Generating Linear Temporal Logic Formulas for Pattern-Based Specifications

Salamah Salamah, Vladik Kreinovich, and Ann Q. Gates
Dept. of Computer Science, University of Texas at El Paso
El Paso, TX 79968, USA
isalamah, vladik, and agates@utep.edu

Abstract

Software property classifications and patterns, i.e., high-level abstractions that describe program behavior, have been used to assist practitioners in specifying properties. The Specification Pattern System (SPS) provides descriptions of a collection of patterns. Each pattern is associated with a scope that defines the extent of program execution over which a property pattern is considered. Based on a selected pattern, SPS provides a specification for each type of scope in multiple formal languages including Linear Temporal Logic (LTL). The Property Specification tool (Prospec) extends SPS by introducing the notion of Composite Propositions (CP), that classify sequential and concurrent behavior over pattern and scope parameters.

In this work, we present an approach to support the automated generation of Linear Temporal Logic (LTL) formulas for complex pattern-based software specifications that use CPs. We define general LTL formulas for the Response pattern, and provide formal descriptions of the different CP classes. In addition, we formally describe the Response pattern and the different scopes that use CPs.

1 Introduction and Motivation

Although formal verification methods such as model checking [6], theorem proving [12], and runtime monitoring [14] have been shown to improve the dependability of programs, software development professionals have yet to adopt them. The reasons for this hesitance include the high level of mathematical sophistication required for reading and writing formal specifications needed for the use of these approaches [5].

Linear Temporal Logic (LTL) is a prominent formal specification language. LTL's popularity stems from the fact that it is highly expressive and that it is widely used in formal verification tools. Such tools include, the popular model checker SPIN [6] that has been used in the verification of a variety of systems such as security protocols [1] and flight software [8]. In addition, LTL is used by the Model checkers NuSMV [2] and Java Path-Finder [7]. LTL is also used in the runtime verification of Java programs [14].

1.1 Specification Pattern System (SPS)

The problem of generating formal specification is difficult, and the temporal nature of LTL makes it even harder to write specifications. The Specification Pattern System [3] defines patterns and scopes to assist the practitioner in formally specifying software properties.

Patterns capture the expertise of developers by describing solutions to recurrent problems. Each pattern describes the structure of specific behavior, defines the pattern's relationship with other patterns, and defines the scope over which the property holds.

The main patterns defined by SPS are: $Universality$ (P), $Absence(P)$, $Existence(P)$, $Precedence(P, Q)$, and $Response(P, Q)$. $Universality(P)$ states that P is true in every point of the execution; $Absence(P)$ states that P is never true during the execution; $Existence(P)$ states that P is true at some point in the execution; $Precedence(P, Q)$ states that if P holds, then Q must hold before P; and $Response(P, Q)$ states that if P holds, then Q must hold at a future state. Response properties represent a temporal relation called cause-effect between two propositions. SPS restricts the specification of sequences to precedence and response patterns.

In SPS, each pattern is associated with a *Scope* that defines the extent of program execution over which a property pattern is considered. There are five types of scopes defined in SPS: $Global$, $Before\ R$, $After\ L$, $Between\ L\ And\ R$, and $After\ L\ Until\ R$. $Global$ denotes the entire program execution; $Before\ R$ denotes the execution before the first time R occurs, i.e., R holds; $After\ L$ denotes execution after the first time L occurs; $Between\ L\ And\ R$ denotes the execution between intervals defined by L and R; and $After\ L\ Until$ denotes the execution between intervals defined by L and R and, in the case when R does not occur, until the end of execution.

The SPS website[4] provides a formal description of patterns and scopes in several specification languages including Linear Temporal Logic (LTL), Computational Tree Logic (CTL), and Graphical Interval Logic (GIL). These formulas are provided for patterns and scopes involving *single propositions*, i.e., patterns and scopes in which P, Q, L, and R each occur at a single moment of time. SPS also provides formulas for the cases of multiple cause and single effect (when P is made of multiple propositions and Q is single) and of single effect and multiple cause (when P is single and Q is composed of multiple propositions).

