Under consideration for publication in Formal Aspects of Computing

# On Closure Under Stuttering

**Dimitrie O. Păun[1] and Marsha Chechik[1]**

[1]Department of Computer Science

University of Toronto

Toronto, Ontario M5S 3G4, Canada.

**Keywords:** LTL, model-checking, closure under stuttering, event-based systems.

**Abstract.** For over a decade, researchers in formal methods tried to create formalisms that permit natural specification of systems and allow mathematical reasoning about their correctness. The availability of fully-automated reasoning tools enables non-experts to use formal methods effectively — their responsibility reduces to just specifying the model and expressing the desired properties. Thus, it is essential that these properties be represented in a language that is easy to use, sufficiently expressive and succinct.

Linear-time temporal logic [MP89] is a formalism that has been extensively used by researchers for program specification and verification. One of the essential properties of LTL formulas is *closure under stuttering*, i.e. their interpretation is not modified by transitions that leave the system in the same state. This property is important from both practical and theoretical prospectives; all closed under stuttering properties can be expressed in $\text{LTL}_{-X}$ – a subset of LTL without the "next" operator. However, it is often unintuitive and error prone to express properties in this subset of LTL; further, determining whether a given LTL property is closed under stuttering is PSPACE-complete [PWW98].

In this paper we introduce a notion of *edges* of LTL formulas and present a formal theory of closure under stuttering. Edges allow natural modeling of systems with events, and the theory enables syntactic reasoning about closure under stuttering of resulting properties. Finally, we apply the theory to the pattern-based approach to specifying temporal formulas [DAC98, DAC99]. Once a formula is determined to be closed under stuttering, it can be converted into $\text{LTL}_{-X}$ using algorithms in [Ete00, PW97] for the use in model-checking.

*Correspondence and offprint requests to*: Marsha Chechik, University of Toronto Department of Computer Science, Toronto, ON, M5S 2G4, Canada.

## 1. Introduction

Formal specification of systems has been an active area of research for several decades. From finite-state machines to process algebras to logics, researchers try to create formalisms that would permit natural specification of systems and allow mathematical reasoning about their correctness. Most of the research styles developed as the result of this research can be broadly classified in two categories: *requirements specification* and *system specification* [Pnu86].

In the requirements style of specification, the system is specified by a list of properties (or requirements) that it should satisfy. Usually, a designer starts with an informal list of requirements provided by the client which are translated to an appropriate formalism [Pnu92]. The system is then modeled (usually in another language) without formal dependence on the requirements. Once the modeling of the system is complete, the verifier employs a variety of techniques to ensure that the system satisfies the desired properties. For finite-state models, many of these techniques are completely automated. The resulting tools, model-checkers, are based on a variety of approaches: for example, SPIN [Hol97] and SMV [McM93] are based on state-space exploration, Concurrency Workbench [CPS93] on bisimulation, and COSPAN [Kur96] on language containment. As model-checkers mature, the responsibility of the user reduces just to specifying the model and expressing the desired properties. In this context, it is vital that the the properties be represented in a language that is easy to use, sufficiently expressive and succinct, to enable even fairly novice users to use it effectively.

In the system specification style, the designer must describe the behavior of the system in terms of responses to incoming events. The high-level characteristics of the system are thus specified implicitly by the interaction of these low-level properties. The designer is provided with a list of specifications; although similar to those in the requirements style, these specifications are not formalized. An informal refinement process is then used to express them in terms of low-level, event-based properties which are believed to be easier to implement correctly.

Temporal logics (TL) (e.g., [BH81, HO83, Lam83, Ost90, MP92]) have received a lot of attention in the research community as potential languages for both styles of specification. A vital property of TL formulas is *closure under stuttering* [AL91]. Intuitively, a formula is closed under stuttering if its interpretation is not modified by transitions that leave the system in the same state, *stutter*. This property of temporal formulas is essential in both styles of specification: in system specification, it is required for basic separation between abstraction levels, while in requirement specification it enables powerful partial-order reduction algorithms which are necessary for successful mechanized checking [Hol97].

The TL used in the paper is *linear-time TL* (LTL) [MP89]. Closure under stuttering can be easily guaranteed for a subset of LTL without the "next" operator [Lam94]. We refer to this language as $\mathrm{LTL}_{-X}$. Peled and Wilke [PW97] showed that *all* closed under stuttering formulas of LTL can be expressed in $\mathrm{LTL}_{-X}$. They provide a translation, $\tau$, which converts a (closed under stuttering) LTL formula $\varphi$ into an equivalent $\mathrm{LTL}_{-X}$ formula $\tau(\varphi)$, with the worst case upper bound of $|\tau(\varphi)| \leq 2^{O(n \times k)} \times |\varphi|$, where $n$ is the number of distinct propositions in $\varphi$, and $k$ is the "next"-depth of $\varphi$. The result has been improved by Etessami [Ete00], but still remains exponential in the "next"-depth of $\varphi$[1]. Thus, ex-

[1] However, in cases where formulas are represented as circuits sharing identical subformulas

pressing properties in $\mathrm{LTL}_{-X}$ directly can be difficult and error-prone, especially when we want to reason about systems containing events. Further, determining whether a formula expressed in the complete LTL is closed under stuttering is hard: the problem has been shown to be PSPACE-complete [PWW98, Ete00]. A more computationally feasible algorithm which can identify a subclass of closed under stuttering formulas has been proposed in [HK96] but not yet implemented. It is also not clear how often the subclass of formulas identified by the algorithm is encountered in practice.

Even though a complete solution is virtually impractical, we can attempt to categorize a subset that is *useful in practice*. In this paper we develop a formal theory of closure under stuttering; in particular, we state and prove theorems that enable reasoning about closure under stuttering of formulas effectively by "eye-balling" them, or using short *syntactic* proofs. We also introduce the notion of *edges* in LTL formulas and use our framework in several applications. Given results of this paper, users can quickly determine a subclass of closed under stuttering formulas, and, if necessary, use conversion algorithms of [Ete00] to achieve a "next"-free formulation of their property.

The rest of the paper is organized as follows: Section 2 describes the notation used throughout the paper and gives some background on LTL and closure under stuttering. The main contribution of the paper is given in Section 3. This section introduces the notion of edges in LTL formulas, present the algebra of edges, and develops the theory of closure under stuttering. Section 4 describes an application of this work to property patterns identified by Dwyer and his colleagues in [DAC98, DAC99]. We discuss related work within our closure under stuttering framework in Section 5. Section 6 outlines directions for future work and concludes the paper.

## 2. Background

In this section we introduce the notation used throughout the paper and discuss LTL and closure under stuttering. For a more complete treatment of the notation, please refer to [P$\check{9}$9].

### 2.1. Notation

The notation given below is adopted from [Heh93]. We write $\top$ and $\bot$ to indicate boolean values *true* and *false*. Boolean expressions are connected by $\neg$(not), $\wedge$ (and), $\vee$ (or), $\Rightarrow$ (implies), $\Leftarrow$ (is implied by), and $=$ (if and only if). To reduce the clutter of parenthesis, we denote the main connective in a formula by a bigger and bolder symbol, e.g. $\boldsymbol{=}$. We consider the connectives to have decreasing precedence as follows: $=$, $\neg$, $\wedge$, $\vee$, $\Rightarrow$; the connectives $\boldsymbol{=}$, $\boldsymbol{\Rightarrow}$, and $\boldsymbol{\Leftarrow}$ have the lowest precedence. For example, the formula $a \wedge b \vee b \boldsymbol{=} a \Rightarrow b$ should be parsed as: $((a \wedge b) \vee b) = (a \Rightarrow b)$.

In addition, we will be using a number of logic properties and will refer to these by name, e.g.

and size of $\varphi$ is measured in the number of distinct subformulas, the bound is $|\tau(\varphi)| \leq O(|\varphi| + n^2)$ [Ete00].

$$a \wedge \bot = \bot \qquad \text{base}$$

where $a$ is a boolean variable. A list of properties used throughout this paper is given in Appendix A.

A sequence (or string) is a succession of elements joined by semicolons. For example, we write the sequence composed 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: $(0; 3; 1; 5; 8; 2)_{1;2;3} = (0; 3; 1; 5; 8; 2)_{1;..4} = 3; 1; 5$.

In this paper we will mostly concern ourselves with infinite sequences. So, we denote the set of natural numbers (including zero) as $\mathbb{N}$, the set of natural numbers greater than zero as $\mathbb{N}^*$, the set of real numbers as $\mathbb{R}$, and the set of all infinite sequences where each element $w_i \in \mathbb{N}^*$ as $\Omega^*$. 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 $\mathbb{S}$, so $\Omega^* \subseteq \mathbb{S}$. To define a subsequence starting at position $i$, we use notation $s_{\geq i}$, which gives us $s_i; ...; s_\infty$.

