Events in Property Patterns

Dimitrie O. Păun and Marsha Chechik

Department of Computer Science, University of Toronto,
Toronto, ON M5S 3G4, Canada.
Email: {dimi,chechik}@cs.toronto.edu

Abstract. A pattern-based approach to the presentation, codification and reuse of property specifications for finite-state verification was proposed by Dwyer and his colleagues in [4, 3]. The patterns enable non-experts to read and write formal specifications for realistic systems and facilitate easy conversion of specifications between formalisms, such as LTL, CTL, QRE. In this paper we extend the pattern system with *events* — changes of values of variables in the context of LTL.

1 Introduction

Temporal logics (TL) (e.g., [1], [5], [9], [16], [12]) have received a lot of attention in the research community. Not only are they useful for specifying properties of systems, recent advances in model-checking allow effective automatic checking of models of systems against such properties, e.g. using tools like SPIN [8] and SMV [13].

One important obstacle to using temporal logic is the ability to express complex properties correctly. To remedy this problem, Dwyer and his colleagues have proposed a pattern-based approach to the presentation, codification and reuse of property specifications. The system allows patterns like "P is absent between Q and S" or "S precedes P between Q and R" to be easily expressed in and translated between linear-time temporal logic (LTL) [11], computational tree logic (CTL) [2], quantified regular expressions (QRE) [15], and other state-based and event-based formalisms. Dwyer et. al. also performed a large-scale study in which specifications containing over 500 temporal properties were collected and analyzed. They noticed that over 90% of these could be classified under one of the proposed patterns [4].

In earlier work [20], we attempted to use the pattern-based approach to help us formalize properties of the Production Cell system in LTL. We found that we could not use the pattern-based approach directly, because our properties used *events* — changes of values of variables, e.g., "magnet should become deactivated", which we wanted to formalize as "magnet is active now and will be inactive in the next state". We called such events *edges*.

LTL is a temporal logic comprised of propositional formulas and temporal connectives $\Box$ ("always"), $\Diamond$ ("eventually"), $\circ$ ("next"), and $\mathcal{U}$ ("until"). The first three operators are unary, while the last one is binary. $\mathcal{U}$ is the *strong*

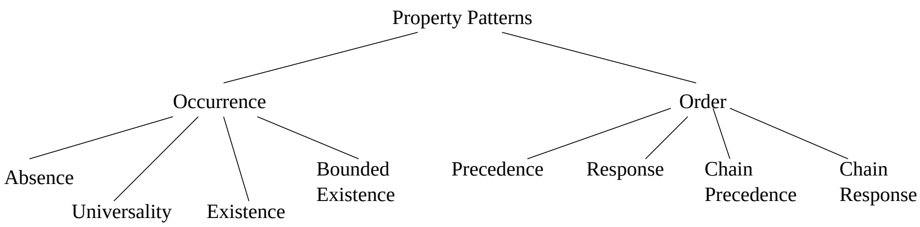

Fig. 1. A Pattern Hierarchy.

until; that is, it requires that B actually happen sometime in the future. In this context, we define edges as follows:

$$\uparrow A = \neg A \wedge \circ A \text{ — up or rising edge}$$
$$\downarrow A = A \wedge \circ\neg A \text{ — down or falling edge}$$
$$\updownarrow A = \uparrow A \vee \downarrow A \text{ — any edge}$$

LTL formulas containing events may have problems caused by the use of the "next" operator in the definition of edges. Temporal formulas that make use of "next" may not be *closed under stuttering*, i.e. their interpretation may be modified by transitions that leave the system in the same state ("stutter"). As we discuss later in the paper, this is an essential property for effective use of temporal formulas.

In this paper we extend the properties of Dwyer et. al. to include events in LTL properties. The rest of the paper is organized as follows: Section 2 overviews the pattern-based system. Section 3 presents our extension to the pattern-based system and discusses the extension process. Section 4 contains an informal summary of our treatment of closure under stuttering and presents a set of theorems that allow syntactic checking of formulas for this property. In addition, it shows how to use these theorems to prove that our extensions of the pattern-based system are closed under stuttering. Section 5 concludes the paper.

2 Overview of the Pattern-Based Approach

In this section we survey the pattern-based approach. For more information, please refer to [3, 4]. The patterns are organized hierarchically based on their semantics, as illustrated in Figure 1. Some of the patterns are described below:

Absence A condition does not occur within a scope;
Existence A condition must occur within a scope;
Universality A condition occurs throughout a scope;
Response A condition must always be followed by another within a scope;
Precedence A condition must always be preceded by another within a scope.

Each pattern is associated with several *scopes* — the regions of interest over which the condition is evaluated. There are five basic kinds of scopes:

A. Global The entire state sequence;
B. Before R The state sequence up to condition R;
C. After Q The state sequence after condition Q;
D. Between Q **and** R The part of the state sequence between condition Q and condition R;
E. After Q **Until** R Similar to the previous one, except that the designated part of the state sequence continues even if the second condition does not occur.

These scopes are depicted in Figure 2. The scopes were initially defined in [4] to be closed-left, open-right intervals, although it is also possible to define other combinations, such as open-left, closed-right intervals.

