [-A, -B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) \le x \le l^{-}_{\mathcal{I}}(B,t)\} \tag{8}$$
$$\llbracket\, [+A, -B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) \le x \le l^{-}_{\mathcal{I}}(B,t)\} \tag{9}$$
$$\llbracket\, (a, b]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a < x \le b\} \tag{10}$$
$$\llbracket\, (+A, b]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) < x \le b\} \tag{11}$$
$$\llbracket\, (-A, b]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) < x \le b\} \tag{12}$$
$$\llbracket\, (a, +B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a < x \le l^{+}_{\mathcal{I}}(B,t)\} \tag{13}$$
$$\llbracket\, (a, -B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a < x \le l^{-}_{\mathcal{I}}(B,t)\} \tag{14}$$
$$\llbracket\, (+A, +B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) < x \le l^{+}_{\mathcal{I}}(B,t)\} \tag{15}$$
$$\llbracket\, (-A, +B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) < x \le l^{+}_{\mathcal{I}}(B,t)\} \tag{16}$$
$$\llbracket\, (-A, -B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) < x \le l^{-}_{\mathcal{I}}(B,t)\} \tag{17}$$
$$\llbracket\, (+A, -B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) < x \le l^{-}_{\mathcal{I}}(B,t)\} \tag{18}$$
$$\llbracket\, [a, b)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a \le x < b\} \tag{19}$$
$$\llbracket\, [+A, b)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) \le x < b\} \tag{20}$$
$$\llbracket\, [-A, b)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) \le x < b\} \tag{21}$$
$$\llbracket\, [a, +B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a \le x < l^{+}_{\mathcal{I}}(B,t)\} \tag{22}$$
$$\llbracket\, [a, -B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a \le x < l^{-}_{\mathcal{I}}(B,t)\} \tag{23}$$
$$\llbracket\, [+A, +B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) \le x < l^{+}_{\mathcal{I}}(B,t)\} \tag{24}$$
$$\llbracket\, [-A, +B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) \le x < l^{+}_{\mathcal{I}}(B,t)\} \tag{25}$$
$$\llbracket\, [-A, -B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) \le x < l^{-}_{\mathcal{I}}(B,t)\} \tag{26}$$
$$\llbracket\, [+A, -B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) \le x < l^{-}_{\mathcal{I}}(B,t)\} \tag{27}$$
$$\llbracket\, (a, b)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a < x < b\} \tag{28}$$
$$\llbracket\, (+A, b)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) < x < b\} \tag{29}$$
$$\llbracket\, (-A, b)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) < x < b\} \tag{30}$$
$$\llbracket\, (a, +B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a < x < l^{+}_{\mathcal{I}}(B,t)\} \tag{31}$$
$$\llbracket\, (a, -B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a < x < l^{-}_{\mathcal{I}}(B,t)\} \tag{32}$$
$$\llbracket\, (+A, +B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) < x < l^{+}_{\mathcal{I}}(B,t)\} \tag{33}$$
$$\llbracket\, (-A, +B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) < x < l^{+}_{\mathcal{I}}(B,t)\} \tag{34}$$
$$\llbracket\, (-A, -B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) < x < l^{-}_{\mathcal{I}}(B,t)\} \tag{35}$$
$$\llbracket\, (+A, -B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) < x < l^{-}_{\mathcal{I}}(B,t)\} \tag{36}$$
$$\llbracket\, [a, +\infty)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a \le x\} \tag{37}$$
$$\llbracket\, [+A, +\infty)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) \le x\} \tag{38}$$
$$\llbracket\, [-A, +\infty)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) \le x\} \tag{39}$$
$$\llbracket\, (a, +\infty)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid a < x\} \tag{40}$$
$$\llbracket\, (+A, +\infty)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{+}_{\mathcal{I}}(A,t) < x\} \tag{41}$$
$$\llbracket\, (-A, +\infty)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid l^{-}_{\mathcal{I}}(A,t) < x\} \tag{42}$$
$$\llbracket\, (-\infty, b]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid x \le b\} \tag{43}$$
$$\llbracket\, (-\infty, +B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid x \le l^{+}_{\mathcal{I}}(B,t)\} \tag{44}$$
$$\llbracket\, (-\infty, -B]\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid x \le l^{-}_{\mathcal{I}}(B,t)\} \tag{45}$$
$$\llbracket\, (-\infty, b)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid x < b\} \tag{46}$$
$$\llbracket\, (-\infty, +B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid x < l^{+}_{\mathcal{I}}(B,t)\} \tag{47}$$
$$\llbracket\, (-\infty, -B)\, \rrbracket_{\mathcal{I},t} = \{x \in \mathbb{Z} \mid x < l^{-}_{\mathcal{I}}(B,t)\} \tag{48}$$
$$\llbracket\, (-\infty, +\infty)\, \rrbracket_{\mathcal{I},t} = \mathbb{Z} \tag{49}$$