In addition, we introduce a *stuttering*[2] operator which repeats, or stutters, elements in a sequence:

$$\omega \in \Omega^* \Rightarrow s^{\mp\omega} = \omega_0 \times s_0; \omega_1 \times s_1; ...; \omega_i \times s_i; ... \qquad \text{stuttering}$$

In the above definition we use notation $n \times s_i$ to indicate that the $i$'th element of $s$ is repeated $n$ times. In other words, $\omega$ in $s^{\mp\omega}$ allows to simultaneously stutter a (potentially infinite) number of elements of $s$, with each $\omega_i$ indicating the number of times each element should be repeated. In the rest of the paper we often combine stuttering and subsequences, resulting in expressions like $s^{\mp\omega}_{\geq 1}$. We assume that superscript has a higher precedence than subscript, so this expression should be interpreted as $(s^{\mp\omega})_{\geq 1}$.

We will also define two additional notations which we will use in proofs later in the paper. Let $\Omega_{\{1,2\}}$ define a set of all infinite sequences where each element $w_i \in \{1, 2\}$. Then,

$$s^{\mp 0} = s_0; s \qquad \text{stuttering 0th step only}$$
$$\omega \in \Omega_{\{1,2\}} \Rightarrow s^{\mp\omega>0} = s_0; \omega_1 \times s_1; ...; \omega_i \times s_i; ... \qquad \text{not stuttering 0th step}$$

Note that in the first definition we stutter 0th state only once, whereas in the second definition we stutter up to an infinite number of states, but repeating each at most once.

## 2.2. LTL

Linear time temporal logic (LTL) is a language for describing and reasoning about sequences of states. These sequences can be interpreted in a variety of ways: the state of the world as it evolves over time, the state of a program as

[2] This definition and Definition 1 in Section 2.3 are equivalent to the explicit definition of [Ete99]. However, we use a somewhat different notation.

it is executing, and so forth. Due to its clean semantics and expressive power, LTL has received a lot of attention from researchers who have tried to apply the language to program specification and verification (e.g. [BH81, Leu95, Pnu92, BP98, HKL+98, Hol99]).

Informally, LTL is comprised of propositional formulas and the temporal connectives $\Box$ (always), $\Diamond$ (eventually), $\circ$ (next), and $\mathcal{U}$ (until). The first three operators are unary, while the last one is binary. Using these operators, one can express properties about the evolution of a system. For example, a formula $p \vee \circ q$ indicates that either $p$ holds in the starting state of the system, or $q$ holds in the following state. A formula $\Box(p \Rightarrow \Diamond q)$ indicates that each occurrence of $p$ is followed, at some point, by an occurrence of $q$. We now define the formal semantics of an LTL formula, based on its syntactic structure. Let $A$ and $B$ be LTL formulas, let $a$ be a variable name, and let $s$ be a sequence of states, i.e. $s \in \mathbb{S}$. We denote an interpretation of formula $F$ in state sequence $s$ as $s[\![F]\!]$, and define it as follows:

$$\begin{aligned}
s[\![a]\!] &= s_0(a) \\
s[\![\neg A]\!] &= \neg s[\![A]\!] \\
s[\![A \wedge B]\!] &= s[\![A]\!] \wedge s[\![B]\!] \\
s[\![\circ A]\!] &= s_{\geq 1}[\![A]\!] \\
s[\![\Box A]\!] &= \forall i \in \mathbb{N} \cdot s_{\geq i}[\![A]\!] \\
s[\![\Diamond A]\!] &= \exists i \in \mathbb{N} \cdot s_{\geq i}[\![A]\!] \\
s[\![A\,\mathcal{U}\,B]\!] &= \exists i \in \mathbb{N} \cdot (s_{\geq i}[\![B]\!] \wedge \forall j \in \{0 ,.., i-1\} \cdot s_{\geq j}[\![A]\!])
\end{aligned}$$

For example, $s[\![\Diamond A]\!]$ means that a formula $A$ must hold at some point $i$ in the sequence $s$. We do not describe other boolean operators as they can be expressed in terms of negation and conjunction.

A number of laws for manipulating LTL formulas have been developed. We will use these laws later in the paper and provide the summary in Appendix B.

## 2.3. Closure Under Stuttering

We say that a temporal formula $F$ is closed under stuttering when its interpretation remains the same under state sequences that differ only by repeated states. We denote a closed under stuttering formula as $\ll F \gg$, and formally define it as follows:

**Definition 1.** $\ll F \gg \;=\; \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s[\![F]\!] = s^{\equiv\omega}[\![F]\!]$

In other words, given any state sequence $s$, we can repeat up to an infinite number of its states without changing the interpretation of $F$. We will use the term *open under stuttering* to refer to formulas that are not closed under stuttering.

Closure under stuttering is important in both requirements and system specification. In system specification, open under stuttering formulas expose too much detail of the internal structure, while in requirements specification the use of such formulas precludes efficient model checking.

Lamport was the first to argue [Lam83] that open under stuttering formulas should not be used in system specification. To that end, he objected against the use of the *next* operator which can open a formula under stuttering. The problem he observed pertains to independent (parallel) composition of programs.

In particular, consider a program $P$ specified by an open under stuttering formula $F$. A run of this program $P$ with idle states may be different than a run without idle states. This situation arises from the fact that users are allowed too much expressiveness in the specification language: they can describe the behaviour of the program based on how and when it is scheduled — a clear violation of abstraction boundaries. In particular, this means that it is not possible to parallely compose $P$ with another program $Q$ without potentially modifying $P$'s behaviour!

In requirements specification it is common to use some form of mechanized checking to verify system conformance to the desired requirement. When the temporal formula describing the requirement is closed under stuttering, various partial reduction algorithms can be used to reduce the state space of the model [Hol97, CGMP99]. This is not just a cosmetic optimization — the savings provided by such techniques can be substantial and often can transform an infeasible task into a tractable problem.

It is easy to see that any LTL formula that does not contain the "$\circ$" operator is closed under stuttering. For example, $\Box a$ 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 the state sequence $s$ in which $s_0(a)$ is true and $s_1(a)$ is false. Then $\circ a$ is false when we evaluate it in $s$, and true when we evaluate it in $s_0; s$. In fact, model-checkers such as SPIN [Hol97] recommend avoiding the next operator all together. Peled and Wilke [PWW98] showed that all closed under stuttering properties can be expressed without the next operator, but the resulting formula is exponentially larger than the original[3]. Thus, we lose in succinctness of expression of closed under stuttering properties and "naturalness" of expression of properties.

## 3. Closure Under Stuttering and Edges

Our aim is to capture a large set of formulas that use the "$\circ$" operator but are still closed under stuttering. Let an *edge* on $a$ be a point where $a$ changes (we formalize this notion in the next subsection), i.e., $a$ is true (false) in the current state and false (true) in the next. Intuitively, stuttering does not add or remove edges. As states get stuttered, the exact location of an edge may change; but stuttering will not introduce new edges (because only existing states get repeated) nor will it remove them. Further, it will not change the relative order of edges. Thus, although edges contain the "next" operator, attaching formulas to edges will allow their closure under stuttering. In the rest of this section we present the concept of edges and develop the above intuition into a number of theorems that enable syntactic-based reasoning about closure under stuttering.

### 3.1. Edges: Definition and Algebra

An *edge* of an arbitrary formula $A$ expresses a *change* in $A$. We denote an up edge by $\uparrow$, a down edge by $\downarrow$, and an up or down edge by $\updownarrow$. Their formal definitions follow.

[3] These results have been improved upon by Etessami in [Ete00]. See Section 5 for discussion.

**Definition 2.** If $A$ is an LTL formula, then

| | |
|---|---|
| $\uparrow A = \neg A \land \circ A$ | up-edge |
| $\downarrow A = A \land \circ\neg A$ | down-edge |
| $\updownarrow A = \uparrow A \lor \downarrow A$ | any-edge |

The edges are also called *events*, as they capture the same notion as the events proposed by the Software Cost Reduction (SCR) researchers [Hen80, HJL96]. However, the SCR events are defined just on propositional formulas, whereas our formalism also enables reasoning about events on LTL formulas. For example, a formula $\uparrow\Box A$ has a well-defined interpretation in our language. We also note here a strong analogy between our (logical) edges and signal edges. Computer engineers have made good use of edges in electrical signals — most circuitry is driven by edges, e.g. the clock. Edges are also widely used in other engineering disciplines, e.g. electrical engineering and telecommunications.