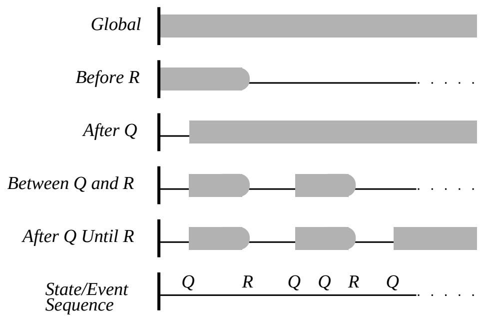

Fig. 2. Pattern Scopes.

For example, an LTL formulation of the property "S precedes P between Q and R" (**Precedence** pattern with "between Q and R" scope) is:

$$\Box((Q \wedge \Diamond R) \Rightarrow (\neg P\, \mathcal{U}\, (S \vee R)))$$

Even though the pattern system is formalism-independent [3], in this paper we are only concerned with the expression of properties in LTL. S

3 Edges and the Pattern-Based System

LTL is a state-based formalism, and thus the original pattern system does not specify the expression of events in LTL. In this section we show how to include reasoning about events to the pattern system. These events can be used for specifying conditions as well as for defining the bounding scopes.

We start by introducing some notation that allows us a more compact representation of properties. We define the *weak* version of "until" as:

$$A\, \mathcal{W}\, B = \Box A \vee (A\, \mathcal{U}\, B)$$

That is, we no longer require B to happen in the future; if it does not, than A should hold indefinitely. Another useful operator is "precedes":

$$A \mathcal{P} B = \neg(\neg A \mathcal{U} B)$$

That is, we want A to hold *before* B does. Note that in this case B may never happen. Also, we use $y \lhd x \rhd z$ to indicate **if** x **then** y **else** z, or $(x \wedge y) \vee (\neg x \wedge z)$. Finally, we write $\top$ and $\bot$ to indicate boolean values *true* and *false*, respectively.

Since our extension involves edges, we give a few relevant properties below:

$$\Box\uparrow A = \bot \tag{1}$$

$$\Box\downarrow A = \bot \tag{2}$$

$$\uparrow A \mathcal{U} B = B \vee (\uparrow A \wedge \circ B) \tag{3}$$

Properties (1) and (2) indicate that edges of the same type cannot occur in every state of the system, whereas property (3) allows us to replace an "until" with a propositional expression.

We have explored the concept of edges in [19, 17], and list some of edge properties in the Appendix.

3.1 Extending the Pattern System

Introducing edges into the patterns generates an explosion in the number of formulas: conditions can be state-based or edge-based, inclusive or exclusive, and the interval ends can be either opened or closed. Our extension does not include all the possibilities, but rather a significant and representative set of them, as discussed below.

We were able to extend five of the nine patterns: **Absence** (Figure 3), **Existence** (Figure 4), **Universality** (Figure 5), **Response** (Figure 7), **Precedence** (Figure 6). For each of the five scopes, we list four formulas corresponding to the four combinations of state-based and edge-based conditions and interval bounds we have considered:

0. P, S — states, Q, R — states;
1. P, S — states, Q, R — up edges;
2. P, S — up edges, Q, R — states;
3. P, S — up edges, Q, R — up edges.

Combination 0 corresponds to the original formulation of [3], where all of P, S, Q and R are state-based. The remaining three combinations are our extensions to the pattern system. We assume that multiple events can happen simultaneously, but only consider closed-left, open-right intervals, as in the original system. Also, we consider events P and S to be exclusive in the **Precedence** pattern and inclusive in the **Response** pattern[1]. We note, however, that it is perfectly

[1] Two events are considered exclusive if they are not allowed to happen at the same time, and inclusive otherwise.

A. Globally	B. Before R	C. After Q
0. $\Box\neg P$	0. $\Diamond R \Rightarrow (\neg P\, \mathcal{U}\, R)$	0. $\Box(Q \Rightarrow \Box\neg P)$
1. $\Box\neg P$	1. $\Diamond{\uparrow}R \Rightarrow ({\uparrow}R\, \mathcal{P}\, P)$	1. $\Box({\uparrow}Q \Rightarrow \circ\Box\neg P)$
2. $\Box\neg{\uparrow}P$	2. $\Diamond R \Rightarrow (\neg{\uparrow}P\, \mathcal{U}\, R)$	2. $\Box(Q \Rightarrow \Box\neg{\uparrow}P)$
3. $\Box\neg{\uparrow}P$	3. $\Diamond{\uparrow}R \Rightarrow (\neg{\uparrow}P\, \mathcal{U}\, {\uparrow}R)$	3. $\Box({\uparrow}Q \Rightarrow \Box\neg{\uparrow}P)$

D. Between Q and R	E. After Q Until R
0. $\Box((Q \wedge \Diamond R) \Rightarrow (\neg P\, \mathcal{U}\, R))$	0. $\Box((Q \wedge \Diamond P) \Rightarrow (\neg P\, \mathcal{U}\, R))$
1. $\Box(({\uparrow}Q \wedge \Diamond{\uparrow}R \wedge \neg{\uparrow}R) \Rightarrow \circ({\uparrow}R\, \mathcal{P}\, P))$	1. $\Box(({\uparrow}Q \wedge \neg{\uparrow}R \wedge \circ\Diamond P) \Rightarrow \circ({\uparrow}R\, \mathcal{P}\, P))$
2. $\Box((Q \wedge \Diamond R) \Rightarrow (\neg{\uparrow}P\, \mathcal{U}\, R))$	2. $\Box(Q \Rightarrow (\neg{\uparrow}P\, \mathcal{W}\, R))$
3. $\Box(({\uparrow}Q \wedge \Diamond{\uparrow}R) \Rightarrow (\neg{\uparrow}P\, \mathcal{U}\, {\uparrow}R))$	3. $\Box({\uparrow}Q \Rightarrow (\neg{\uparrow}P\, \mathcal{W}\, {\uparrow}R))$