Fig. 6. Definition of $\llbracket \cdot \rrbracket_{\mathcal{I},t}$

The deduction rules for basic logical operators are the following:

$$\top I \quad \frac{}{\vdash \top} \qquad\qquad \bot E \quad \frac{\vdash \bot}{\vdash P}$$

$$\wedge I \quad \frac{\vdash P \quad \vdash Q}{\vdash P \wedge Q} \qquad\qquad \wedge E1 \quad \frac{\vdash P \wedge Q}{\vdash P} \qquad\qquad \wedge E2 \quad \frac{\vdash P \wedge Q}{\vdash Q}$$

$$\vee E \quad \frac{\vdash P \vee Q \quad P \vdash R \quad Q \vdash R}{\vdash R} \qquad\qquad \vee I1 \quad \frac{\vdash P}{\vdash P \vee Q} \qquad\qquad \vee I2 \quad \frac{\vdash Q}{\vdash P \vee Q}$$

$$\Rightarrow I \quad \frac{P \vdash Q}{\vdash P \Rightarrow Q} \qquad\qquad \Rightarrow E(\mathrm{MP}) \quad \frac{\vdash P \Rightarrow Q \quad \vdash P}{\vdash Q}$$

$$\forall I \quad \frac{\vdash P(x)}{\vdash \forall x.P(x)} \quad x \text{ free only in } P \qquad\qquad \forall E \quad \frac{\vdash \forall x.P(x)}{\vdash P[t/x]}$$

$$\exists I \quad \frac{\vdash P[t/x]}{\vdash \exists x.P(x)} \qquad\qquad \exists E \quad \frac{\vdash \exists x.P(x) \quad P(x) \vdash Q}{\vdash Q} \quad x \text{ free only in } P$$

The introduction and elimination rules for **@** and **?** are:

$$\textbf{@}\, I \quad \frac{\frac{\vdash_t x\ \mathbf{in}\ I}{\vdash_{x+t} P}}{\vdash_t P\ \textbf{@}\, I} \quad x \text{ not free in any assump.} \qquad\qquad \textbf{@}\, E \quad \frac{\vdash_t P\ \textbf{@}\, I \quad \vdash_t x\ \mathbf{in}\ I}{\vdash_{x+t} P}$$

$$\textbf{?}\, I \quad \frac{\vdash_{x+t} P \quad \vdash_t x\ \mathbf{in}\ I}{\vdash_t P\ \textbf{?}\, I} \qquad\qquad \textbf{?}\, E \quad \frac{\vdash_t P\ \textbf{?}\, I \quad \frac{\vdash_{x+t} P \quad \vdash_t x\ \mathbf{in}\ I}{\vdash R}}{\vdash R} \quad x \text{ not free in any assump.}$$

These rules are similar to the ones provided for TILCO except that operator **in** replaces the standard $\in$ set operator. The **in** operator establishes if an integer value is in a Dynamic Interval and its evaluation depends on the evaluation time t.

The **in** operator applied to a Dynamic Interval can be defined in the following form:

$$x \,\mathbf{in}\, I = \begin{cases} x \succeq b1 \wedge x \preceq b2 & \text{if } I = [b1, b2] \\ (x-1) \succeq b1 \wedge (x+1) \preceq b2 & \text{if } I = (b1, b2) \\ x \succeq b1 \wedge (x+1) \preceq b2 & \text{if } I = [b1, b2) \\ (x-1) \succeq b1 \wedge x \preceq b2 & \text{if } I = (b1, b2] \\ x \succeq b \wedge x \preceq b & \text{if } I = [b] \end{cases}$$

This definition uses two new operators, $\succeq$ and $\preceq$, to check if an integer value is after or before an interval bound, where a bound can be an integer value, plus or minus infinity or a dynamic bound as $+A$ or $-A$. Since a dynamic bound depends on the evaluation instant also $\succeq$ and $\preceq$ operators depend on the evalutation instant.

These operators ($\succeq$, $\preceq$ and **in**) have been introduced to avoid the specification of two rules (introduction and elimination) for each of the 49 possible combinations of intervals, and to permit to prove generic properties about intervals.
For example:

$$12 \succeq +B$$

is true in the evalutation time instant if there exists an instant in the future where B is true, B is false upto that instant, and this instant is distant less or equal to 12 time units from the evaluation time instant.
Formula

$$8 \preceq +B$$

is true in the evaluation time instant in two cases: if B will not be true in the future, and if the first time when B will be true is after 8 time units from the current time.
These examples are reported in Figure 7

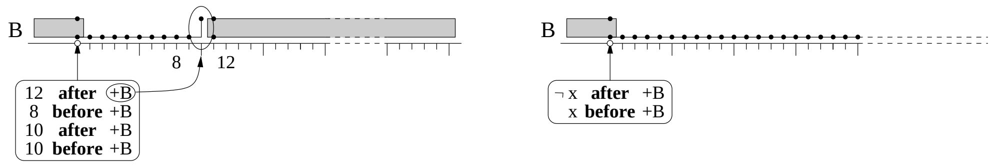

Fig. 7. Examples of $\succeq$ (after) and $\preceq$ (before)

$\succeq$ and $\preceq$ operators are defined in the following way:

$$x \succeq b = \begin{cases} b \leq x & \text{if } b \text{ is an integer} \\ \bot & \text{if } b = +\infty \\ \top & \text{if } b = -\infty \\ \exists\, t.0 < t \wedge t \leq x \wedge N(t) \wedge \\ \quad \forall\, t'.0 < t' < t \rightarrow \neg N(t') & \text{if } b = +N \\ (\exists\, t.t < 0 \wedge t \leq x \wedge P(t) \wedge \\ \quad \forall\, t'.t < t' < 0 \rightarrow \neg P(t')) \vee & \text{if } b = -P \\ \forall\, t'.t' < 0 \rightarrow \neg P(t') \end{cases}$$

$$x \preceq b = \begin{cases} x \leq b & \text{if } b \text{ is an integer} \\ \top & \text{if } b = +\infty \\ \bot & \text{if } b = -\infty \\ (\exists\, t.0 < t \wedge x \leq t \wedge N(t) \wedge \\ \quad \forall\, t'.0 < t' < t \rightarrow \neg N(t')) \vee & \text{if } b = +N \\ \forall\, t'.0 < t' \rightarrow \neg N(t') & \\ \exists\, t.t < 0 \wedge x \leq t \wedge P(t) \wedge & \\ \quad \forall\, t'.t < t' < 0 \rightarrow \neg P(t') & \text{if } b = -P \end{cases}$$

where $P(t)$ (or $N(t)$) is true if expression P (or N) is true t time instants from the evaluation instant (in the future or in the past, it depends on the sign of t).