We now look at the algebra of edges and how they interact with each other and LTL operators. We start by listing the representative set of properties and follow it with proofs of some of them. The rest of the proofs are available in [P9̆9].

Edges are related:

| | |
|---|---|
| $\uparrow\neg A = \downarrow A$ | semiduality 1 |
| $\downarrow\neg A = \uparrow A$ | semiduality 2 |
| $\updownarrow\neg A = \updownarrow A$ | semiduality 3 |

Note that up and down edges are not duals: negating an edge does not result in the other:

| | |
|---|---|
| $\neg\uparrow A = A \Leftarrow \circ A$ | no change 1 |
| $\neg\downarrow A = A \Rightarrow \circ A$ | no change 2 |
| $\neg\updownarrow A = A = \circ A$ | no change 3 |

Edges interact with the boolean operators as follows:

| | |
|---|---|
| $\uparrow(A \land B) = (\uparrow A \land \circ B) \lor (\uparrow B \land \circ A)$ | pseudo-distributivity 1 |
| $\uparrow(A \lor B) = (\uparrow A \land \neg B) \lor (\uparrow B \land \neg A)$ | pseudo-distributivity 2 |
| $\downarrow(A \land B) = (\downarrow A \land B) \lor (\downarrow B \land A)$ | pseudo-distributivity 3 |
| $\downarrow(A \lor B) = (\downarrow A \land \circ\neg B) \lor (\downarrow B \land \neg\circ A)$ | pseudo-distributivity 4 |

Edges interact with each other:

| | |
|---|---|
| $\downarrow\downarrow A = \downarrow A$ | idempotency |
| $\downarrow\uparrow A = \uparrow A$ | identity |
| $\uparrow\downarrow A = \circ\downarrow A$ | next-edge 1 |
| $\uparrow\uparrow A = \circ\uparrow A$ | next-edge 2 |

Edges interact with temporal operators as follows:

| | |
|---|---|
| $\uparrow\circ A = \circ\uparrow A$ | commutativity 1 |
| $\downarrow\circ A = \circ\downarrow A$ | commutativity 2 |
| $\uparrow\Box A = \uparrow A \land \circ\Box A$ | pseudo-distributivity 5 |

| | |
|---|---|
| $\downarrow\Box A = \bot$ | pseudo-distributivity 6 |
| $\uparrow\Diamond A = \bot$ | pseudo-distributivity 7 |
| $\downarrow\Diamond A = \downarrow A \wedge \circ\Box\neg A$ | pseudo-distributivity 8 |
| $\uparrow(A\,\mathcal{U}\,B) = \neg(A \vee B) \wedge \circ(A\,\mathcal{U}\,B)$ | pseudo-distributivity 9 |
| $\downarrow(A\,\mathcal{U}\,B) = B \wedge \neg\circ(A\,\mathcal{U}\,B)$ | pseudo-distributivity 10 |
| $\Box\uparrow A = \bot$ | cus-always 1 |
| $\Box\downarrow A = \bot$ | cus-always 2 |

**Proof:**

semiduality 1:

| | | |
|---|---|---|
| | $\uparrow\neg A$ | definition of up-edge |
| $\equiv$ | $\neg\neg A \wedge \circ\neg A$ | double negation |
| $\equiv$ | $A \wedge \circ\neg A$ | duality |
| $\equiv$ | $A \wedge \neg\circ A$ | definition of down-edge |
| $\equiv$ | $\downarrow A$ | QED |

semiduality 2:

| | | |
|---|---|---|
| | $\downarrow\neg A$ | semiduality |
| $\equiv$ | $\uparrow\neg\neg A$ | double negation |
| $\equiv$ | $\downarrow A$ | QED |

semiduality 3:

| | | |
|---|---|---|
| | $\updownarrow\neg A$ | definition of any edge |
| $\equiv$ | $\uparrow\neg A \vee \downarrow\neg A$ | semiduality 1 and 2 |
| $\equiv$ | $\downarrow A \vee \uparrow A$ | definition of any edge |
| $\equiv$ | $\updownarrow A$ | QED |

no change 1:

| | | |
|---|---|---|
| | $\neg\uparrow A$ | definition of up-edge |
| $\equiv$ | $\neg(\neg A \wedge \circ A)$ | distribute $\neg$ |
| $\equiv$ | $A \vee \neg\circ A$ | material implication |
| $\equiv$ | $A \Leftarrow \circ A$ | QED |

pseudodistributivity 1:

| | | |
|---|---|---|
| | $\uparrow(A \wedge B)$ | definition of up-edge |
| $\equiv$ | $\neg(A \wedge B) \wedge \circ(A \wedge B)$ | distribute $\neg$, $\circ$ |
| $\equiv$ | $(\neg A \vee \neg B) \wedge (\circ A \wedge \circ B)$ | distribute $\wedge$ |
| $\equiv$ | $(\neg A \wedge \circ A \wedge \circ B) \vee (\neg B \wedge \circ A \wedge \circ B)$ | definition of up-edge and down-edge |
| $\equiv$ | $(\uparrow A \wedge \circ B) \vee (\uparrow B \wedge \circ A)$ | QED |

idempotency:

| | | |
|---|---|---|
| | $\downarrow\downarrow A$ | definition of down-edge |
| $\equiv$ | $\downarrow(A \wedge \circ\neg A)$ | definition of down-edge |
| $\equiv$ | $(A \wedge \circ\neg A) \wedge \circ\neg(A \wedge \circ\neg A)$ | distribute $\circ\neg$ |
| $\equiv$ | $A \wedge \circ\neg A \wedge (\circ\neg A \vee \circ\circ A)$ | distribute $\wedge$ |
| $\equiv$ | $A \wedge (\circ\neg A \vee (\circ\neg A \wedge \circ\circ A))$ | absorption |
| $\equiv$ | $A \wedge \circ\neg A$ | definition of down-edge |
| $\equiv$ | $\downarrow A$ | QED |

identity:

$$\begin{array}{lll} & \downarrow\uparrow A & \text{semiduality} \\ = & \downarrow\downarrow\neg A & \text{idempotency} \\ = & \downarrow\neg A & \text{semiduality} \\ = & \uparrow A & \text{QED} \end{array}$$

next-edge 1:

$$\begin{array}{lll} & \uparrow\downarrow A & \text{definition of down-edge} \\ = & \uparrow(A \wedge \circ\neg A) & \text{definition of up-edge} \\ = & \neg(A \wedge \circ\neg A) \wedge \circ(A \wedge \circ\neg A) & \text{distribute } \circ \text{ and } \neg \\ = & (\neg A \vee \circ A) \wedge (\circ A \wedge \circ\circ\neg A) & \text{distribute } \wedge \\ = & (\neg A \wedge \circ A \wedge \circ\circ\neg A) \vee (\circ A \wedge \circ A \wedge \circ\circ\neg A) & \text{idempotency} \\ = & (\neg A \wedge \circ A \wedge \circ\circ\neg A) \vee (\circ A \wedge \circ\circ\neg A) & \text{absorption} \\ = & \circ A \wedge \circ\circ\neg A & \text{distributivity} \\ = & \circ(A \wedge \circ\neg A) & \text{definition of down-edge} \\ = & \circ\downarrow A & \text{QED} \end{array}$$

next-edge 2:

$$\begin{array}{lll} & \uparrow\uparrow A & \text{semiduality} \\ = & \uparrow\downarrow\neg A & \text{next-edge} \\ = & \circ\downarrow\neg A & \text{semiduality} \\ = & \circ\uparrow A & \text{QED} \end{array}$$

pseudo-distributivity 5:

$$\begin{array}{lll} & \uparrow\Box A & \text{definition of up-edge} \\ = & \neg\Box A \wedge \circ\Box A & \text{iteration} \\ = & \neg(A \wedge \circ\Box A) \wedge \circ\Box A & \text{distribute } \neg \\ = & (\circ\Box A \Rightarrow \neg A) \wedge \circ\Box A & \text{discharge} \\ = & \neg A \wedge \circ\Box A & \text{iteration, identity} \\ = & \neg A \wedge \circ A \wedge \circ\Box A & \text{definition of up-edge} \\ = & \uparrow A \wedge \circ\Box A & \text{QED} \end{array}$$

pseudo-distributivity 6:

$$\begin{array}{lll} & \downarrow\Box A & \text{definition of down-edge} \\ = & \Box A \wedge \neg\circ\Box A & \text{iteration} \\ = & A \wedge \circ\Box A \wedge \neg\circ\Box A & \text{contradiction} \\ = & A \wedge \bot & \text{base} \\ = & \bot & \text{QED} \end{array}$$