Fig. 3. Formulations of the **Absence** Pattern

A. Globally	B. Before R	C. After Q
0. $\Diamond P$	0. $\Diamond R \Rightarrow (P\, \mathcal{P}\, R)$	0. $\Diamond Q \Rightarrow \Diamond(Q \wedge \Diamond P)$
1. $\Diamond P$	1. $\Diamond{\uparrow}R \Rightarrow (\neg{\uparrow}R\, \mathcal{U}\, P)$	1. $\Diamond{\uparrow}Q \Rightarrow \Diamond({\uparrow}Q \wedge \circ\Diamond P)$
2. $\Diamond{\uparrow}P$	2. $\Diamond R \Rightarrow ({\uparrow}P\, \mathcal{P}\, R)$	2. $\Diamond Q \Rightarrow \Diamond(Q \wedge \Diamond{\uparrow}P)$
3. $\Diamond{\uparrow}P$	3. $\Diamond{\uparrow}R \Rightarrow ({\uparrow}P\, \mathcal{P}\, {\uparrow}R)$	3. $\Diamond{\uparrow}Q \Rightarrow \Diamond({\uparrow}Q \wedge \Diamond{\uparrow}P)$

D. Between Q and R	E. After Q Until R
0. $\Box((Q \wedge \Diamond R) \Rightarrow ((P\, \mathcal{P}\, R) \wedge \neg R))$	0. $\Box(Q \Rightarrow (P\, \mathcal{P}\, R) \wedge \neg R \lhd \Diamond R \rhd \Diamond P)$
1. $\Box(({\uparrow}Q \wedge \Diamond{\uparrow}R) \Rightarrow (\circ(\neg{\uparrow}R\, \mathcal{U}\, P) \wedge \neg{\uparrow}R))$	1. $\Box({\uparrow}Q \Rightarrow \circ(\neg{\uparrow}R\, \mathcal{U}\, P) \wedge \neg{\uparrow}R)$
2. $\Box((Q \wedge \Diamond R) \Rightarrow (({\uparrow}P\, \mathcal{P}\, R) \wedge \neg R))$	2. $\Box(Q \Rightarrow ({\uparrow}P\, \mathcal{P}\, R) \wedge \neg R \lhd \Diamond R \rhd \Diamond{\uparrow}P)$
3. $\Box(({\uparrow}Q \wedge \Diamond{\uparrow}R) \Rightarrow (({\uparrow}P\, \mathcal{P}\, {\uparrow}R) \wedge \neg{\uparrow}R))$	3. $\Box({\uparrow}Q \Rightarrow ({\uparrow}P\, \mathcal{P}\, {\uparrow}R) \wedge \neg{\uparrow}R \lhd \Diamond{\uparrow}R \rhd \Diamond{\uparrow}P)$

Fig. 4. Formulations of the **Existence** Pattern

possible to have formulas for all other combinations of interval bounds. Down edges can be substituted for up edges without changing the formulas. We have modified several of the 0-formulas (i.e. state-based conditions and intervals) from their original formulations of [3] to remove assumptions of interleaving and make them consistent with the closed-left, open-right intervals. Note that in the case of the **Universality** pattern, we do not list formulas for edge-based events as edges cannot be universally present (by (1) and (2)).

The four patterns that we did not extend are: **Bounded Existence**, **Precedence Chain**, **Response Chain**, **Constrained Chain**. While we considered the **Bounded Existence** pattern to be too convoluted to be useful in practice and thus not worth the effort of extending, the other three patterns were not extended for reasons that will become apparent in the next Section.

3.2 Discussion

What is involved in adding events to a property? Consider specifying the absence pattern under the "Between Q and R scope" where Q and R are (up) edges. The

A. Globally
0. $\Box P$
1. $\Box P$

B. Before R
0. $\Diamond R \Rightarrow (P \,\mathcal{U}\, R)$
1. $\Diamond{\uparrow}R \Rightarrow ({\uparrow}R \,\mathcal{P}\, \neg P)$

C. After Q
0. $\Box(Q \Rightarrow \Box P)$
1. $\Box({\uparrow}Q \Rightarrow \circ\Box P)$

D. Between Q and R
0. $\Box((Q \wedge \Diamond R) \Rightarrow (P \,\mathcal{U}\, R))$
1. $\Box(({\uparrow}Q \wedge \Diamond{\uparrow}R \wedge \neg{\uparrow}R) \Rightarrow \circ({\uparrow}R \,\mathcal{P}\, \neg P))$

E. After Q Until R
0. $\Box(Q \Rightarrow P \,\mathcal{W}\, R)$
1. $\Box({\uparrow}Q \Rightarrow \circ({\uparrow}R \,\mathcal{P}\, \neg P) \lhd \Diamond{\uparrow}R \rhd \Box P)$

Fig. 5. Formulations of the **Universality** Pattern

A. Globally
0. $\Diamond P \Rightarrow S \,\mathcal{P}\, P$
1. $\Diamond P \Rightarrow S \,\mathcal{P}\, P$
2. $\Diamond{\uparrow}P \Rightarrow {\uparrow}S \,\mathcal{P}\, {\uparrow}P$
3. $\Diamond{\uparrow}P \Rightarrow {\uparrow}S \,\mathcal{P}\, {\uparrow}P$