It should be noted that $\succeq$ and $\preceq$ operators are not strict, since the bound value is also considered to satisfy the relation (so they are an extension of $\geq$ and $\leq$).

The introduction and elimination rules for $\succeq$ operator are:

$$\succeq vI \quad \frac{\vdash v \leq x}{\vdash_t x \succeq v} \qquad \succeq vE \quad \frac{\vdash_t x \succeq v}{\vdash v \leq x}$$

$$\succeq -\infty I \quad \frac{}{\vdash_t x \succeq -\infty} \qquad \succeq +\infty E \quad \frac{\vdash_t x \succeq +\infty}{\vdash \bot}$$

$$\succeq nextI \quad \frac{\vdash_{t+x'} N \quad \vdash_t \neg N \,@\, (0, x') \quad \vdash 0 < x' \quad \vdash x' \leq x}{\vdash_t x \succeq +N}$$

$$\succeq nextE \quad \frac{\vdash_t x \succeq +N \quad \dfrac{\vdash_{t+x'} N \quad \vdash_t \neg N \,@\, (0,x') \quad \vdash 0 < x' \quad \vdash x' \leq x}{\vdash R}}{\vdash R}$$

$$\succeq prevI1 \quad \frac{\vdash_{t+x'} P \quad \vdash_t \neg P \,@\, (x', 0) \quad \vdash x' < 0 \quad \vdash x' \leq x}{\vdash_t x \succeq -P}$$

$$\succeq prevI2 \quad \frac{\vdash_t \neg P \,@\, (-\infty, 0)}{\vdash_t x \succeq -P}$$

$$\succeq prevE \quad \frac{\vdash_t x \succeq -P \quad \dfrac{\vdash_{t+x'} P \quad \vdash_t \neg P \,@\, (x',0) \quad \vdash x' < 0 \quad \vdash x' \leq x}{\vdash R}}{\vdash R}$$

Introduction and/or elimination rules are reported for each kind of bound: integer value, plus and minus infinity, next ($+N$) and prev ($-P$).

Rules for $\preceq$ operator are similar to the previous:

$$\preceq vI \quad \frac{\vdash x \leq v}{\vdash_t x \preceq v} \qquad \preceq vE \quad \frac{\vdash_t x \preceq v}{\vdash x \leq v}$$

$$\preceq +\infty I \quad \frac{}{\vdash_t x \preceq +\infty} \qquad \preceq -\infty E \quad \frac{\vdash_t x \preceq -\infty}{\vdash \bot}$$

$$\preceq nextI1\ \frac{\vdash_{t+x'} N \quad \vdash_t \neg N\,@\,(0,x') \quad \vdash 0 < x' \quad \vdash x \leq x'}{\vdash_t x \preceq +N}$$

$$\preceq nextI2\ \frac{\vdash_t \neg N\,@\,(0,+\infty)}{\vdash_t x \preceq +N}$$

$$\preceq nextE\ \frac{\vdash_t x \preceq +N \quad \frac{\vdash_{t+x'} N \quad \vdash_t \neg N\,@\,(0,x') \quad \vdash 0<x' \quad \vdash x\leq x'}{\vdash R}}{\vdash R}$$

$$\preceq prevI\ \frac{\vdash_{t+x'} P \quad \vdash_t \neg P\,@\,(x',0) \quad \vdash x' < 0 \quad \vdash x \leq x'}{\vdash_t x \preceq -P}$$

$$\preceq prevE\ \frac{\vdash_t x \preceq -P \quad \frac{\vdash_{t+x'} P \quad \vdash_t \neg P\,@\,(x',0) \quad \vdash x'<0 \quad \vdash x\leq x'}{\vdash R}}{\vdash R}$$

Introduction and elimination rules for **in** operator have been provided for each combination of interval parenthesis (open/close):

$$\mathbf{in}\,ccI\ \frac{\vdash_t x \succeq l \quad \vdash_t x \preceq u}{\vdash_t x\,\mathbf{in}\,[l,u]} \qquad \mathbf{in}\,ccE\ \frac{\vdash_t x\,\mathbf{in}\,[l,u] \quad \frac{\vdash_t x \succeq l \quad \vdash_t x \preceq u}{\vdash R}}{\vdash R}$$

$$\mathbf{in}\,ocI\ \frac{\vdash_t (x-1) \succeq l \quad \vdash_t x \preceq u}{\vdash_t x\,\mathbf{in}\,(l,u]} \qquad \mathbf{in}\,ocE\ \frac{\vdash_t x\,\mathbf{in}\,(l,u] \quad \frac{\vdash_t (x-1) \succeq l \quad \vdash_t x \preceq u}{\vdash R}}{\vdash R}$$

$$\mathbf{in}\,coI\ \frac{\vdash_t x \succeq l \quad \vdash_t (x+1) \preceq u}{\vdash_t x\,\mathbf{in}\,[l,u)} \qquad \mathbf{in}\,coE\ \frac{\vdash_t x\,\mathbf{in}\,[l,u) \quad \frac{\vdash_t x \succeq l \quad \vdash_t (x+1) \preceq u}{\vdash R}}{\vdash R}$$

$$\mathbf{in}\,ooI\ \frac{\vdash_t (x-1) \succeq l \quad \vdash_t (x+1) \preceq u}{\vdash_t x\,\mathbf{in}\,(l,u)} \qquad \mathbf{in}\,ooE\ \frac{\vdash_t x\,\mathbf{in}\,(l,u) \quad \frac{\vdash_t (x-1) \succeq l \quad \vdash_t (x+1) \preceq u}{\vdash R}}{\vdash R}$$