1.2 Composite Propositions (CP)

In practical applications, we often need to describe properties where one or more of the pattern or scope parameters are made of multiple propositions, i.e., composite propositions (CP). For example, the property that every time data is sent at moment t_i the data is read at moment $t_1 \geq t_i$, the data is processed at moment t_2, and data is stored at moment t_3. This property can be described using the Response pattern where P stands for "Data is sent', and Q is composed of q_1, q_2, and q_3 (data is read, data is processed, and data is stored, respectively).

To describe such patterns, Mondragon et al. [10] extended SPS by introducing a classification for defining sequential and concurrent behavior to describe patterns and scopes parameters. Specifically, the work formally described several types of composite propositions (CP) and showed how these descriptions can be translated into LTL.

Some of the corresponding patterns can be described in a Future Interval Logic (FIL) language, a language which is similar to LTL, but less expressive than LTL. For example, in FIL, we cannot describe a practically important property that an event p must hold at the next moment of time. The corresponding translations have been implemented in the Property Specification tool (Prospec) [9] that uses patterns and scopes involving composite propositions to generate formal specifications in FIL. Similarly to LTL specifications, FIL specifications can also be used to formally verify software. However, in comparison to LTL, FIL has two limitations: first, due to the limited expressiveness of FIL, not all patterns and scopes involving composite propositions can be represented; second, FIL is not as widely used in formal verification tools, so the use of FIL restricts the software engineer's ability to use the resulting specifications.

It is, therefore, important to provide a translation of all possible patterns and scopes involving composite propositions into the more expressive (and more widely used) language LTL. It is also important to show that these translations are indeed correct for all patterns and scopes. In this paper, due to page limitations, we concentrate on the case of the most widely used $Response$ pattern [3]. The matter of proving the correctness of these translations is left as a future work.

The rest of the paper is outlined as follows. We start with a brief description of LTL. Section 3 provides a formal description of the different CP classes. Section 4 formally describes the $Response$ pattern, and Section 5 provides the description of the LTL formulas for the $Response$ pattern within the $Global$ scope. Sections 6 and 7 provide formal definition of the other scopes and describe the LTL formulas for the $Response$ pattern within these scopes.

2 Linear Temporal Logic: A Brief Description

Formulas of LTL are constructed inductively from elementary propositions $p_1, p_2, \ldots$ by applying Boolean connectives $\neg$, $\vee$, and $\wedge$ and temporal operators X (*next*), U (*until*), $\diamond$ (*eventually*), and $\Box$ (*always*). These formulas assume discrete time, i.e., moments $t = 0, 1, 2, \ldots$ The meaning of the temporal operators is straightforward. The formula XP holds at the moment t means that P holds at the next moment of time $t+1$. To check whether $P\,U\,Q$ holds at the moment t, we must find the first moment of time $s \geq t$ at which Q is true; then, the truth of $P\,U\,Q$ means that P is true at all moments of time t' for which $t \leq t' < s$ (if Q never happens at moment t or later, then $P\,U\,Q$ is false). The formula $\diamond P$ holds at moment t means P is true at some moment of time $t' \geq t$. Finally, the formula $\Box P$ holds at moment t if P is true at all moments of time $t' \geq t$.

3 Composite Propositions: A Formal Description

We consider the following 8 CP classes: $AtLeastOne_C$, $AtLeastOne_E$, $Parallel_C$, $Parallel_E$, $Consecutive_C$, $Consecutive_E$, $Eventual_C$, and $Eventual_E$. CP classes of type T_C (called *condition type*) are defined as follows:

- $AtLeastOne_C(p_1, \ldots, p_n)$ means that at least one of p_i holds at a given moment of time t, i.e., that $p_1 \vee \ldots \vee p_n$ holds;
- $Parallel_C(p_1, \ldots, p_n)$ means that all p_i hold at time t, i.e., $p_1 \wedge \ldots \wedge p_n$;
- $Consecutive_C(p_1, \ldots, p_n)$ means that p_1 holds at moment $t_1 = t$, p_2 holds at moment $t_2 = t+1$, ..., and p_n holds at moment $t_n = t + (n-1)$; the corresponding LTL formula $Consecutive_C^{LTL}(p_1, \ldots, p_n)$ is $(p_1 \wedge X(p_2 \wedge X(\ldots \wedge X(p_n)) \ldots))$;