E. After Q Until R
0. $\Box(Q \Rightarrow (\Diamond P \Rightarrow \neg P \,\mathcal{U}\, ((S \wedge \neg P) \vee R)))$
1. $\Box({\uparrow}Q \Rightarrow \circ(\Diamond P \Rightarrow ((\neg{\uparrow}R \,\mathcal{U}\, P) \Rightarrow (S \,\mathcal{P}\, P))))$
2. $\Box(Q \Rightarrow (\Diamond P \Rightarrow \neg{\uparrow}P \,\mathcal{U}\, (({\uparrow}S \wedge \neg{\uparrow}P) \vee R)))$
3. $\Box({\uparrow}Q \Rightarrow \circ(\Diamond P \Rightarrow (({\uparrow}P \,\mathcal{P}\, {\uparrow}R) \Rightarrow ({\uparrow}S \,\mathcal{P}\, {\uparrow}P))))$

B. Before R
0. $\Diamond R \Rightarrow (\neg P \,\mathcal{U}\, ((S \wedge \neg P) \vee R))$
1. $\Diamond{\uparrow}R \Rightarrow ((\neg{\uparrow}R \,\mathcal{U}\, P) \Rightarrow (S \,\mathcal{P}\, P))$
2. $\Diamond R \Rightarrow \neg{\uparrow}P \,\mathcal{U}\, (({\uparrow}S \wedge \neg{\uparrow}P) \vee R)$
3. $\Diamond{\uparrow}R \Rightarrow (({\uparrow}P \,\mathcal{P}\, {\uparrow}R) \Rightarrow ({\uparrow}S \,\mathcal{P}\, {\uparrow}P))$

C. After Q
0. $\Diamond Q \Rightarrow \Diamond(Q \wedge (\Diamond P \Rightarrow (S \,\mathcal{P}\, P)))$
1. $\Diamond{\uparrow}Q \Rightarrow \Diamond({\uparrow}Q \wedge \circ(\Diamond P \Rightarrow (S \,\mathcal{P}\, P)))$
2. $\Diamond Q \Rightarrow \Diamond(Q \wedge (\Diamond{\uparrow}P \Rightarrow ({\uparrow}S \,\mathcal{P}\, {\uparrow}P)))$
3. $\Diamond{\uparrow}Q \Rightarrow \Diamond({\uparrow}Q \wedge (\Diamond{\uparrow}P \Rightarrow ({\uparrow}S \,\mathcal{P}\, {\uparrow}P)))$

D. Between Q and R
0. $\Box((Q \wedge \Diamond R) \Rightarrow (\neg P \,\mathcal{U}\, ((S \wedge \neg P) \vee R)))$
1. $\Box(({\uparrow}Q \wedge \neg{\uparrow}R \circ \Diamond{\uparrow}R) \Rightarrow \circ((\neg{\uparrow}R \,\mathcal{U}\, P) \Rightarrow (S \,\mathcal{P}\, P)))$
2. $\Box((Q \wedge \Diamond R) \Rightarrow (\neg{\uparrow}P \,\mathcal{U}\, (({\uparrow}S \wedge \neg{\uparrow}P) \vee R)))$
3. $\Box(({\uparrow}Q \wedge \neg{\uparrow}R \circ \Diamond{\uparrow}R) \Rightarrow \circ(({\uparrow}P \,\mathcal{P}\, {\uparrow}R) \Rightarrow ({\uparrow}S \,\mathcal{P}\, {\uparrow}P)))$

Fig. 6. Formulations of the **Precedence** Pattern

original formula is

$$\Box(Q \wedge \Diamond R \Rightarrow \neg P \,\mathcal{U}\, R)$$

This formula does not include P when Q and R occur simultaneously. This behavior is desired since the founding interval is half open and thus becomes empty when the two ends coincide. If we want to transform the condition and the interval bounds into edges, we may be tempted to use the formula:

$$\Box({\uparrow}Q \wedge \Diamond{\uparrow}R \Rightarrow \neg P \,\mathcal{U}\, {\uparrow}R)$$

However, in order to effectively express properties containing edges, we need to realize that an edge is detected just *before* it occurs, as illustrated in Figure 8[2]. That is, ${\uparrow}A$ becomes true in the state where A is false.

[2] Note that we can move the detection of the edge after it occurs if we replace the "next" by the "previous" operator.

A. Globally	B. Before R
0. $\Box(P \Rightarrow \Diamond S)$	0. $\Diamond R \Rightarrow (P \Rightarrow (\neg R\,\mathcal{U}\,S))\,\mathcal{U}\,R$
1. $\Box(P \Rightarrow \Diamond S)$	1. $\Diamond{\uparrow}R \Rightarrow ((P \Rightarrow (\neg{\uparrow}R\,\mathcal{U}\,S)) \wedge \neg{\uparrow}R)\,\mathcal{U}\,({\uparrow}R \wedge (P \Rightarrow Q))$
2. $\Box({\uparrow}P \Rightarrow \Diamond{\uparrow}S)$	2. $\Diamond R \Rightarrow ({\uparrow}P \Rightarrow (\neg R\,\mathcal{U}\,{\uparrow}S))\,\mathcal{U}\,R$
3. $\Box({\uparrow}P \Rightarrow \Diamond{\uparrow}S)$	3. $\Diamond{\uparrow}R \Rightarrow ({\uparrow}P \Rightarrow (\neg{\uparrow}R\,\mathcal{U}\,{\uparrow}S))\,\mathcal{U}\,{\uparrow}R$