Bounded Happen

For bounded happen, a more complex formalization has been provided. Happen-min, $\mathbf{?}_m$, has been defined using a list datatype, in the following way:

$$P\,\mathbf{?}_m I = \exists\, f.length(f) = m \wedge (\forall i.i < m \rightarrow P(f[i]) \wedge f[i]\,\mathbf{in}\,I) \wedge (\forall 0 < j < m \rightarrow f[j-1] < f[j])$$

stating that exists a list f that enumerates m instants in I where P is true, and these instants are strictly monotone[1]. Note that the [ ] operator is used to access to the elements of the list starting from 0. Happen-max, $\mathbf{?}^M$, has been defined, using happen-min, as:

$$P\,\mathbf{?}^M I = \neg(P\,\mathbf{?}_{M+1} I)$$

stating that P happens at most M times in I if P does not happen more than $M+1$ times in I.

The introduction/elimination rules of bounded happen are based on inductive properties of happen-min and happen-max:

$$m \neq 0 \Rightarrow P\,\mathbf{?}_m[a,b] \Leftrightarrow \exists x \in [a,b].P@[x,x] \wedge (P\,\mathbf{?}_{m-1}[x+1,b] \vee P\,\mathbf{?}_{m-1}[a,x-1])$$

meaning that P happens at least $m > 0$ times in interval [a,b] iff exists a time instant x in the interval where P is true and P happens at least $m-1$ times before or after time instant x. Moreover property $P\,\mathbf{?}_0 I = \top$ is used to terminate the recursion.

[1] The strict monotone condition is not strictly necessary. Instants have only to be different, but in that case there exists a monotone list of instants where P is true. Therefore this definition has been used to have simpler proofs.

Similar properties hold for happen-max:

$$M \neq 0 \Rightarrow P\,\mathbf{?}^{M}[a,b] \;\Leftrightarrow\; \exists x \in [a,b].P@[x,x] \wedge ((\neg P@[a,x-1] \wedge P\,\mathbf{?}^{M-1}[x+1,b]) \vee (P\,\mathbf{?}^{M-1}[a,x-1] \wedge \neg P@[x+1,b]))$$

In this case, the additional constraint that P must not happen before or after time instant x has been added and property $P\,\mathbf{?}^{0} I = \neg P@I$ is used to terminate recursion.

The previous properties have been presented for a constant interval while similar ones hold for any type of interval. To avoid providing specific introduction/elimination rules for each kind of interval, functions sublft() and subrgt() have been introduced. sublft() and subrgt() are functions that from an interval and a value in it give the left or right subinterval excluding the given value - e.g., $\mathrm{sublft}([+B,10),x) = [+B,x)$ and $\mathrm{subrgt}([+B,10),x) = (x,10)$ so $I = \mathrm{sublft}(I,x) \cup \{x\} \cup \mathrm{subrgt}(I,x)$ holds.

Using the definition of happen-min and these functions the following inductive introduction/elimination rules have been proved:

$$?_0 I \qquad \frac{}{\vdash_t P\,\mathbf{?}_0 I}$$

$$\mathbf{?}_{min}rgtI \quad \frac{\vdash_{t+x} P \quad \vdash m \neq 0 \quad \vdash_t x\,\mathbf{in}\,I \quad \vdash_t P\,\mathbf{?}_{m-1}\,\mathrm{subrgt}(I,x)}{\vdash_t P\,\mathbf{?}_m I}$$

$$\mathbf{?}_{min}rgtE \quad \frac{\vdash_t P\,\mathbf{?}_m I \quad \dfrac{\vdash_{t+x} P \quad \vdash m \neq 0 \quad \vdash_t x\,\mathbf{in}\,I \quad \vdash_t P\,\mathbf{?}_{m-1}\,\mathrm{subrgt}(I,x)}{\vdash R}}{\vdash R}$$

$$\mathbf{?}_{min}lftI \quad \frac{\vdash_{t+x} P \quad \vdash m \neq 0 \quad \vdash_t x\,\mathbf{in}\,I \quad \vdash_t P\,\mathbf{?}_{m-1}\,\mathrm{sublft}(I,x)}{\vdash_t P\,\mathbf{?}_m I}$$

$$\mathbf{?}_{min}lftE \quad \frac{\vdash_t P\,\mathbf{?}_m I \quad \dfrac{\vdash_{t+x} P \quad \vdash m \neq 0 \quad \vdash_t x\,\mathbf{in}\,I \quad \vdash_t P\,\mathbf{?}_{m-1}\,\mathrm{sublft}(I,x)}{\vdash R}}{\vdash R}$$

Note that, there are different introduction/elimination rules for considering the interval split on the right or on the left.