pseudo-distributivity 9:

$$\begin{array}{lll} & \uparrow(A\,\mathcal{U}\,B) & \text{definition of up-edge} \\ = & \neg(A\,\mathcal{U}\,B) \wedge \circ(A\,\mathcal{U}\,B) & \text{iteration} \\ = & \neg(B \vee (A \wedge \circ(A\,\mathcal{U}\,B))) \wedge \circ(A\,\mathcal{U}\,B) & \text{distribute } \neg \\ = & \neg B \wedge (\neg A \vee \neg\circ(A\,\mathcal{U}\,B)) \wedge \circ(A\,\mathcal{U}\,B) & \text{material implication} \\ = & \neg B \wedge (\circ(A\,\mathcal{U}\,B) \Rightarrow \neg A) \wedge \circ(A\,\mathcal{U}\,B) & \text{discharge} \\ = & \neg B \wedge \neg A \wedge \circ(A\,\mathcal{U}\,B) & \text{duality} \\ = & \neg(A \vee B) \wedge \circ(A\,\mathcal{U}\,B) & \text{QED} \end{array}$$

pseudo-distributivity 10:

$$\begin{array}{lll} & \downarrow(A\,\mathcal{U}\,B) & \text{definition of down-edge} \\ = & (A\,\mathcal{U}\,B) \wedge \neg\circ(A\,\mathcal{U}\,B) & \text{iteration} \end{array}$$

| | | |
|---|---|---|
| $\equiv$ | $(B \vee (A \wedge \circ(A\,\mathcal{U}\,B))) \wedge \neg\circ(A\,\mathcal{U}\,B)$ | distribute $\wedge$ |
| $\equiv$ | $(B \wedge \neg\circ(A\,\mathcal{U}\,B)) \vee (A \wedge \circ(A\,\mathcal{U}\,B) \wedge \neg\circ(A\,\mathcal{U}\,B))$ | contradiction |
| $\equiv$ | $(B \wedge \neg\circ(A\,\mathcal{U}\,B)) \vee \bot$ | base |
| $\equiv$ | $B \wedge \neg\circ(A\,\mathcal{U}\,B)$ | QED |

■

## 3.2. Stuttering Properties

Now we are ready to define and prove several closure under stuttering properties. The properties are:

| | |
|---|---|
| $const(a) \vee var(a) \Rightarrow \ll a \gg$ | cus-var |
| $\ll A \gg = \ll \neg A \gg$ | cus-not (semiduality) |
| $\ll A \gg \wedge \ll B \gg \Rightarrow \ll A \wedge B \gg$ | cus-and |
| $\ll A \gg \wedge \ll B \gg \Rightarrow \ll A \vee B \gg$ | cus-or |
| $\ll A \gg \wedge \ll B \gg \Rightarrow \ll A \Rightarrow B \gg$ | cus-imp |
| $\ll A \gg \Rightarrow \ll \Box A \gg$ | cus-always |
| $\ll A \gg \Rightarrow \ll \Diamond A \gg$ | cus-eventually |
| $\ll A \gg \wedge \ll B \gg \Rightarrow \ll A\,\mathcal{U}\,B \gg$ | cus-until |

**Proof:**
We prove these properties from the first principles, while recognizing that they follow from the theorem of Peled and Wilke [PW97]. The interpretation of a variable depends *only* on the first state of the sequence. If $a$ is a variable, then
cus-var:

| | | |
|---|---|---|
| | $\ll a \gg$ | definition of cus |
| $=$ | $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s[\![a]\!] = s^{=\omega}[\![a]\!]$ | interp |
| $=$ | $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s_0[\![a]\!] = (s^{=\omega})_0[\![a]\!]$ | stutter |
| $=$ | $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s_0[\![a]\!] = (\omega_0 \times s_0; \omega_1 \times s_1; ...)_0[\![a]\!]$ | index |
| $=$ | $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s_0(a) = (\omega_0 \times s_0; \omega_1 \times s_1; ...)_0(a)$ | reflectivity |
| $=$ | $\top$ | QED |

cus-not (note that it is an equality):

| | | |
|---|---|---|
| | $\ll A \gg$ | definition of cus |
| $=$ | $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s[\![A]\!] = s^{=\omega}[\![A]\!]$ | reflectivity |
| $=$ | $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot \neg s[\![A]\!] = \neg s^{=\omega}[\![A]\!]$ | distributivity |
| $=$ | $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s[\![\neg A]\!] = s^{=\omega}[\![\neg A]\!]$ | definition of cus |
| $=$ | $\ll \neg A \gg$ | QED |

cus-and:

| | | |
|---|---|---|
| | $\ll A \gg \wedge \ll B \gg$ | definition of cus twice |
| $=$ | $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s[\![A]\!] = s^{=\omega}[\![A]\!] \wedge$ | |
| | $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s[\![B]\!] = s^{=\omega}[\![B]\!]$ | distribution |
| $=$ | $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s[\![A]\!] = s^{=\omega}[\![A]\!] \wedge s[\![B]\!] = s^{=\omega}[\![B]\!]$ | confutation |
| $\Rightarrow$ | $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s[\![A]\!] \wedge s[\![B]\!] = s^{=\omega}[\![A]\!] \wedge s^{=\omega}[\![B]\!]$ | distributivity |
| $=$ | $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s[\![A \wedge B]\!] = s^{=\omega}[\![A \wedge B]\!]$ | definition of cus |
| $=$ | $\ll A \wedge B \gg$ | QED |

cus-until:

$$
\begin{array}{ll}
 & \ll A\,\mathcal{U}\,B \gg \\
 & \text{by the definition of closure under stuttering} \\
= & \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s\llbracket A\,\mathcal{U}\,B \rrbracket = s^{=\omega}\llbracket A\,\mathcal{U}\,B \rrbracket \\
 & \text{by the semantics of until we get} \\
= & \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot \\
 & \quad (\exists k \in \mathbb{N} \cdot s_{\geq k}\llbracket B \rrbracket \wedge \forall j \in \{0 ,.., k-1\} \cdot s_{\geq j}\llbracket A \rrbracket) = \\
 & \quad (\exists k' \in \mathbb{N} \cdot s^{=\omega}_{\geq k'}\llbracket B \rrbracket \wedge \forall j \in \{0 ,.., k'-1\} \cdot s^{=\omega}_{\geq j}\llbracket A \rrbracket) \\
 & \text{we know that } \ll A \gg \text{ and } \ll B \gg \text{, so } \forall s \in \mathbb{S} \cdot s^{=\omega}\llbracket A \rrbracket = s\llbracket A \rrbracket \;\wedge\; s^{=\omega}\llbracket B \rrbracket = s\llbracket B \rrbracket \\
= & \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot \\
 & \quad (\exists k \in \mathbb{N} \cdot s_{\geq k}\llbracket B \rrbracket \wedge \forall j \in \{0 ,.., k-1\} \cdot s_{\geq j}\llbracket A \rrbracket) = \\
 & \quad (\exists k'' \in \mathbb{N} \cdot s_{\geq k''}\llbracket B \rrbracket \wedge \forall j \in \{0 ,.., k''-1\} \cdot s_{\geq j}\llbracket A \rrbracket) \\
 & \text{since the two quantities are identical, by reflectivity we get} \\
= & \top
\end{array}
$$

■

This set of laws justifies the statement that temporal formulas which do not contain the next operator are closed under stuttering. More importantly, they enable reasoning about closure under stuttering by considering the components of a formula, that is, *syntactic* structure of the formula. Finally, semiduality of edges and closure under stuttering allows the interchangeable use of $\downarrow$ and $\uparrow$ when analyzing properties of the form

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

Thus, the rest of this paper will discuss closure under stuttering using only $\uparrow$-edges.

## 3.3. Intuition

We are looking to characterize a relation between the next operator and closure under stuttering for a large set of temporal formulas. We will start by giving some intuition about this notion.

Let $B$ be an LTL formula that is closed under stuttering. Consider the formula $\circ B$: it may no longer be closed under stuttering; furthermore, we can not correct this by simply quantifying $\circ B$ with '$\Box$' or '$\Diamond$'. To understand how stuttering can modify the interpretation of $\circ B$, we illustrate two different stuttering scenarios in Figure 1, where $s$ is a sequence of states, and $s'$ and $s''$ are derived from $s$ by stuttering the fifth and the first states respectively. Note that the type of stuttering exemplified in $s'$, that is, repeating any part of the sequence with the exception of the first state, causes no harm because $B$ itself is closed under stuttering. However, stuttering the first state, as shown in $s''$, may cause problems, because $\circ B$ is evaluated on a state sequence that includes a new state. Or, to generalize, if $\circ B$ is evaluated right before the beginning of a stuttered sequence, its interpretation does not change. However, if $\circ B$ is evaluated in the middle of a stuttered sequence, its interpretation may change.