- $Eventual_C(p_1, \ldots, p_n)$ means that p_1 holds at $t_1 = t$, p_2 holds at some moment $t_2 > t_1$, ..., and p_n holds at some moment $t_n > t_{n-1}$; the corresponding LTL formula $Eventual_C^{LTL}(p_1, \ldots, p_n)$ is

$$p_1 \wedge X(\neg p_2 \, U \, (p_2 \wedge X(\neg p_3 \, U \, (\ldots \wedge X(\neg p_n \, U \, p_n)) \ldots))).$$

CP classes of the type T_E (called *event type*) can be defined in terms of a new class of auxiliary formulas $T_H(p_1, \ldots, p_n)$. The main motivation for T_H is that in T_C we only required each p_i to hold at a certain moment of time t_i, and we do not make any assumptions about other propositions p_j ($j \neq i$) at this moment t_i. In some practical applications, it is important to require that p_i become true in the prescribed order, i.e., that not only p_i becomes true at moment t_i, but that it also remains false until then. In precise terms, we have the following:

Definition 1

- *By a* CP class*, we mean one of the following four terms: $AtLeastOne$, $Parallel$, $Consecutive$, and $Eventual$.*
- *By a* type of proposition*, we will mean C, E, or H; the type H will be called* auxiliary.
- *By a* composite proposition *P, we mean an expression of the type $T_y(p_1, \ldots, p_n)$, where T is a CP class, y is a type of proposition, and $p_1, \ldots, p_n$ are (single) propositions.*

Definition 2

- *For T = AtLeastOne and for T = Parallel, $T_H(p_1, \ldots, p_n)$ means the same as $T_C(p_1, \ldots, p_n)$, and $T_H^{LTL}(p_1, \ldots, p_n)$ is defined as $T_C^{LTL}(p_1, \ldots, p_n)$.*
- *For $T = Consecutive$ and for $T = Eventual$, $T_H(p_1, \ldots, p_n)$ means*

$$T_C(p_1 \wedge \neg p_2 \wedge \ldots \wedge \neg p_n, p_2 \wedge \neg p_3 \wedge \ldots \wedge \neg p_n, \ldots, p_n),$$

and $T_H^{LTL}(p_1, \ldots, p_n)$ is defined as

$$T_C^{LTL}(p_1 \wedge \neg p_2 \wedge \ldots \wedge \neg p_n, p_2 \wedge \neg p_3 \wedge \ldots \wedge \neg p_n, \ldots, p_n).$$

For example, $Consecutive_H(p_1, \ldots, p_n)$ holds at the moment t if:

- at the moment of time t, the proposition p_1 holds and all the further propositions $p_2, \ldots, p_n$ are false;
- at the next moment of time $t+1$, the proposition p_2 holds and all the further propositions $p_3, \ldots, p_n$ are false; ..., and
- at the moment $t + (n-1)$, the proposition p_n holds.

In other words, $Consecutive_H(p_1, \ldots, p_n)$ holds at the moment t if the following formula for T_H^{LTL} holds at moment t:

$$(p_1 \wedge \neg p_2 \wedge \neg p_3 \wedge \ldots \wedge \neg p_n \wedge$$
$$X(p_2 \wedge \neg p_3 \wedge \ldots \wedge \neg p_n \wedge$$
$$X(\ldots \wedge X(p_{n-1} \wedge \neg p_n \wedge X(p_n)) \ldots)))$$

Definition 3 *We say that a composite proposition $T_E(p_1, \ldots, p_n)$ holds at the moment t if at the moment t, all propositions p_i are false, and they remain false until some moment t' when the composite proposition $T_H(p_1, \ldots, p_n)$ becomes true.*

For example, a composite proposition $AtLeastOne_E(p_1, \ldots, p_n)$ holds at moment t if all the propositions $p_1, \ldots, p_n$ are false at the moment t, and at least one of these propositions $p_1, \ldots, p_n$ is true at some future moment of time $t' > t$.

Definition 4 *By an* LTL formula $T_E^{LTL}(p_1, \ldots, p_n)$ *for $T_E(p_1, \ldots, p_n)$, we mean the formula*

$$(\neg p_1 \wedge \neg p_2 \wedge \ldots \wedge \neg p_n) \wedge$$
$$((\neg p_1 \wedge \neg p_2 \wedge \ldots \wedge \neg p_n) \, U \, T_H^{LTL}(p_1, p_2, \ldots, p_n)).$$

Theorem 1 *For every composite proposition P and for every moment of time t, P holds at the moment t if and only if the corresponding LTL formula P^{LTL} holds at this moment t.*

Comment. Due to page limitations, detailed proofs are given in [13].