C. After Q

0. $\Box(Q \Rightarrow \Box(P \Rightarrow \Diamond S))$
1. $\Box({\uparrow}Q \Rightarrow \circ\Box(P \Rightarrow \Diamond S))$
2. $\Box(Q \Rightarrow \Box({\uparrow}P \Rightarrow \Diamond{\uparrow}S))$
3. $\Box({\uparrow}Q \Rightarrow \Box({\uparrow}P \Rightarrow \Diamond{\uparrow}S))$

D. Between Q and R

0. $\Box((Q \wedge \Diamond R) \Rightarrow ((P \Rightarrow (\neg R\,\mathcal{U}\,S))\,\mathcal{U}\,R))$
1. $\Box(({\uparrow}Q \wedge \Diamond{\uparrow}R \wedge \neg{\uparrow}R) \Rightarrow \circ(((P \Rightarrow (\neg{\uparrow}R\,\mathcal{U}\,S)) \wedge \neg{\uparrow}R)\,\mathcal{U}\,({\uparrow}R \wedge (P \Rightarrow Q))))$
2. $\Box((Q \wedge \Diamond R) \Rightarrow (({\uparrow}P \Rightarrow (\neg R\,\mathcal{U}\,{\uparrow}S))\,\mathcal{U}\,R))$
3. $\Box(({\uparrow}Q \wedge \Diamond{\uparrow}R) \Rightarrow (({\uparrow}P \Rightarrow (\neg{\uparrow}R\,\mathcal{U}\,{\uparrow}S))\,\mathcal{U}\,{\uparrow}R))$

E. After Q Until R

0. $\Box(Q \Rightarrow (P \Rightarrow (\neg R\,\mathcal{U}\,S))\,\mathcal{W}\,R)$
1. $\Box({\uparrow}Q \Rightarrow \circ(((P \Rightarrow (\neg{\uparrow}R\,\mathcal{U}\,S)) \wedge \neg{\uparrow}R)\,\mathcal{W}\,({\uparrow}R \wedge (P \Rightarrow Q))))$
2. $\Box({\uparrow}Q \Rightarrow ({\uparrow}P \Rightarrow (\neg R\,\mathcal{U}\,{\uparrow}S))\,\mathcal{W}\,R)$
3. $\Box({\uparrow}Q \Rightarrow ({\uparrow}P \Rightarrow (\neg{\uparrow}R\,\mathcal{U}\,{\uparrow}S))\,\mathcal{W}\,{\uparrow}R)$

Fig. 7. Formulations of the **Response** Pattern

Thus, our formula has a problem: we start testing P *before* the edge in Q because this is when we detect ${\uparrow}Q$. We need to fix this by testing P *after* the edge in Q:

$$\Box({\uparrow}Q \wedge \Diamond{\uparrow}R \Rightarrow \circ(\neg P\,\mathcal{U}\,{\uparrow}R))$$

We fixed the above problem but introduced another: the new formula does not work correctly when ${\uparrow}Q$ and ${\uparrow}P$ occur simultaneously. This happens because we make sure that ${\uparrow}R$ occurs one state too early. We need to fix the antecedent by making sure that the interval is non-empty:

$$\Box({\uparrow}Q \wedge \neg{\uparrow}R \wedge \Diamond{\uparrow}R \Rightarrow \circ(\neg P\,\mathcal{U}\,{\uparrow}R))$$

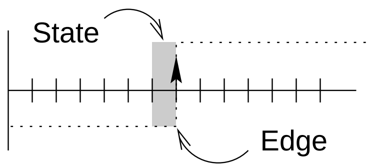

Fig. 8. Edge-detecting state

Unfortunately, the resulting formula is still incorrect: if P and $\uparrow R$ occur simultaneously, then that occurrence of P will be ignored since $\uparrow R$ is detected in the state *before* the edge. This is not the intended behavior as the state before the edge is considered part of the interval. We need to fix it one more time:

$$\Box(\uparrow Q \wedge \neg\uparrow R \wedge \Diamond\uparrow R \Rightarrow \circ\neg(\neg\uparrow R\,\mathcal{U}\,P))$$

or better yet:

$$\Box(\uparrow Q \wedge \neg\uparrow R \wedge \Diamond\uparrow R \Rightarrow \circ(\uparrow R\,\mathcal{P}\,P))$$

4 Closure Under Stuttering

The extension of the pattern system presented in Section 3, is an important development, and we hope that the resulting patterns provide real value to the end user. Still, can practitioners use our extensions directly, without worrying about any unexpected behavior?

The patterns that we have created in the previous section contain the "next" operator. Thus, they may not be *closed under stuttering.* Intuitively, a formula is closed under stuttering when its interpretation is not modified by transitions that leave the system in the same state. For example, a formula $\Box a$ is closed under stuttering because no matter how much we repeat states, we cannot change the value of a. On the other hand, the formula $\circ a$ is not closed under stuttering. We can see that by considering a state sequence in which a is true in the first state and false in the second. Then $\circ a$ is false if we evaluate it on this sequence, and true if we repeat the first state.