To circumvent this problem, we want to conjoin $\circ B$ with some other formula that becomes true only in the beginning of a stutter sequence. Further, we want to quantify the resulting expression with $\Diamond$ so that the formula can move up the

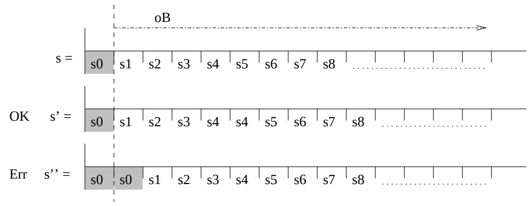


**Fig. 1.** Effects of stuttering on formula $\circ B$.

sequence of states.[4] A possible way to determine when a state is repeated is to pick an arbitrary LTL formula $A$ that is closed under stuttering; then $A$ will not change its interpretation as states get stuttered, whereas $\uparrow A$ may. Either $\uparrow A$ is always false, in which case $\Diamond(\uparrow A \wedge \circ B)$ is also always false and thus closed under stuttering; or $\uparrow A$ becomes true in the beginning of a stuttered sequence and not in the middle. Thus, we effectively removed the "bad" prefix on which $B$ changes its interpretation.

The above was the proof sketch that the formula $\Diamond(\uparrow A \wedge \circ B)$ is closed under stuttering:

**Theorem 1.**

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

## 3.4. Main Theorem

We formalize the intuition given in Section 3.3 in the following theorem. For simplicity in this and the following sections we assume that stuttering repeats each state at most two times. It can be shown by induction that reasoning under this definition generalizes to that under Definition 1.

**Theorem 2.**

$$\begin{array}{l} \quad \forall s \in \mathbb{S} \cdot (s^{=0}[\![X]\!] = \top \wedge s^{=0}[\![Y]\!] = \bot) \wedge \\ \qquad \forall \omega \in \Omega_{\{1,2\}} \cdot (s^{=\omega>0}[\![X]\!] = s[\![X]\!] \wedge s^{=\omega>0}[\![Y]\!] = s[\![Y]\!]) \\ \Rightarrow \\ \quad \ll X \,\mathcal{U}\, Y \gg \end{array}$$

This theorem is concerned with closure under stuttering of the $\mathcal{U}$ operator since other temporal operators (with the exception of next) can be expressed from it. The conditions imposed on $X$ and $Y$ are derived from the assumption that these formulas do not contain nested next operators.

The first conjunct says that when $X$ and $Y$ are interpreted on a sequence that starts with a repeated state, they evaluate to $\top$ and $\bot$, respectively. The second conjunct says that if we stutter any states but $s_0$ in $s$, the interpretation of $X$ and $Y$ on this sequence will not change.

---

[4] Note that we could not have used $\Box$ for quantification as it distributes over conjunction. The choice of a quantifier is dictated by the boolean operator between $\circ B$ and our new formula.

First, we will break the antecedent into four conjuncts, calling them

$$\begin{aligned}\forall s \in \mathbb{S} \cdot s^{=0}[\![X]\!] &= \top && (1)\\ \forall s \in \mathbb{S} \cdot s^{=0}[\![Y]\!] &= \bot && (2)\\ \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot s^{=\omega>0}[\![X]\!] &= s[\![X]\!] && (3)\\ \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot s^{=\omega>0}[\![Y]\!] &= s[\![Y]\!] && (4)\end{aligned}$$

Our goal is thus to prove

$$(1) \wedge (2) \wedge (3) \wedge (4) \Rightarrow \ll X\, \mathcal{U}\, Y \gg$$

**Proof:**

$\ll X\, \mathcal{U}\, Y \gg$

by simplified definition of cus, repeating each state at most twice

$= \quad \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot s^{=\omega}[\![X\, \mathcal{U}\, Y]\!] = s[\![X\, \mathcal{U}\, Y]\!]$

0th state can be stuttered or not

$= \quad \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot (s^{=0}[\![X\, \mathcal{U}\, Y]\!] = s[\![X\, \mathcal{U}\, Y]\!]) \wedge (s^{=\omega>0}[\![X\, \mathcal{U}\, Y]\!] = s[\![X\, \mathcal{U}\, Y]\!])$

by splitting laws

$= \quad (\forall s \in \mathbb{S} \cdot s^{=0}[\![X\, \mathcal{U}\, Y]\!] = s[\![X\, \mathcal{U}\, Y]\!]) \wedge$
$(\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot s^{=\omega>0}[\![X\, \mathcal{U}\, Y]\!] = s[\![X\, \mathcal{U}\, Y]\!])$

To prove this, we will prove the two main conjuncts independently. We begin with the first conjunct:

$\forall s \in \mathbb{S} \cdot s^{=0}[\![X\, \mathcal{U}\, Y]\!] = s[\![X\, \mathcal{U}\, Y]\!]$

by iteration on $\mathcal{U}$

$= \quad \forall s \in \mathbb{S} \cdot s^{=0}[\![Y \vee (X \wedge \circ(X\, \mathcal{U}\, Y))]\!] = s[\![X\, \mathcal{U}\, Y]\!]$

distribute $[\![]\!]$ over the boolean operators

$= \quad \forall s \in \mathbb{S} \cdot s^{=0}[\![Y]\!] \vee (s^{=0}[\![X]\!] \;\wedge\; s^{=0}[\![\circ(X\, \mathcal{U}\, Y)]\!]) = s[\![X\, \mathcal{U}\, Y]\!]$

by (1) and (2) we have $s^{=0}[\![X]\!] = \top$ and $s^{=0}[\![Y]\!] = \bot$

$= \quad \forall s \in \mathbb{S} \cdot \bot \vee (\top \wedge s^{=0}[\![\circ(X\, \mathcal{U}\, Y)]\!]) = s[\![X\, \mathcal{U}\, Y]\!]$

by base laws

$= \quad \forall s \in \mathbb{S} \cdot s^{=0}[\![\circ(X\, \mathcal{U}\, Y)]\!] = s[\![X\, \mathcal{U}\, Y]\!]$

expand the stuttering step

$= \quad \forall s \in \mathbb{S} \cdot s_0; s[\![\circ(X\, \mathcal{U}\, Y)]\!] = s[\![X\, \mathcal{U}\, Y]\!]$

by semantics of next

$= \quad \forall s \in \mathbb{S} \cdot s[\![X\, \mathcal{U}\, Y]\!] = s[\![X\, \mathcal{U}\, Y]\!]$

by reflectivity

$= \quad \top$

Now that we have proven the first conjunct, we proceed with the second:

$\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot s^{=\omega>0}[\![X\, \mathcal{U}\, Y]\!] = s[\![X\, \mathcal{U}\, Y]\!]$

by semantics of $\mathcal{U}$

$= \quad \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot \exists k \in \mathbb{N} \cdot (s_{\geq k}^{=\omega>0}[\![Y]\!] \wedge \forall j \in \{0 ,.., k-1\} \cdot s_{\geq j}^{=\omega>0}[\![X]\!]) = s[\![X\, \mathcal{U}\, Y]\!]$

since (3) and (4) hold, $s^{=\omega>0}[\![X]\!] = s[\![X]\!]$ and $s^{=\omega>0}[\![Y]\!] = s[\![Y]\!]$

$= \quad \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot \exists k' \in \mathbb{N} \cdot (s_{\geq k'}[\![Y]\!] \wedge \forall j' \in \{0 ,.., k'-1\} \cdot s_{\geq j'}[\![X]\!]) = s[\![X\, \mathcal{U}\, Y]\!]$

again, by semantics of $\mathcal{U}$

$= \quad \forall s \in \mathbb{S} \cdot s[\![X\, \mathcal{U}\, Y]\!] = s[\![X\, \mathcal{U}\, Y]\!]$

$$= \quad \begin{array}{l} \text{reflectivity} \\ \top \end{array}$$

This concludes the proof of the theorem, as both conjuncts have been shown to be true. ∎

The following theorem greatly simplifies reasoning about closure under stuttering:

**Theorem 3.**

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

Before proving this theorem, we will prove a number of lemmas.