4 Response Pattern within Global Scope: Case of Composite Propositions

Patterns such as $Response$ were introduced in [3] for single propositions. In particular, "q responds to p within global scope" means that every time the property p holds, the property q must hold either after it or at this same moment of time. To extend this description to composite propositions, we therefore need to extend the notion "after" to such propositions.

Single propositions describe a single moment of time. In general, composite propositions deal with a time interval (although, of course, this time interval may be degenerate, i.e., it may consist of a single moment of time). Specifically, for every composite proposition $P = T(p_1, \ldots, p_n)$, there is a starting moment s_P – the first moment of time when one of the propositions p_i becomes true, and the ending moment e_P – the first moment of time when the condition T is fulfilled. These moments can be defined as follows.

Definition 5

- *For a composite proposition P of the type $AtLeastOne_C(p_1, \ldots, p_n)$ that holds at the moment t, we take $s_P(t) = e_P(t) = t$.*
- *For a composite proposition P of the type $AtLeastOne_E(p_1, \ldots, p_n)$ that holds at the moment t, we take, as $s_P(t) = e_P(t)$, the first moment of time $t' > t$ at which one of the propositions p_i becomes true.*
- *For a composite statement P of the type $Parallel_C(p_1, \ldots, p_n)$ that holds at the moment t, we take $s_P(t) = e_P(t) = t$.*
- *For a composite proposition P of the type $Parallel_E(p_1, \ldots, p_n)$ that holds at the moment t, we take, as $s_P(t) = e_P(t)$, the first moment of time $t' > t$ at which all the propositions p_i become true.*
- *For a composite proposition P of the type $Consecutive_C(p_1, \ldots, p_n)$ that holds at the moment t, we take $s_P(t) = t$ and $e_P(t) = t + (n - 1)$.*
- *For a composite proposition P of the type $Consecutive_E(p_1, \ldots, p_n)$ that holds at the moment t, we take, as $s_P(t)$, the first moment of time $t' > t$ at which the proposition p_1 becomes true, and we take $e_P(t) = t' + (n - 1)$.*
- *For a composite proposition P of the type $Eventual_C(p_1, \ldots, p_n)$ that holds at the moment t, we take $s_P(t) = t$, and as $e_P(t)$, we take the first moment of time $t_n > t$ at which the last proposition p_n is true and the previous propositions $p_2, \ldots, p_{n-1}$ were true at the corresponding moments of time $t_2, \ldots, t_{n-1}$ for which $t < t_2 < \ldots < p_{n-1} < t_n$.*
- *For a composite proposition P of the type $Eventual_E(p_1, \ldots, p_n)$ that holds at the moment t, we take as $s_P(t)$, the first moment of time t_1 at which the first proposition p_1 becomes true, and as $e_P(t)$, the first moment of time t_n at which the last proposition p_n becomes true.*

Definition 6 *Let P and Q be composite propositions. We say that Q* responds to P within global scope *if once P holds at some moment t, then Q also holds at some moment t' for which $s_Q(t') \geq e_P(t)$.*

5 LTL Formulas For Response Pattern Within Global Scope: Case of Composite Propositions

To describe LTL formulas $\mathcal{P}^{LTL}$ corresponding to patterns $\mathcal{P}$ with composite propositions, we need to describe new "and" operations. The need for these new operations comes from the temporal character of the logic. In non-temporal logic, the formula $A \wedge B$ simply means that both A and B are true. In particular, if we consider a non-temporal formula A as a particular case of LTL formulas, then A means simply that the statement A holds at the given moment of time, and the formula $A \wedge B$ means that both A and B hold at this same moment of time.