Closure under stuttering is an essential property of temporal formulas to ensure basic separation between abstraction levels and to enable powerful partial-order reduction algorithms utilized in mechanized checking, e.g. [8]. This property can be easily guaranteed for a subset of LTL that does not include the "next" operator [10]; however, events cannot be expressed in this subset. Determining whether an LTL formula is closed under stuttering is hard: the problem has been shown to be PSPACE-complete [18]. A computationally feasible algorithm which can identify a subclass of closed under stuttering formulas has been proposed in [7] but have not yet been implemented; without an implementation, one can not say how often the subclass of formulas identified by the algorithm is encountered in practice. Several temporal logics that try to solve the problem have been proposed. Such logics, e.g. TLA [10] and MTL [14], restrict the language so that all formulas expressed in it are, by definition, closed under stuttering. However, it is not clear if these languages are expressive enough for practical use.

In this section we briefly present a set of theorems that allow syntactic reasoning about closure under stuttering in LTL formulas and show how to apply them to our extensions of the pattern system. For a more complete treatment of closure under stuttering, please refer to [17, 19].

4.1 Formal Definition

The notation below is adopted from [6]. A sequence (or string) is a succession of elements joined by semicolons. For example, we write the sequence comprised of the first five natural numbers, in order, as $0; 1; 2; 3; 4$ or, more compactly, as $0; ..5$ (note the left-closed, right-open interval). We can obtain an item of the sequence by subscripting: $(0; 2; 4; 5)_2 = 4$. When the subscript is a sequence, we obtain a subsequence: $(0; 3; 1; 5; 8; 2)_{1;2;3} = (0; 3; 1; 5; 8; 2)_{1;..4} = 3; 1; 5$. A state is modeled by a function that maps variables to their values, so the value of variable a in state s_0 is $s_0(a)$. We denote the set of all infinite sequences of states as $stseq$, and the set of natural numbers as $\mathbb{N}$.

We say that an LTL formula F is closed under stuttering when its interpretation remains the same under state sequences that differ only by repeated states. We denote an interpretation of formula F in state sequence s as $s[\![F]\!]$, and a closed under stuttering formula as $\ll F \gg$. $\ll F \gg$ is formally defined as follows:

Definition 1. $\ll F \gg = \forall s \in stseq \cdot \forall i \in \mathbb{N} \cdot s[\![F]\!] = (s_{0;..i}; s_i; s_{i;..\infty})[\![F]\!]$

In other words, given any state sequence s, we can repeat any of its states without changing the interpretation of F. Note that $s_{0;..i}; s_i; s_{i;..\infty}$ is a sequence of states that differs from s only by repeating a state s_i.

4.2 Properties

Here we present several theorems that allow syntactic reasoning about closure under stuttering. First, we note that $\downarrow$ and $\uparrow$ can be used interchangeably when analyzing properties of the form

$$\ll A \gg \Rightarrow f(\uparrow A)$$

Thus, in what follows we will only discuss the $\uparrow$-edge.

We start with a few generic properties of closure under stuttering:

$$(const(a) \vee var(a)) \Rightarrow \ll a \gg \tag{4}$$
$$\ll A \gg = \ll \neg A \gg \tag{5}$$
$$(\ll A \gg \wedge \ll B \gg) \Rightarrow \ll A \wedge B \gg \tag{6}$$
$$(\ll A \gg \wedge \ll B \gg) \Rightarrow \ll A \vee B \gg \tag{7}$$
$$(\ll A \gg \wedge \ll B \gg) \Rightarrow \ll A \Rightarrow B \gg \tag{8}$$
$$\ll A \gg \Rightarrow \ll \Box A \gg \tag{9}$$
$$\ll A \gg \Rightarrow \ll \Diamond A \gg \tag{10}$$
$$(\ll A \gg \wedge \ll B \gg) \Rightarrow \ll A \,\mathcal{U}\, B \gg \tag{11}$$

For example, (4)-(8) indicate that all propositional formulas are closed under stuttering. The above properties do not include reasoning about formulas that contain the "next" operator. For those, we need the following theorem, proven in [17]:

Theorem 1 (cus-main-thm).

$$\begin{array}{l} \quad (\ll A \gg \wedge \ll B \gg \wedge \ll C \gg \wedge \ll D \gg \wedge \ll E \gg \wedge \ll F \gg) \\ \Rightarrow \\ \quad \ll (\neg \uparrow A \vee \circ B \vee C) \,\mathcal{U}\, (\uparrow D \wedge \circ E \wedge F) \gg \end{array}$$

This theorem establishes an important relationship between the "next" operator, edges, and closure under stuttering. It gives rise to a number of corollaries that we found to be useful in practice.

Property 1

$$(\ll A \gg \wedge \ll B \gg \wedge \ll C \gg) \Rightarrow \ll \Diamond (\uparrow A \wedge \circ B \wedge C) \gg$$

If we take $B = \top$ or $C = \top$ respectively, we obtain two simplified versions:

$$(\ll A \gg \wedge \ll B \gg) \Rightarrow \ll \Diamond (\uparrow A \wedge B) \gg$$

and

$$(\ll A \gg \wedge \ll B \gg) \Rightarrow \ll \Diamond (\uparrow A \wedge \circ B) \gg$$

These formulas represent an *existence* property: an event $\uparrow A$ must happen and then B, evaluated before or after the event, should hold.