The first set of requirements of Theorem 2 indicate that the formula should evaluate to $\top$ for $X$ and $\perp$ for $Y$ when interpreted on state sequence where the first state is repeated. When $X$ is closed under stuttering, its interpretation does not change when we remove the repeated state. Thus, if we require a change in such a formula, say $\uparrow X$, we will always obtain $\perp$ when we evaluate this expression on a state sequence which starts with a repeated state. This observation leads to the following lemma:

**Lemma 1.**

$$\ll X \gg \Rightarrow \forall s \in \mathbb{S} \cdot s^{=0}[\![\uparrow X \wedge Y]\!] = \perp$$

**Proof:**

$$\begin{array}{lll} & \forall s \in \mathbb{S} \cdot s^{=0}[\![\uparrow X \wedge Y]\!] = \perp & \text{up-edge} \\ = & \forall s \in \mathbb{S} \cdot s^{=0}[\![\neg X \wedge \circ X \wedge Y]\!] = \perp & \text{distributivity} \\ = & \forall s \in \mathbb{S} \cdot \neg s^{=0}[\![X]\!] \wedge s^{=0}[\![\circ X]\!] \wedge s^{=0}[\![Y]\!] = \perp & \text{by semantics of next} \\ = & \forall s \in \mathbb{S} \cdot \neg s^{=0}[\![X]\!] \wedge s[\![X]\!] \wedge s^{=0}[\![Y]\!] = \perp & s^{=0}[\![X]\!] = s[\![X]\!] \text{ since } \ll X \gg \\ = & \forall s \in \mathbb{S} \cdot \neg s[\![X]\!] \wedge s[\![X]\!] \wedge s^{=0}[\![Y]\!] = \perp & \text{contradiction} \\ = & \forall s \in \mathbb{S} \cdot \perp \wedge s^{=0}[\![Y]\!] = \perp & \text{absorption, reflectivity} \\ = & \top & \text{QED} \end{array}$$

∎

Using the semiduality of closure under stuttering, we directly obtain a corollary:

**Corollary 1.**

$$\ll X \gg \Rightarrow \forall s \in \mathbb{S} \cdot s^{=0}[\![\neg\uparrow X \vee Y]\!] = \top$$

We now consider the second set of requirements — the evaluation of the formula should not change when the state sequence is stuttered anywhere except the first state. We note that this requirement almost amounts to closure under stuttering but is a bit weaker: we do not care if the formula change its interpretation when we stutter the very first state. In effect, we allow a formula which is otherwise closed under stuttering to contain the next operator, but without nesting, as formalized below:

**Lemma 2.**

$$\ll X \gg \wedge \ll Y \gg \Rightarrow \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot s^{=\omega>0}[\![\circ X \vee Y]\!] = s[\![\circ X \vee Y]\!]$$

**Proof:**

$$
\begin{array}{ll}
 & \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot s^{=\omega>0}[\![\circ X \vee Y]\!] = s[\![\circ X \vee Y]\!] \\
 & \text{distribute } [\![]\!] \\
= & \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot s^{=\omega>0}[\![\circ X]\!] \vee s^{=\omega>0}[\![Y]\!] = s[\![\circ X]\!] \vee s[\![Y]\!] \\
 & Y \text{ is closed under stuttering} \\
= & \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot s^{=\omega>0}[\![\circ X]\!] \vee s[\![Y]\!] = s[\![\circ X]\!] \vee s[\![Y]\!] \\
 & \text{by semantics of next} \\
= & \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot s_{\geq 1}^{=\omega>0}[\![X]\!] \vee s[\![Y]\!] = s_{\geq 1}[\![X]\!] \vee s[\![Y]\!] \\
 & \text{1st state law} \\
= & \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot (s_{\geq 1})^{=\omega>0}[\![X]\!] \vee s[\![Y]\!] = s_{\geq 1}[\![X]\!] \vee s[\![Y]\!] \\
 & X \text{ is closed under stuttering} \\
= & \forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot s_{\geq 1}[\![X]\!] \vee s[\![Y]\!] = s_{\geq 1}[\![X]\!] \vee s[\![Y]\!] \\
 & \text{reflectivity} \\
= & \top
\end{array}
$$

■

Putting the last two lemmas and a corollary together, we get another lemma:

**Lemma 3.**

$$
\begin{array}{ll}
 & (\ll A \gg \wedge \ll B \gg \wedge \ll C \gg) \\
\Rightarrow & \\
 & \forall s \in \mathbb{S} \cdot (s^{=0}[\![\neg\uparrow A \vee \circ B \vee C]\!] = \top) \wedge \\
 & \qquad \forall \omega \in \Omega_{\{1,2\}} \cdot (s^{=\omega>0}[\![\neg\uparrow A \vee \circ B \vee C]\!] = s[\![\neg\uparrow A \vee \circ B \vee C]\!])
\end{array}
$$

That is, $\neg\uparrow A \vee \circ B \vee C$ can be plugged in for $X$ in the antecedent of Theorem 2.

**Proof:**

$$
\begin{array}{lll}
 & \neg\uparrow A \vee \circ B \vee C & \text{up-edge} \\
= & \neg(\neg A \wedge \circ A) \vee \circ B \vee C & \text{duality} \\
= & A \vee \circ\neg A \vee \circ B \vee C & \text{distributivity} \\
= & \circ(\neg A \vee B) \vee (A \vee C) &
\end{array}
$$

which is the form required by the last theorem. ■

**Lemma 4.**

$$
\begin{array}{ll}
 & (\ll D \gg \wedge \ll E \gg \wedge \ll F \gg) \\
\Rightarrow & \\
 & \forall s \in \mathbb{S} \cdot (s^{=0}[\![\uparrow D \wedge \circ E \wedge F]\!] = \bot) \wedge \\
 & \qquad \forall \omega \in \Omega_{\{1,2\}} \cdot (s^{=\omega>0}[\![\uparrow D \wedge \circ E \wedge F]\!] = s[\![\uparrow D \wedge \circ E \wedge F]\!])
\end{array}
$$

That is, $\uparrow D \wedge \circ E \wedge F$ can be plugged in for $Y$ in Theorem 2.

**Proof:**
The proof is by semiduality and the above lemma. ■

We are now ready to prove Theorem 3.

**Proof:**
The above two lemmas and Theorem 2 allow us to conclude that

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

which is the statement of the theorem. ■

Theorem 3 will be used during the rest of this paper for proving closure under stuttering.

## 3.5. Some Closure Under Stuttering Properties

We discuss some special cases of the stuttering theorem proven in the previous section, illustrating it with some examples.

The first property we look at is a corollary of Theorem 3:

**Property 1.**

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

**Proof:**
By taking $A = \top$, $B = \top$, and $C = \top$ in Theorem 3, we get

$$\begin{array}{lll} & \ll \top \gg \wedge \ll \top \gg \wedge \ll \top \gg \wedge \ll D \gg \wedge \ll E \gg \wedge \ll F \gg \Rightarrow & \\ & \ll (\neg\uparrow\top \vee \circ\top \vee \top)\,\mathcal{U}\,(\uparrow D \wedge \circ E \wedge F) \gg & \text{base laws} \\ \equiv & \ll D \gg \wedge \ll E \gg \wedge \ll F \gg \Rightarrow \ll \top\,\mathcal{U}\,(\uparrow D \wedge \circ E \wedge F) \gg & \text{eventually} \\ \equiv & \ll D \gg \wedge \ll E \gg \wedge \ll F \gg \Rightarrow \ll \Diamond(\uparrow D \wedge \circ E \wedge F) \gg & \end{array}$$

■

Furthermore, 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

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

that we often encountered in practice. In both cases an *existence* property is formalized. The formulas say that the event $\uparrow A$ must happen and then $B$ holds. In these versions, $B$ is evaluated right before or right after the event, respectively.

By using the semiduality of closure under stuttering and material implication on the previous property, we obtain the following:

**Property 2.**

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

This property 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 the event $\uparrow A$ happens, $B$ will hold. As in the case of Property 1, $B$ is evaluated right before or right after the event, respectively.

Consider the following example:

*Items should only be dropped on the table.*

This property can be formalized as

$$\Box(\downarrow hold \Rightarrow pos = above\_tbl),$$

where *hold* is a state variable that is true when we hold an item, and *pos* is a state variable indicating the position. Note that an item is considered dropped if we hold it in one state and do not hold it in the next. Formally, we express "dropped" as $hold \wedge \circ\neg hold$ or as $\downarrow hold$.

The last property deals with the "until" operator:

**Property 3.**

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

This property is the same as Theorem 3 but we list it here for symmetry. There are many simplified expressions for this property which are omitted here for brevity.