For a general LTL formula A, the fact that A holds at a moment of time t means that some "subformulas" of A hold at this same moment of time, while some other subformulas may hold at different moments of time. For example, the formula $p_1 \wedge Xp_2$ means that p_1 holds at the moment t while p_2 holds at the moment $t+1$; the formula $p_1 \wedge X \diamond p_2$ means that p_1 holds at the moment t and p_2 holds at some future moment $t_2 > t$, etc. If we simply write $A \wedge B$ that would mean that different subformulas of A hold at the corresponding different moments of time but B only holds at the original moment t. It is therefore desirable to come up with a different "and" operation that would ensure that B holds at all the moments of time at which different "subformulas" of A hold. For example, for this new "and" operation, $(p_1 \wedge Xp_2)$ and B would mean that B holds both at the moment t and at the moment $t + 1$.

Similarly, $(p_1 \wedge X \diamond p_2)$ and B should mean that B holds both at the moment t and at the moment $t_2 > t$ when p_2 holds. In other words, we want to state that at the original moment of time t, we must have $p_1 \wedge B$, and that at some moment of time $t_2 > t$, we must have $p_2 \wedge B$. This can be described as $(p_1 \wedge B) \wedge X \diamond (p_2 \wedge B)$.

To distinguish this new "and" operation from the original LTL operation $\wedge$, we will use a different "and" symbol & to describe this new operation. However, this symbol by itself is not sufficient since people use & in LTL as well; so, to emphasize that our "and" operation means "and" applied at several different moments of time, we will use a combination $\&_r$ of several & symbols.

In addition to the original "and" $A \wedge B$ which means that B holds at the original moment of time t and to the new "repeated and" $A \&_r, B$ meaning that B holds at all moments of time that are relevant for the LTL formula A, we will also need the operation $A \&_l B$, which will indicate that B holds at the *last* of A-relevant moments of time. Let us give formal definitions of these operations. We only give the definition for the particular cases needed in our patterns.

Definition 7

- *When P is of the type $T_C(p_1, \ldots, p_n)$ or $T_H(p_1, \ldots, p_n)$, with T = Parallel or T = AtLeastOne, then $P \&_r A$ is defined as $P \wedge A$.*
- *When P is of the type $T_C(p_1, \ldots, p_n)$, with T = Consecutive or T = Eventual, then $P \&_r A$ is defined as $T_C(p_1 \wedge A, \ldots, p_{n-1} \wedge A, p_n \wedge A)$.*

- *When P is of the type $T_H(p_1,\ldots,p_n)$, with $T = Consecutive$ or $T = Eventual$, then $P \,\&_r\, A$ is defined as*

  $$T_C(p_1\wedge\neg p_2\wedge\ldots\wedge\neg p_n\wedge A,\ldots,p_{n-1}\wedge\neg p_n\wedge A,p_n\wedge A).$$

- *When P is of the type $T_E(p_1,\ldots,p_n$, then $P \,\&_r\, A$ is defined as*

  $$(\neg p_1\wedge\ldots\wedge\neg p_n\wedge A)\wedge$$

  $$((\neg p_1\wedge\ldots\wedge\neg p_n\wedge A)\,U\,(T_H(p_1,\ldots,p_n\wedge A)))).$$

Definition 8

- *When P is of the type $T_C(p_1,\ldots,p_n)$ or $T_H(p_1,\ldots,p_n)$, with T = Parallel or T = AtLeastOne, then $P \,\&_l\, A$ is defined as $P\wedge A$.*

- *When P is of the type $T_C(p_1,\ldots,p_n)$, with $T = Consecutive$ or $T = Eventual$, then $P \,\&_l\, A$ is defined as $T_C(p_1,\ldots,p_{n-1},p_n\wedge A)$.*

- *When P is of the type $T_H(p_1,\ldots,p_n)$, with $T = Consecutive$ or $T = Eventual$, then $P \,\&_l\, A$ is defined as*

  $$T_C(p_1\wedge\neg p_2\wedge\ldots\wedge\neg p_n,\ldots,p_{n-1}\wedge\neg p_n,p_n\wedge A).$$