Similarly for happen-max, the following rules have been derived:

$$\mathbf{?}^{max}I1 \qquad \frac{\vdash_t \neg P\,\mathbf{@}\,I}{\vdash_t P\,\mathbf{?}^M I}$$

$$\mathbf{?}^{max}rgtI2 \quad \frac{\vdash_{t+x} P \quad \vdash M \neq 0 \quad \vdash_t x\,\mathbf{in}\,I \quad \vdash_t \neg P\,\mathbf{@}\,\mathrm{sublft}(I,x) \quad \vdash_t P\,\mathbf{?}^{M-1}\,\mathrm{subrgt}(I,x)}{\vdash_t P\,\mathbf{?}^M I}$$

$$\mathbf{?}^{max}lftI2 \quad \frac{\vdash_{t+x} P \quad \vdash M \neq 0 \quad \vdash_t x\,\mathbf{in}\,I \quad \vdash_t \neg P\,\mathbf{@}\,\mathrm{subrgt}(I,x) \quad \vdash_t P\,\mathbf{?}^{M-1}\,\mathrm{sublft}(I,x)}{\vdash_t P\,\mathbf{?}^M I}$$

$$\mathbf{?}^{max}rgtE \quad \frac{\vdash_t P\,\mathbf{?}^M I \quad \dfrac{\vdash_t \neg P\,\mathbf{@}\,I}{\vdash R} \quad \dfrac{\vdash_{t+x} P \quad \vdash M \neq 0 \quad \vdash_t \neg P\,\mathbf{@}\,\mathrm{sublft}(I,x) \quad \vdash_t x\,\mathbf{in}\,I \quad \vdash_t P\,\mathbf{?}^{M-1}\,\mathrm{subrgt}(I,x)}{\vdash R}}{\vdash R}$$

$$\mathbf{?}^{max}lftE \quad \frac{\vdash_t P\,\mathbf{?}^M I \quad \dfrac{\vdash_t \neg P\,\mathbf{@}\,I}{\vdash R} \quad \dfrac{\vdash_{t+x} P \quad \vdash M \neq 0 \quad \vdash_t \neg P\,\mathbf{@}\,\mathrm{subrgt}(I,x) \quad \vdash_t x\,\mathbf{in}\,I \quad \vdash_t P\,\mathbf{?}^{M-1}\,\mathrm{sublft}(I,x)}{\vdash R}}{\vdash R}$$

Properties

Using these deduction rules and the induction principle some properties have been proved:

Intervals

$$
\begin{array}{l}
x \succeq b \Rightarrow (x+1) \succeq b \\
\neg(x \succeq b) \Rightarrow (x+1) \preceq b \\
x \,\mathbf{in}\, [b1, b3] \Rightarrow x \,\mathbf{in}\, [b1, b2] \vee x \,\mathbf{in}\, (b2, b3] \\
y \,\mathbf{in}\, \mathrm{sublft}(I, x) \wedge x \,\mathbf{in}\, I \Rightarrow y < x \\
y \,\mathbf{in}\, \mathrm{subrgt}(I, x) \wedge x \,\mathbf{in}\, I \Rightarrow x \,\mathbf{in}\, \mathrm{sublft}(I, y) \\
x \,\mathbf{in}\, I \wedge y \,\mathbf{in}\, \mathrm{sublft}(I, x) \Rightarrow \mathrm{sublft}(\mathrm{sublft}(I, x), y) = \mathrm{sublft}(I, y)
\end{array}
$$

TILCO-X
The following properties hold for happen-min:

$$
\begin{array}{rcll}
A\,?_1\, I & \Leftrightarrow & A\,?\, I & \\
A\,?_m\, I & \Rightarrow & A\,?_{m-1}\, I & \text{if } m > 0 \\
A\,?_m\, [a, b] & \Leftrightarrow & \perp & \text{if } m > b - a + 1 \\
A \wedge A\,?_m\, [0, b] & \Rightarrow\!\!\rhd & A\,?_{m-1}\, [0, b-1] & \text{if } m > 0 \\
\neg A \wedge A\,?_m\, [0, b] & \Rightarrow\!\!\rhd & A\,?_m\, [0, b-1] & \\
A\,?_m\, [a, 0] \wedge A\,@\,[1, 1] & \Rightarrow\!\!\rhd & A\,?_{m+1}\, [a-1, 0] & \\
A\,?_m\, [a, 0] \wedge \neg A\,@\,[1, 1] & \Rightarrow\!\!\rhd & A\,?_m\, [a-1, 0] & \\
\forall m.\, A\,?_m [0, +\infty) & \Leftrightarrow & (A?(0, +\infty))@[0, +\infty) & \\
A@[0, m) & \Rightarrow & A\,?_m [0, m) &
\end{array}
$$

and the following properties hold for happen-max:

$$
\begin{array}{rcll}
A\,?^0\, I & \Leftrightarrow & \neg A\,@\, I & \\
A\,?^M\, I & \Rightarrow & A\,?^{M+1}\, I & \\
A\,?^M\, [a, b] & \Leftrightarrow & \top & \text{if } M \geq b - a + 1 \\
A \wedge A\,?^M\, [0, b] & \Rightarrow\!\!\rhd & A\,?^{M-1}\, [0, b-1] & \text{if } M > 0 \\
\neg A \wedge A\,?^M\, [0, b] & \Rightarrow\!\!\rhd & A\,?^M\, [0, b-1] & \\
A\,?^M\, [a, 0] \wedge A\,@\,[1, 1] & \Rightarrow\!\!\rhd & A\,?^{M+1}\, [a-1, 0] & \\
A\,?^M\, [a, 0] \wedge \neg A\,@\,[1, 1] & \Rightarrow\!\!\rhd & A\,?^M\, [a-1, 0] & \\
A\,?^{M+1} [0, M] & \Leftrightarrow & \top &
\end{array}
$$

and

$$
A\,?_{min}^{max}\, I \Leftrightarrow \perp \qquad \text{if } min > max
$$

6 Specification example

In this section, a specification example to highlight the use of TILCO-X is presented. It is considered a system where a process has to respond to an external stimulus within 100ms. If the process does not respond within the given time, a controller has to retry up to 3 more times. If after all the temptatives the process does not respond the operation is aborted. After an abort the process cannot be started again for 500ms and an eventual request has to be