For example, the property

*Initially, no items should be dropped on the table before the operator pushes and releases the GO button.*

can be formalized as

$$\neg\downarrow hold\,\mathcal{U}\downarrow button,$$

where *hold* has the same meaning as before, and *button* is a state variable which is true when the button is pressed and false otherwise.

## 4. Edges and Patterns

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 [DAC99, DAC98]. They 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 [DAC99].

We discuss two directions for integrating our work into the pattern system: extending the system to include events based on edges, and evaluating the effectiveness of our theorems in determining closure under stuttering for the newly created, edge-based formulas. In the rest of the section we briefly discuss the property pattern system (following the presentation in [DAC98]), describe our extensions based on the usage of edges, and conclude with discussing closure under stuttering.

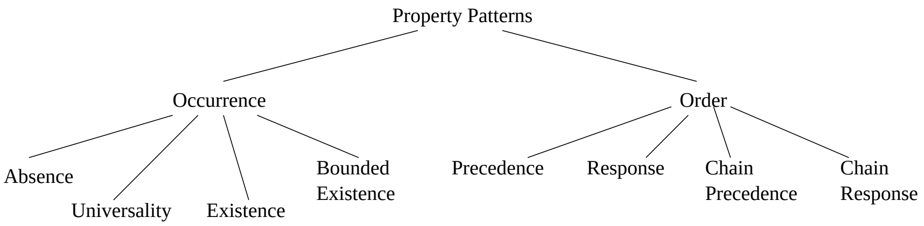


**Fig. 2.** A Pattern Hierarchy.

### 4.1. The Pattern System

The patterns enable non-experts to read and write formal specifications for realistic systems and facilitate easy conversion of specifications between formalisms. Currently, the properties can be expressed in a variety of formalisms such as LTL, computational tree logic (CTL) [CES86], quantified regular expressions (QRE) [OO90], and other state-based and event-based formalisms.

The patterns are organized hierarchically based on their semantics, as illustrated in Figure 2. 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 3. The scopes were initially defined in [DAC99] to be closed-left, open-right intervals, although it is also possible to define other combinations, such as open-left, closed-right intervals.

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 [DAC98], we are only concerned with the expression of properties in LTL.

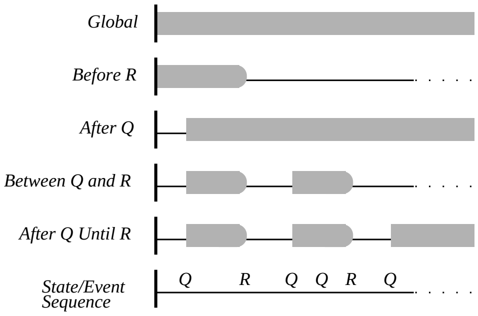


**Fig. 3.** Pattern Scopes.

## 4.2. Events in LTL Patterns

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. For example, we may want to express the properties "Event $S$ precedes event $P$ between states $Q$ and $R$", or "State $S$ precedes state $P$ between events $Q$ and $R$". These properties are often hard to specify correctly, and since all of them contain the "$\circ$" operator (either directly or indirectly through the use of edges), it is not trivial to determine if they are closed under stuttering.

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 $(x \wedge y) \vee (\neg x \wedge z)$.

## 4.3. Extending the Pattern System

Introducing edges into the patterns generates an explosion in the number of formulas. The patterns contain up to two conditions, while the bounding interval has up to two ends; each of these can be either state-based or edge-based, giving us up to 16 different combinations. Moreover, when a condition and an interval end are of the same type (either state-based or edge-based), we can choose to make the interval open or closed, leading to even more possibilities. Note that when the condition and the interval end are of different types, there is no ambiguity because edges occur *between* states.

We were able to extend five of the nine patterns: **Absence** (Figure 4), **Existence** (Figure 5), **Universality** (Figure 6), **Response** (Figure 8), **Precedence** (Figure 7). For each of the five scopes, we list four formulas corresponding to the

| 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 $R$

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

E. After $Q$ Until $R$

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

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

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 [DAC98], 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[5]. We note, however, that it is perfectly 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 [DAC98] 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 cus-always)).

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. The complete set of formulas we developed is available at `http://www.cs.toronto.edu/~chechik/patterns/index.html`.

[5] 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. $\Diamond P$ | 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 $R$

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

E. After $Q$ Until $R$

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

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

| A. Globally | B. Before $R$ | C. After $Q$ |
|---|---|---|
| 0. $\Box P$ | 0. $\Diamond R \Rightarrow (P \, \mathcal{U} \, R)$ | 0. $\Box(Q \Rightarrow \Box P)$ |
| 1. $\Box P$ | 1. $\Diamond{\uparrow}R \Rightarrow ({\uparrow}R \, \mathcal{P} \, \neg P)$ | 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. 6.** Formulations of the **Universality** Pattern

## 4.4. 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 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 9 That is, ${\uparrow}A$ becomes true in the state where $A$ is false.

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. $\Diamond{\uparrow}P \Rightarrow {\uparrow}S\,\mathcal{P}\,{\uparrow}P$

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

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

**Fig. 7.** Formulations of the **Precedence** 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:

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

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:

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

Note that we can avoid many complications of the sort discussed above if we add the "previous" modality, $\ominus$. Using this new operator, we can detect edges just *after* they occur: $a \wedge \neg \ominus a$ (see Figure 9). Although having the "previous" operator can potentially simplify a number of properties, it is currently not supported by SPIN.

| 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. 8.** Formulations of the **Response** Pattern

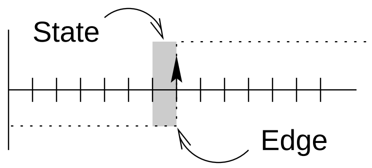


**Fig. 9.** Edge-detecting state

## 4.5. Closure Under Stuttering and Patterns

Our primary concern associated with the edge-based extension of the pattern system is to analyze the newly-created formulas for the closure under stuttering property. Our interest in this problem is twofold: first, we know that edge-based formulas can have practical relevance only if they are closed under stuttering, and second, these formulas provide a good test-bed for the closure under stuttering theorems we have developed.

All properties of Figures 4-8 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.

**Proof:**

$$
\begin{array}{ll}
 & \ll\Box(\uparrow Q \wedge \neg\uparrow R \wedge \circ\Diamond\uparrow R \Rightarrow \circ\neg(\neg\uparrow R\,\mathcal{U}\,P))\gg \\
 & \text{by rules of logic and LTL} \\
= & \ll\Box(\uparrow Q \wedge \neg\uparrow R \Rightarrow \circ(\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R))\gg \\
 & \text{by definition of } \uparrow \text{ and duality of } \circ \\
= & \ll\Box(\uparrow Q \wedge (R \vee \circ\neg R) \Rightarrow \circ(\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R))\gg \\
 & \text{again, by rules of logic and LTL} \\
= & \ll\Box((\uparrow Q \wedge R \Rightarrow \circ(\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R))) \wedge \\
 & \quad \Box(\uparrow Q \Rightarrow \circ(\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R \vee R))\gg \\
 & \text{distribute } \ll\gg \text{ over the main conjunction} \\
\Leftarrow & \ll\Box((\uparrow Q \wedge R \Rightarrow \circ(\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R)))\gg \wedge \\
 & \ll\Box(\uparrow Q \Rightarrow \circ(\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R \vee R))\gg \\
 & \text{we can use now Property 2 on both conjuncts} \\
\Leftarrow & \ll Q\gg \wedge \ll R\gg \wedge \ll\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R\gg \wedge \\
 & \ll Q\gg \wedge \ll\neg(\neg\uparrow R\,\mathcal{U}\,P) \vee \neg\Diamond\uparrow R \vee R\gg \\
 & \text{by semiduality and cus-and} \\
\Leftarrow & \ll Q\gg \wedge \ll R\gg \wedge \ll\neg\uparrow R\,\mathcal{U}\,P\gg \wedge \ll\Diamond\uparrow R\gg \wedge \\
 & \ll Q\gg \wedge \ll\neg\uparrow R\,\mathcal{U}\,P\gg \wedge \ll\Diamond\uparrow R\gg \wedge \ll R\gg \\
 & \text{by Property 3, and Property 1 we get} \\
\Leftarrow & \ll Q\gg \wedge \ll R\gg \wedge \ll R\gg \wedge \ll P\gg \wedge \ll R\gg
\end{array}
$$

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. Related Work

There has been some recent work in checking that an LTL formula is closed under stuttering and in converting LTL formulas to their "next"-free counterparts [Ete99, Ete00, PW97, PWW98], which we discussed in introduction. Somewhat earlier work in this field concentrated on defining languages that would make it natural to express formulas that are closed under stuttering. In this section we show that all of the proposed logics can be analyzed using our formalism.