- *When P is of the type $T_E(p_1,\ldots,p_n)$, then $P \,\&_l\, A$ is defined as*

  $$(\neg p_1\wedge\ldots\wedge\neg p_n)\wedge$$

  $$((\neg p_1\wedge\ldots\wedge\neg p_n)\,U\,(T_H(p_1,\ldots,p_n)\,\&_l A)).$$

Definition 9 *An LTL formula corresponding to $Response$ (P,Q) is*

$$\Box(P^{LTL}\rightarrow(P^{LTL}\,\&_l\,\Diamond Q^{LTL})).$$

For example, if P is of type $Consecutive_C(p_1,p_2)$ and Q is of type $Parallel_E(q_1,q_2)$, then $\mathcal{P}_1^{LTL}$ is:

$$\Box((p_1\wedge Xp_2)\rightarrow(p_1\wedge X(p_2\wedge\Diamond$$

$$((\neg q_1\wedge\neg q_2)\wedge((\neg q_1\wedge\neg q_2)\,U\,(q_1\wedge q_2))))))$$

Theorem 2 *For the formula "Q responds to P within a global scope", this formula holds at the moment t if and only if the corresponding LTL formula holds at this moment t.*

Comment. Similar results hold for all other patterns from Section 1.1.

6 Other Scopes: Motivations and Definitions

In the previous text, we considered all the propositions in the "global scope", when, e.g., the existence of P means that P holds for some moment of time t. In practice, we are often only interested in moments of time that occur before or after a certain event. For single propositions, the following scopes were proposed in [3]:

- "before R" means that we only consider the moments of time which strictly precede the first occurrence of R; if R never occurs then we do not consider any moments of time at all;

- "after L" means that we only consider the moments of time which are equal to or follow the first occurrence of L; if L never occurs then, of course, we do not consider any moments of time at all;

- "between L and R" means that we only consider moments of time between each occurrence of L (including the moment of this occurrence) and the corresponding next occurrence of R (not including that moment);

- "after L until R" means that we only consider moments of time after each occurrence of L and until the next occurrence of R.

In the scope "after L until R", if R never happens after a certain occurrence of L, we consider all moments of time after this occurrence of L. In contrast, in the scope "between L and R", if R never happens after a certain occurrence of L, we do not consider any moments of time at all.

We want to extend the above definitions of patterns to the case of scopes. For example, we want to define "Q precedes P within a scope s", meaning that every time P holds within the scope s, Q should hold at a preceding moment of time within the same scope.

In general, a composite proposition P holds at a moment t if several single propositions hold at different moments of time. It is therefore reasonable to say that P occurs within a scope if all these single propositions occur within this scope, i.e., if the whole time interval $[t,e_P(t)]$ is within the scope s. Let us formally describe what this means.

Since we consider composite propositions P, it is natural to allow L and R to be composite propositions as well. A composite proposition P, in general, does not occur at a single moment of time, it usually has a starting moment s_P and the ending moment e_P. So, for composite propositions, "before R" can be naturally interpreted as "before the starting moment of R", and "after L" can be naturally interpreted as "after the ending moment of L". To describe this formally, we will use the above definitions (of section 4) of the starting and ending moments.

Definition 10 *By a* scope statement*, we mean a statement of the type "before R", "after L", "between L and R", and "after L until R", where L and R are composite propositions.*

Definition 11 *By a* scope *corresponding to the scope statement, we mean the following time interval:*

- *For a scope statement "before R", there is exactly one scope – the interval $[0, s_R(t_f))$, where t_f is the first moment of time when R becomes true.*
- *For a scope statement "after L", there is exactly one scope – the interval $[e_L(t_f), \infty)$, where t_f is the first moment of time when L becomes true.*
- *For a scope statement "between L and R", a scope is an interval $[e_L(t_L), s_R(t_R))$, where t_L is a moment of time at which L holds and t_R is the first moment of time $> e_L(t_L)$ when R becomes true.*
- *For a scope statement "after L until R", in addition to scopes corresponding to "between L and R", we also allow a scope $[e_L(t_L), \infty)$, where t_L is a moment of time at which L holds and for which R does not hold at any moment $t > e_L(t_L)$.*

Definition 12 *Let P and Q be composite propositions, and let s be a scope.*

- *We say that P s-*holds *at a moment $t_P \in s$ if P holds at the moment t_p and the ending moment of time $e_P(t_p)$ belongs to the same scope s.*
- *We say that Q* responds to P within the scope s *if once P s-holds at some moment t, then Q also s-holds at some moment t' for which $s_Q(t') \geq e_P(t)$.*

Definition 13 *We say that a pattern* holds *within a scope statement if it holds within each scope corresponding to this scope statement.*

Now that we have defined what it means for a pattern to hold within the different types of scopes, we are ready to provide the LTL description of the five patterns within the scopes ("before R", "after L", "between L and R", and "after L until R").