Property 2

$$(\ll A \gg \wedge \ll B \gg \wedge \ll C \gg) \Rightarrow \ll \Box (\uparrow A \Rightarrow \circ B \vee C) \gg$$

is similar to Property 1. Its two simplified versions are

$$(\ll A \gg \wedge \ll B \gg) \Rightarrow \ll \Box (\uparrow A \Rightarrow B) \gg$$

and

$$(\ll A \gg \wedge \ll B \gg) \Rightarrow \ll \Box (\uparrow A \Rightarrow \circ B) \gg$$

They express a *universality* property: whenever an event $\uparrow A$ happens, B, evaluated right before or right after the event, will hold.

4.3 Closure Under Stuttering and the Pattern System

All properties of Figures 3- 7 have been shown to be closed under stuttering. This was done using general rules of logic and properties identified above. For example, consider checking a property

$$\Box (\uparrow Q \wedge \neg \uparrow R \wedge \Diamond \uparrow R \Rightarrow \circ \neg (\neg \uparrow R \,\mathcal{U}\, P))$$

for closure under stuttering. The proof goes as follows:

$$
\begin{array}{ll}
& \ll\Box(\uparrow Q \wedge \neg\uparrow R \wedge \circ\Diamond\uparrow R \Rightarrow \circ\neg(\neg\uparrow R\,\mathcal{U}\,P))\gg \\
& \text{by rules of logic and LTL} \\
= & \ll\Box(\uparrow Q \wedge \neg\uparrow R \Rightarrow \circ(\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R))\gg \\
& \text{by definition of } \uparrow \text{ and rules of logic} \\
= & \ll\Box(\uparrow Q \wedge (R \vee \circ\neg R) \Rightarrow \circ(\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R))\gg \\
& \text{again, by rules of logic and LTL} \\
= & \ll\Box((\uparrow Q \wedge R \Rightarrow \circ(\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R))) \wedge \\
& \quad \Box(\uparrow Q \Rightarrow \circ(\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R \vee R))\gg \\
& \text{distribute } \ll\gg \text{ over the main conjunction} \\
\Leftarrow & \ll\Box((\uparrow Q \wedge R \Rightarrow \circ(\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R)))\gg \wedge \\
& \ll\Box(\uparrow Q \Rightarrow \circ(\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R \vee R))\gg \\
& \text{we can use now Property 2 on both conjuncts} \\
\Leftarrow & \ll Q\gg \wedge \ll R\gg \wedge \ll\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R\gg \wedge \\
& \ll Q\gg \wedge \ll\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R \vee R\gg \\
& \text{by rules of logic, (5) and (6)} \\
\Leftarrow & \ll Q\gg \wedge \ll R\gg \wedge \ll\neg\uparrow R\,\mathcal{U}\,P\gg \wedge \ll\Diamond\uparrow R\gg \wedge \\
& \ll Q\gg \wedge \ll\neg\uparrow R\,\mathcal{U}\,P\gg \wedge \ll\Diamond\uparrow R\gg \wedge \ll R\gg \\
& \text{by Theorem 1 and Property 1 we get} \\
\Leftarrow & \ll Q\gg \wedge \ll R\gg \wedge \ll R\gg \wedge \ll P\gg \wedge \ll R\gg
\end{array}
$$

We have thus proved that

$$(\ll P\gg \wedge \ll Q\gg \wedge \ll R\gg) \Rightarrow \ll\Box(\uparrow Q \wedge \neg\uparrow R \wedge \Diamond\uparrow R \Rightarrow \circ\neg(\neg\uparrow R\,\mathcal{U}\,P))\gg$$

Although the property is fairly complicated, the proof is not long, is completely syntactic, and each step in the proof is easy. Such a proof can potentially be performed by a theorem-prover with minimal help from the user.

As we noted earlier, we did not extend all patterns to include events. The reason is that **Precedence Chain**, **Response Chain** and **Constrained Chain** were not closed under stuttering even in their state-based formulations. Consider, for example, the **Response Chain** pattern under the Global scope, formalized as

$$\Box((S \wedge \circ\Diamond T) \Rightarrow \circ\Diamond(T \wedge \Diamond P))$$

When we evaluate this formula on the state sequence s:

	s_0	s_1	s_2	$\cdots$
S	$\top$	$\bot$	$\bot$	$\cdots$
T	$\top$	$\bot$	$\bot$	$\cdots$
P	$\bot$	$\bot$	$\bot$	$\cdots$

we get $\top$ because the antecedent is always $\bot$. However, if we stutter the first state s_0, we get the sequence $s_0; s$:

	s_0	s_0	s_1	s_2	$\cdots$
S	$\top$	$\top$	$\bot$	$\bot$	$\cdots$
T	$\top$	$\top$	$\bot$	$\bot$	$\cdots$
P	$\bot$	$\bot$	$\bot$	$\bot$	$\cdots$

The interpretation of the formula on this sequence is now $\bot$ because the antecedent is $\top$ and the consequent is $\bot$ (since $\Diamond P$ is always $\bot$). As stuttering causes a change in the interpretation of the formula, we can conclude that the formula is not closed under stuttering.

5 Conclusion

In this paper we developed a concept of edges and used it to extend the pattern-based system of Dwyer et. al. to reasoning about events. We have also presented a set of theorems that enable the syntax-based analysis of a large class of formulas for closure under stuttering. This class includes all LTL formulas of the patterns that appeared in this paper. Since research shows that patterns account for 90% of the temporal properties that have been specified so far [4], we believe that our approach is highly applicable to practical problems.