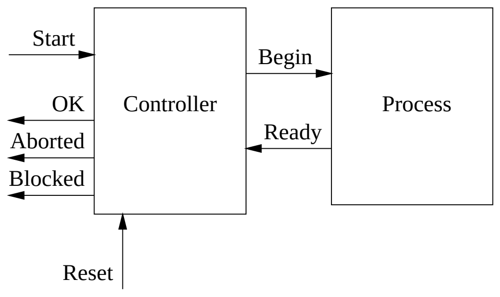

Fig. 8. An example of a process control system

ignored. If three consecutive operations are aborted the system is blocked until system reset by the user.

In Figure 8 the system structure is reported. *Process* is the process under control that has to respond to the *Begin* signal with a Ready signal being true within 100ms. The *Controller* is specified by using TILCO-X with the formulæ reported in the following. An internal signal *Enabled* is used to state that the Controller is enabled to consider the *Start* signal externally issued.

The response of the Controller to the *Start* signal is specified with the formulæ:

$$\begin{aligned} Start \wedge \neg Blocked \wedge Enabled &\Rightarrow Begin \\ Start \wedge \neg Enabled &\Rightarrow (\neg Begin \wedge \neg Aborted \wedge \neg Ok) \mathbin{@} [0, +Start) \\ Start \wedge Blocked &\Rightarrow (\neg Begin \wedge \neg Aborted \wedge \neg Ok) \mathbin{@} [0, +Reset) \end{aligned}$$

The first condition states that if the *Start* signal is true, the system is not Blocked and is Enabled the signal *Begin* is asserted. The second formula specifies the behavior of the system if it is not Enabled. In this case signals *Begin*, *Aborted* and *Ok* are false until the next Start. The last formula specifies the response to the *Start* signal if the system is Blocked. In this case signals *Begin*, *Aborted* and *Ok* are false until the next *Reset*.

When signal *Begin* is true, then the system has to wait for *Ready* within 100ms. If it happens signal Ok is true. This behavior can be specified with:

$$\begin{aligned} Begin \wedge Ready?(0,100] &\Rightarrow (\neg Begin \wedge \neg Aborted \wedge \neg Enabled \wedge (Ready \Leftrightarrow Ok)) \mathbin{@} (0, +Ready] \\ Begin \wedge \neg Ready@(0,100] &\Rightarrow (\neg Begin \wedge \neg Aborted \wedge \neg Ok \wedge \neg Enabled) \mathbin{@} (0, 100) \\ Begin &\Rightarrow \neg Aborted \wedge \neg Ok \end{aligned}$$

The condition on the repetition of the *Begin* signal is specified using Bounded Happen:

$$\begin{aligned} Begin \mathbin{@} [-100] \wedge \neg Ready \mathbin{@} (-100, 0] \wedge Begin\, ?^{3}[-(Start \wedge Enabled), 0) &\Rightarrow Begin \wedge \neg Enabled \\ Begin \mathbin{@} [-100] \wedge \neg Ready@(-100, 0] \wedge Begin\, ?_{4}[-(Start \wedge Enabled), 0) &\Rightarrow Aborted \wedge \neg Ok \end{aligned}$$

The first formula specifies that, if the last *Begin* has failed and the *Begin* has been issued up to 3 times, since the last enabled Start, then the *Begin* has to be retried. The second formula specifies the case in which *Begin* has been retried more than 3 times, and in this case, the signal *Aborted* is asserted.

The behavior of the *Enabled* signal is specified with formulæ:

$$\begin{aligned} Aborted &\Rightarrow \neg Begin \wedge \neg Enabled \mathbin{@} [0, 500) \wedge (Enabled \mathbin{@} [0, +Start])@[500] \\ Ok \vee (Reset \wedge Enabled) &\Rightarrow Enabled \mathbin{@} (0, +Start] \end{aligned}$$

The first formula specifies that, when *Aborted* is true the system is not enabled to satisfy a Start request for 500ms and after this period the system is enabled until a Start is received. Similarly, when Ok or Reset is true the system is enabled until the next Start.

The system is Blocked if and only if *Aborted* is true more than 3 times since last *Ok* or last *Reset*, in formula:

$$Blocked \Leftrightarrow Aborted ?_3(-(Ok \vee Reset), 0)$$

After *Ok* or *Aborted* are true or after a *Reset* signals *Begin*, *Aborted* and *Ok* are false:

$$Ok \vee Aborted \Rightarrow (\neg Begin \wedge \neg Aborted \wedge \neg Ok) @ (0, +(Start \wedge Enabled))$$
$$Reset \wedge Enabled \Rightarrow (\neg Begin \wedge \neg Aborted \wedge \neg Ok) @ [0, +Start)$$

Validation

This specification has been firstly validated by using the TILCO-X executor (presented in the next chapter). In Figure 9 temporal traces of the system execution are reported. A *Reset* at time 0 has been issued for the proper initialization, after that, signal *Start* is asserted and since the *Ready* signal is false the signal Begin is issued four times and then the signal *Aborted* is trued. The signal Start is asserted other two times and since the Process does not response the operations are aborted. The failure of three operations brings up the *Blocked* signal. The *Reset* is issued and enables the system to the receipt for the *Start* signal. The *Start* is issued again and at the second temptative the *Ready* signal is received and *Ok* asserted.