7 Response Pattern For Scopes Other Than Global: Case of Composite Propositions

Definition 14 *Let $\mathcal{P}$ be a response pattern, i.e., "Q responds to P", and let $\mathcal{P}_{<R}$ denote this pattern in the scope "before R", i.e., a pattern "Q Responds to P Before R". Then, the corresponding LTL-formula $\mathcal{P}^{LTL}_{<R}$ has the following form:*

- *when R is of the type $T_C(r_1, \ldots, r_n)$, then $\mathcal{P}^{LTL}_{<R}$ is*

$$\neg((\neg R^{LTL})\, U\, ((P^{LTL} \&_r \neg R^{LTL})$$

$$\&_l((\neg(Q^{LTL} \&_r \neg R^{LTL}))\, U\, R^{LTL})));$$

- *when R is of the type $T_E(r_1, \ldots, r_n)$, then $\mathcal{P}^{LTL}_{<R}$ is*

$$\neg((\neg((\neg r_1 \wedge \neg r_2 \wedge \ldots \wedge \neg r_n) \wedge X(R^{LTL}_H)))U$$

$$((P^{LTL} \&_r \neg R^{LTL}_H) \&_l$$

$$((\neg(Q^{LTL} \&_r \neg R^{LTL}_H))U R^{LTL}_H))).$$

For example, the formula $\mathcal{P}^{LTL}_2$ for Q responds to $P\ Before\ R$ where R is $AtLeastOne_C(r_1, r_2)$, P is $Consecutive_C(p1, p2)$, and Q is $Parallel_C(q_1, q_2)$ is:

$$\neg((\neg(r_1 \wedge X r_2))\, U\, (((p_1 \wedge (\neg(r_1 \vee r_2)) \wedge X((p_2 \wedge \neg(r_1 \vee r_2))$$

$$\wedge(((\neg((q_1 \wedge q_2 \wedge \neg(r_1 \vee r_2))))\, U\, (r_1 \vee r_2))))))))$$

Pattern formulas for the scopes "After L", "Between L and R", and "After L until R" can be generated using the formula $\mathcal{P}^{LTL}$ for the Global scope and the formula $\mathcal{P}^{LTL}_{<R}$ for the scope "before R":

Definition 15 *For a pattern $\mathcal{P}$ in the "After L" scope, the corresponding LTL formula is:*

$$\neg((\neg L^{LTL})\, U\, (L^{LTL} \&_l \neg \mathcal{P}^{LTL})).$$

For example the formula for Q responds to P, $After$ L, where P is $Consecutive_C$ (p_1, p_2), Q is $Parallel_E$ (q_1, q_2), and L is $Eventual_C$ (l_1, l_2) is:

$$\neg((\neg(l_1 \wedge X(\neg l_2\, U\, l_2)))$$

$$U\, (l_1 \wedge X(\neg l_2\, U\, (l_2 \wedge \neg(\mathcal{P}^{LTL}_1)))))$$

where $\mathcal{P}^{LTL}_1$ is the formula defined in Definition 9 in Section 5.

Definition 16 *For a pattern $\mathcal{P}$ in the "Between L And R" scope, the corresponding LTL formula is as follows:*

- *$\Box((L^{LTL} \&_l \neg R^{LTL}) \rightarrow (L^{LTL} \&_l \mathcal{P}^{LTL}_{<R})$ if R is of type C;*
- *$\Box(L^{LTL} \rightarrow (L^{LTL} \&_l \mathcal{P}^{LTL}_{<R})))$ if R is of type E.*

For example, if L is of type $Parallel_C$ (l_1, l_2), P is of type $Consecutive_C$ (p_1, p_2), Q is of type $Parallel_C$ (q_1, q_2), and R is of type $AtLeastOne_C$ (r_1, r_2), then the formula $\mathcal{P}_3$ for Q responds to P, $Between\ L\ And\ R$ is:

$$\Box((l_1 \wedge l_2 \wedge \neg(r_1 \vee r_2)) \rightarrow ((l_1 \wedge l_2 \wedge (\mathcal{P}^{LTL}_2))))$$