The goal of the pattern-based approach is to enable practitioners to easily codify complex properties into temporal logic. The extensions presented in this paper allow them to express events easily and effectively, without worrying about closure under stuttering. We hope that this work has moved us, as a community, one step closer to making automatic verification more widely usable.

References

1. A. Bernstein and P. K. Harter. Proving real time properties of programs with temporal logic. In *Proceedings of the Eight Symposium on Operating Systems Principles*, pages 1–11. ACM, 1981.
2. E.M. Clarke, E.A. Emerson, and A.P. Sistla. "Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications". *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.
3. Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. "Property Specification Patterns for Finite-state Verification". In *Proceedings of 2nd Workshop on Formal Methods in Software Practice*, March 1998.
4. Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. "Patterns in Property Specifications for Finite-State Verification". In *Proceedings of 21st International Conference on Software Engineering*, May 1999.
5. B. T. Hailpern and S. S. Owicki. Modular verification of computer communication protocols. *IEEE Transactions on Communication*, 1(COM-31):56–68, 1983.
6. Eric C. R. Hehner. *A Practical Theory of Programming*. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1993.
7. G. J. Holzmann and O. Kupferman. "Not Checking for Closure under Stuttering". In *Proceedings of SPIN'96*, 1996.
8. G.J. Holzmann. "The Model Checker SPIN". *IEEE Transactions on Software Engineering*, 23(5):279–295, May 1997.

9. Leslie Lamport. Specifying concurrent program modules. *ACM Transactions of Programming Languages and Systems*, 5:190–222, 1983.
10. Leslie Lamport. "The Temporal Logic of Actions". *ACM Transactions on Programming Languages and Systems*, 16:872–923, May 1994.
11. Z. Manna and A. Pnueli. "Tools and Rules for the Practicing Verifier". Technical Report STAN-CS-90-1321, Department of Computer Science, Stanford University, 1990. Appeared in *Carnegie Mellon Computer Science: A 25 year Commemorative*.
12. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.
13. K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic, 1993.
14. Abdelillah Mokkedem and Dominique Méry. A stuttering closed temporal logic for modular reasoning about concurrent programs. In *Temporal Logic: First International Conference, ICTL '94*, number 827 in Lecture Notes in Artificial Intelligence, Berlin, July 1994. Springer-Verlag.
15. Kurt M. Olender and Leon J. Osterweil. "Cecil: A Sequencing Constraint Language for Automatic Static Analysis Generation". *IEEE Transactions on Software Engineering*, 16(3):268–280, March 1990.
16. J. S. Ostroff. *Temporal Logic of Real-Time Systems*. Advanced Software Development Series. Research Studies Press (John Wiley & Sons), 1990.
17. Dimi Paun. Closure under stuttering in temporal formulas. Master's thesis, Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3G4, CANADA, 1999. April.
18. Doron Peled, Thomas Wilke, and Pierre Wolper. "An Algorithmic Approach for Checking Closure Properties of ω-Regular Languages". In *Proceedings of CONCUR '96: 7th International Conference on Concurrency Theory*, August 1996.
19. Dimitrie O. Păun and Marsha Chechik. "Events in Linear-Time Properties". In *Proceedings of 4th International Conference on Requirements Engineering*, June 1999.
20. Dimitrie O. Păun, Marsha Chechik, and Bernd Biechelle. "Production Cell Revisited". In *Proceedings of SPIN'98*, November 1998.

A Properties of Edges

We list a few representative properties of edges here. Their proofs appear in [17]. [17] also contains a comprehensive study of the concept of edges.

Edges are related:

$$\uparrow\neg A = \downarrow A \tag{12}$$
$$\downarrow\neg A = \uparrow A \tag{13}$$
$$\updownarrow\neg A = \updownarrow A \tag{14}$$

Edges interact with the boolean operators as follows:

$$\uparrow(A \wedge B) = (\uparrow A \wedge \circ B) \vee (\uparrow B \wedge \circ A) \tag{15}$$
$$\uparrow(A \vee B) = (\uparrow A \wedge \neg B) \vee (\uparrow B \wedge \neg A) \tag{16}$$
$$\downarrow(A \wedge B) = (\downarrow A \wedge B) \vee (\downarrow B \wedge A) \tag{17}$$
$$\downarrow(A \vee B) = (\downarrow A \wedge \circ\neg B) \vee (\downarrow B \wedge \neg\circ A) \tag{18}$$

Edges interact with each other:

$$\downarrow\downarrow A = \downarrow A \tag{19}$$
$$\downarrow\uparrow A = \uparrow A \tag{20}$$
$$\uparrow\downarrow A = \circ\downarrow A \tag{21}$$
$$\uparrow\uparrow A = \circ\uparrow A \tag{22}$$

Edges interact with temporal operators as follows:

$$\uparrow\circ A = \circ\uparrow A \tag{23}$$
$$\downarrow\circ A = \circ\downarrow A \tag{24}$$
$$\uparrow\Box A = \uparrow A \wedge \circ\Box A \tag{25}$$
$$\downarrow\Box A = \bot \tag{26}$$
$$\uparrow\Diamond A = \bot \tag{27}$$
$$\downarrow\Diamond A = \downarrow A \wedge \circ\Box\neg A \tag{28}$$
$$\uparrow(A\, \mathcal{U}\, B) = \neg(A \vee B) \wedge \circ(A\, \mathcal{U}\, B) \tag{29}$$
$$\downarrow(A\, \mathcal{U}\, B) = B \wedge \neg\circ(A\, \mathcal{U}\, B) \tag{30}$$