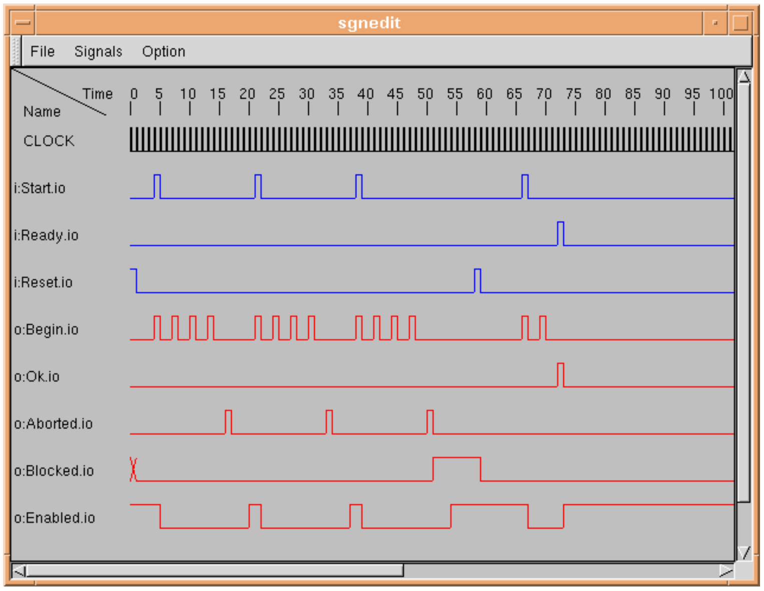

Fig. 9. Execution of the TILCO-X specification

Moreover, some properties has been proved using the TILCO-X theory within Isabelle:

$$Start \wedge Enabled \Rightarrow \neg Enabled @ (0, +(Ok \vee Aborted)]$$
$$Start \wedge Enabled \Rightarrow Enabled \, ? \, (0, +\infty)$$
$$Start \wedge Enabled \Rightarrow (Ok \vee Aborted) \, ? \, (0, +\infty)$$
$$Start \wedge Enabled \Rightarrow Ok \, ? \, (0, +500) \vee Aborted @ [500]$$

The first property states that the controller is not enabled until the termination (with success or failure); the second property states that, if the process is started there will be a future instant in which it will be enabled to be started again; the third property states that, if the process is started it will terminate with success or failure; the last property gives more details on when signals Ok and Aborted will be true.

7 Conclusions

This paper has described an extension of TILCO, named TILCO-X, a temporal logic for the specification, validation and verification of real-time systems. The introduction of the Bounded Happen and Dynamic Interval operators enhanced the expressive power of TILCO. TILCO-X enhanced the readability and conciseness of formulas with respect to TILCO, especially for the order requirements removing the differences between past and future but maintaining at the same time the implicit time specifications.

In summary, TILCO-X differs from other temporal logics proposed in the literature. TILCO-X is a first order interval logic that (i) provides a metric for time (thus allowing specification of qualitative and quantitative timing constraints); (ii) presents a linear implicit time model; (iii) adopts a uniform manipulation of intervals from past to future for actions, events, event ordering; (iv) present specific operators for defining temporal constraints including the counting the occurrence of events; and (iv) provides decidability for a wide set of formulæ (non-temporal quantifications must bind only variables with types over finite domains). In TILCO-X no explicit quantification over the temporal domain is allowed and with the new operators this limitation Is strongly less relevant since the demand of quantification has been reduced with respect to the old TILCO version.

Since TILCO-X is particularly suitable for requirements analysis and the incremental specification of real-time systems. TILCO-X supports validation during all phases of the system life-cycle by means of its formalisation in the automatic theorem prover Isabelle/HOL. This allows validation for refinement and the proof of properties. Moreover, the final operational validation is also supported by a *TILCO-X Executor*, which allows execution and the model-checking of systems specifications.

References

1. G. Bucci, M. Campanai, and P. Nesi, "Tools for specifying real-time systems," *Journal of Real-Time Systems*, vol. 8, pp. 117–172, March 1995.
2. A. D. Stoyenko, "The evolution and state-of-the-art of real-time languages," *Journal of Systems and Software*, pp. 61–84, April 1992.
3. A. Pnueli, "The temporal logic of programs," in *18th IEEE FOCS*, 1977. mattolini.
4. F. Jahanian and A. K.-L. Mok, "Safety analysis of timing properties in real-time systems," *IEEE Transactions on Software Engineering*, vol. 12, pp. 890–904, Sept. 1986.
5. R. L. Schwartz, P. M. Melliar-Smith, and F. H. Vogt, "A interval logic for higher-level temporal reasoning," in *Proc. of the 2nd Annual ACM Symp. Principles of Distributed Computing, Lecture Notes in Computer Science, LNCS N. 164*, (Montreal Canada), pp. 173–186, Springer Verlag, ACM NewYork, 17-19 Aug. 1983.
6. R. Gotzhein, "Temporal logic and applications – a tutorial," *Computer Networks and ISDN Systems, North-Holland*, vol. 24, pp. 203–218, 1992.
7. R. Mattolini and P. Nesi, "An interval logic for real-time system specification," *IEEE Transactions on Software Engineering, in press*, 1999.
8. J. S. Ostroff, "Formal methods for the specification and design of real-time safety critical systems," *Journal of Systems and Software*, pp. 33–60, April 1992.
9. R. Alur and T. A. Henzinger, "A really temporal logic," in *30th IEEE FOCS*, 1989.
10. J. S. Ostroff, *Temporal Logic for Real-Time Systems.* Taunton, Somerset, England: Research Studies Press LTD., Advanced Software Development Series, 1, 1989. NO.