Definition 17 *For a pattern $\mathcal{P}$ in the "After L until R" scope, the corresponding LTL formula is:*

- *if R is of type C, then*

$$\Box((L^{LTL} \&_l \neg R^{LTL}) \rightarrow$$

$$(L^{LTL} \&_l ((\mathcal{P}^{LTL}_{<R} \wedge ((\neg \diamond R^{LTL}) \rightarrow \mathcal{P}^{LTL})))));$$

- *if R is of type E, then*

$$\Box(L^{LTL} \rightarrow$$

$$(L^{LTL} \&_l ((\mathcal{P}^{LTL}_{<R} \wedge ((\neg \diamond R^{LTL}) \rightarrow \mathcal{P}^{LTL}))))).$$

the LTL formula for Q responds to P $After$ L $Until$ R where P is $Consecutive_C$ (p_1, p_2), Q is $Parallel_C$ (q_1, q_2), L is $Parallel_C$ (l_1, l_2), and R is $AtLeastOne_C$ (r_1, r_2) is as follows:

$$\Box((l_1 \wedge l_2 \wedge \neg(r_1 \vee r_2)) \rightarrow$$

$$((l_1 \wedge l_2) \wedge ((\mathcal{P}_2^{LTL}) \wedge ((\neg \diamond (r_1 \vee r_2)) \rightarrow (\mathcal{P}_1^{LTL})))))$$

Theorem 3 *For every response formula $\mathcal{P}$ and for every scope s, the formula $\mathcal{P}$ holds within the scope s at the moment t if and only if the corresponding LTL formula holds at this moment t.*

Comment. Similar results hold for all other patterns from Section 1.1.

Acknowledgments. This work is funded in part by the National Science Foundation (NSF) through grant NSF EAR-0225670, by Texas Department of Transportation grant No. 0-5453, and by the Japan Advanced Institute of Science and Technology (JAIST) International Joint Research Grant 2006-08.

References

[1] Audun, J.,"Security Protocol Verification using SPIN", *Proceedings of the First SPIN Workshop*, October 1995.

[2] Cimatti, A., E. Clarke, F. Giunchiglia, and M. Roveri, "NUSMV: a new Symbolic Model Verifier", *International Conference on Computer Aided Verification CAV*, July 1999.

[3] Dwyer, M. B., G. S. Avrunin, and J. C. Corbett, "Patterns in Property Specification for Finite State Verification," *Proceedings of the 21st international conference on Software engineering*, Los Angeles, CA, 1999, 411–420.

[4] http://patterns.projects.cis.ksu.edu/

[5] Hall, A., "Seven Myths of Formal Methods," IEEE Software, September 1990, 11(8)

[6] Holzmann G. J. *The SPIN Model Checker: Primer and Reference Manual,* Addison-Wesley Professional, 2004.

[7] Havelund, K., and T. Pressburger, "Model Checking Java Programs using Java PathFinder", *International Journal on Software Tools for Technology Transfer*, 2(4), April 2000.

[8] Glck, P, R., and G, J. Holzmann, "Using SPIN Model Checking for Flight Software Verification" *Aerospace Conference, IEEE*, March 2002

[9] Mondragon, O., A. Q. Gates, and S. Roach, "Prospec: Support for Elicitation and Formal Specification of Software Properties," in O. Sokolsky and M. Viswanathan (Eds.), *Proceedings of Runtime Verification Workshop, ENTCS*, 89(2), 2004.

[10] Mondragon, O. and A. Q. Gates, "Supporting Elicitation and Specification of Software Properties through Patterns and Composite Propositions," *Intl. Journal Software Engineering and Knowledge Engineering,* 14(1), Feb. 2004.

[11] Manna, Z. and A. Pnueli, "Completing the Temporal Picture," *Theoretical Computer Science,* 83(1), 1991, 97–130.

[12] Rushby, J., "Theorem Proving for Verification," *Modelling and Verification of Parallel Processes,* June 2000.

[13] Salamah, I. S., *Defining LTL formulas for complex pattern-based software properties*, University of Texas at El Paso, Department of Computer Science, PhD Dissertation, July 2007.

[14] Stolz, V. and E. Bodden, "Temporal Assertions using AspectJ", *Fifth Workshop on Runtime Verification,* July 2005.