11. B. C. Moszkowski, *Executing Temporal Logic Programs.* PhD thesis, Cambridge University, 1986.
12. C. Ghezzi, D. Mandrioli, and A. Morzenti, "Trio, a logic language for executable specifications of real-time systems," *Journal of Systems and Software*, vol. 12, pp. 107–123, May 1990.
13. R. Koymans, *Specifying Message Passing and Time-Critical Systems with Temporal Logic.* No. 651, Lecture Notes in Computer Science,Springer-Verlag, 1992.
14. J. F. Allen and G. Ferguson, "Actions and events in interval temporal logic," tech. rep., University of Rochester Computer Science Department, TR-URCSD 521, Rochester, New York 14627, July 1994.
15. R. Alur and T. A. Henzinger, "Real time logics: complexity and expressiveness," in *Proc. of 5th Annual IEEE Symposium on Logic in Computer Science, LICS 90*, (Philadelphia, USA), pp. 390–401, IEEE, June 1990. YES in Kavi92, TR STANCS901307, Dept. of Comp. Science and Med, Stanford.
16. Z. Manna and A. Pnueli, "Proving precedence properties: The temporal way," in *Proc. of the 10th Colloquium on Automata Languages and Programming, Lecture Notes in Computer Science LNCS N.154* (J. Diaz, ed.), (Barcelona, Spain), pp. 491–512, Springer Verlag, July 1983.
17. R. Rosner and A. Pnueli, "A choppy logic," in *Proc. of the 1st IEEE Symp. on Logic in Computer Science*, pp. 306–313, IEEE Press, 1986.
18. J. Y. Halpern and Y. Shoham, "A propositional modal logic of time intervals," in *Proc. of the 1st IEEE Symp. on Logic in Computer Science*, pp. 274–292, IEEE Press, 1986.
19. P. M. MelliarSmith, "Extending interval logic to real time systems," in *Proc. of Temporal Logic Specification United Kingdom, (B. Banieqbal, H. Barringer, A. Pnueli, eds)*, pp. 224–242, Springer Verlag, Lecture Notes in Computer Sciences, LNCS 398, April 1987.
20. R. R. Razouk and M. M. Gorlick, "A real-time interval logic for reasoning about executions of real-time programs," in *Proc. of the ACM/SIGSOFT'89, Tav.3*, pp. 10–19, ACM Press, Dec. 1989.
21. L. K. Dillon, G. Kutty, L. E. Moser, P. M. Melliar-Smith, and Y. S. Ramakrishna, "A graphical interval logic for specifying concurrent systems," *ACM Transactions on Software Engineering and Methodology*, vol. 3, pp. 131–165, April 1994.
22. J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, pp. 832–843, Nov. 1983.
23. M. Ben-Ari, *Mathematical Logic for Computer Science.* New York: Prentice Hall, 1993.
24. T. A. Henzinger, Z. Manna, and A. Pnueli, "Temporal proof methodologies for real-time systems," in *Proc. of the 18th ACM Symposium on Principles of Programming Languages*, pp. 353–366, ACM, 1991.
25. R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Systems Journal*, vol. 2, pp. 255–299, 1990.
26. R. Alur and T. A. Henzinger, "Real-time logics: Complexity and expressiveness," tech. rep., Dept. of Comp. Science and Medicine STAN-CS-90-1307, Stanford University, Stanford, California, USA, March 1990.
27. L. C. Paulson, *Isabelle: A Generic Theorem Prover.* Lecture Notes in Computer Science, Springer Verlag LCNS 828, 1994.
28. R. Mattolini, *TILCO: a Temporal Logic for the Specification of Real-Time Systems (TILCO: una Logica Temporale per la Specifica di Sistemi di Tempo Reale).* PhD thesis, 1996.
29. R. A. McCauley and W. R. Edwards, "Analysis and measurement techniques for logic-based languages," in *Proc. of the 2nd Euromicro Conference on Software Maintenance and Reengineering*, (Florence, Italy), IEEE Press, 8-11 March 1998.
30. S. N. Cant, D. R. Jeffery, and B. Henderson-Sellers, "A conceptual model of cognitive complexity of elements of the programming process," tech. rep., University of New South Wales, Information Technology Research Centre, n.57, New South Wales 2033, Australia, Oct. 1991.
31. H. Zuse, *Software Complexity: measures and methods.* Berlin, New-York: Walter de Cruyter, 1991.
32. S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models.* Benjaming/Cumminggs, Publ. Co., 1986.
33. P. Nesi and M. Campanai, "Metric framework for object-oriented real-time systems specification languages," *The Journal of Systems and Software*, vol. 34, pp. 43–65, 1996.
34. P. Bellini and P. Nesi, "Temporal logics for real-time system specification," *Submitted to ACM Computing Surveys, also Technical Report of Department of Systems and Informatics, University of Florence, Italy*, 1999.

Table of Contents