### 5.1. TLA

TLA (Temporal Logic of Actions) [Lam94] replaces the "next" operator with a more restrictive one, the prime ($'$), applicable only to variables rather than arbitrary formulas, like in our formalism. In addition, all TLA formulas should have the form

$$\Box(A \vee f' = f)$$

where $f$ is a string of all variables in the model. In other words, every formula in TLA says that either $A$ happens or else nothing changes ($f' = f$). Transforming this to our formalism, we get:

$$\begin{array}{lll} & \Box(A \vee \circ(f = f)) & \text{double negation} \\ \equiv & \Box(A \vee \neg(\circ(f \neq f)) & \text{any edge} \\ \equiv & \Box(A \vee \neg\updownarrow f) & \text{material implication} \\ \equiv & \Box(\updownarrow f \Rightarrow A) & \end{array}$$

which is closed under stuttering by Property 2. This proves that all TLA formulas are closed under stuttering.

However, TLA is inappropriate for expressing properties that require reasoning about a value of a formula in the next state.

### 5.2. TLR

Pnueli [Pnu92] suggests that a temporal logic TLR [BKP86] can be used to solve the closure under stuttering problem. This logic uses nonnegative reals as its time domain, and as a result the model for the logic is a continuous function mapping $\mathbb{R}$ to *state* rather than a discrete function (from $\mathbb{N}$ to *state*). In addition, TLR imposes a strong restriction on the way values of variables can change over time, requiring that the interpretation of variables have the *finite variability* requirement. A function has the finite variability property when for every point $r$ in the domain of the function there exist $u < r$ and $v > r$ such that the function is constant on the intervals $(u, r)$ and $(r, v)$. Since such functions have only finitely many points of discontinuity in any interval $[a, b]$, $0 \leq a \leq b$, this restriction on behaviour is intended to ensure that only finitely many transitions causing state changes are allowed in any finite-time interval.

Now we define the semantics of a subset of TLR that is directly comparable to LTL, to help contrast the two logics. First, we need an operator that shifts the domain of a function $f$ by $d$ ($d : \mathbb{R}$):

$$f!d = \lambda x \in \mathbb{R} \cdot f\ (x + d)$$

That is, $f!d$ returns another function which behaves just like $f$ but shifted by $d$. We are now ready to present the semantics of the language:

$$\begin{array}{l}
\sigma[\![x]\!] = x \lhd const(x) \rhd \sigma\ x \\
\sigma[\![f\ x]\!] = f \sigma[\![x]\!] \\
\sigma[\![\neg x]\!] = \neg \sigma[\![x]\!] \\
\sigma[\![x \vee y]\!] = \sigma[\![x]\!] \vee \sigma[\![y]\!] \\
\sigma[\![\circ x]\!] = \exists d \in \mathbb{V} \cdot \exists v \in \mathbb{R}_{\geq 0} \cdot \forall u \in (0, v) \cdot (\sigma!u) = d[\![x]\!] \\
\sigma[\![p\, \widehat{\mathcal{U}}\, q]\!] = \exists v \in \mathbb{R}_{\geq 0} \cdot (\sigma!v)[\![q]\!] \wedge \forall u \in (0, v) \cdot (\sigma!u)[\![p]\!]
\end{array}$$

The $\widehat{\mathcal{U}}$ is the "strict until" operator. We can define the other forward operators in terms of it, where $P$ is a predicate:

$$\begin{array}{l}
P\, \mathcal{U}\, Q = Q \vee (P \wedge (P\, \widehat{\mathcal{U}}\, Q)) \\
\Diamond P = \top\, \mathcal{U}\, P \\
\Box P = \neg \Diamond \neg P o P = P\, \widehat{\mathcal{U}}\, \top
\end{array}$$

Note that the semantics of TLR operators is the same as in LTL, with the exception of $\circ$ and $\widehat{\mathcal{U}}$.

The important feature of the logic that is relevant to our work is the idempotency of the $\circ$ operator:

$$\circ\circ A = \circ A \qquad\qquad \text{idempotency}$$

This feature of the logic makes any TLR formula closed under stuttering [Pnu92].

However, we feel that this sort of semantics is undesirable. The price we pay for having closure under stuttering is too great: the new language is nonintuitive, at least for modeling computer programs where the next operator should definitely not be idempotent!

## 5.3. MTL

Mokkedem and Méry developed a new temporal logic called MTL (Modular Temporal Logic) [MM94] which tries to tackle the closure under stuttering problem by replacing the next operator with a different operator. The new next operator, denoted $\otimes_w$, looks only at the relevant variables in the formula it is applied to; these variables are denoted by the set $w$.

More precisely, the semantics of $\otimes_w$ was originally defined as:

$$\begin{array}{ll}
s^j[\![\otimes_w p]\!] & \text{iff there is some } k > j \text{ s.t. } s_k\lceil w \neq s_j\lceil w \\
 & \quad \text{and } s^k[\![p]\!] \text{ and for every } i, j \leq i < k, s_j\lceil w = s_i\lceil w
\end{array}$$

where:

- $s$ is a sequence of states
- $s^k$ is the sequence $s^k$, $s^{k+1}$, ...

- $s_j$ is state $j$ of sequence $s$
- $w$ is the *index* of formula $p$. Every set of variables $w$ which includes the free variables of $p$ is an index of $p$.
- $s_k \lceil w$ denotes the projection of the state $s_i$ onto the set of variables $w$, that is the restriction of the mappings $s_i$ to the subset of variables in $w$.

The most serious criticism of this operator is that it has complicated semantics: it is not clear what the operator does, how it can be used, and why it is called "next"! Moreover, it is not clear that formulas using it are closed under stuttering. Also, the original presentation of the semantics of this new operator (given above), is rather confusing.

To translate this definition into our formalism, we consider $w$ to be a sequence of variables, rather than a set. With this little adjustment, the above definition can be formalized as:

$$s[\![\otimes_w p]\!] = \exists k \in \mathbb{N} \cdot s_k[\![w]\!] \neq s[\![w]\!] \wedge s_{\geq k}[\![p]\!] \wedge \forall i \in \{0,..,k-1\} \cdot s_i[\![w]\!] = s[\![w]\!]$$

We can transform this as follows:

$$\begin{array}{ll}
= & \exists k \in \mathbb{N} \cdot s_k[\![w]\!] \neq s[\![w]\!] \wedge s_{\geq k}[\![p]\!] \wedge \forall i \in \{0,..,k-1\} \cdot s_i[\![w]\!] = s[\![w]\!] \\
& \text{since } k > 0 \\
= & \exists k \in \mathbb{N} \cdot s_{k-1}[\![w]\!] = s[\![w]\!] \wedge s_k[\![w]\!] \neq s[\![w]\!] \wedge s_{\geq k}[\![p]\!] \wedge \\
& \forall i \in \{0,..,k-1\} \cdot s_i[\![w]\!] = s[\![w]\!] \\
& \text{by semantics of } \circ \\
= & \exists k \in \mathbb{N} \cdot s_{\geq (k-1)}[\![\circ w]\!] \neq s_{\geq (k-1)}[\![w]\!] \wedge s_{\geq (k-1)}[\![\circ p]\!] \wedge \\
& \forall i \in \{0,..,k-1\} \cdot s_i[\![w]\!] = s[\![w]\!] \\
& \text{by distributivity of } [\![]\!] \\
= & \exists k \in \mathbb{N} \cdot s_{\geq (k-1)}[\![\circ w \neq w \wedge \circ p]\!] \wedge \forall i \in \{0,..,k-1\} \cdot s_i[\![w]\!] = s[\![w]\!] \\
& \text{for } i = 0 \text{ the equality is trivially true so we can drop it} \\
= & \exists k \in \mathbb{N} \cdot s_{\geq (k-1)}[\![\circ w \neq w \wedge \circ p]\!] \wedge \forall i \in \{1,..,k-1\} \cdot s_i[\![w]\!] = s[\![w]\!] \\
& \text{by transitivity of equality we have} \\
= & \exists k \in \mathbb{N} \cdot s_{\geq (k-1)}[\![\circ w \neq w \wedge \circ p]\!] \wedge \forall i \in \{1,..,k-1\} \cdot s_i[\![w]\!] = s_{i-1}[\![w]\!] \\
& \text{by semantics of } \circ \text{ and distributivity of } [\![]\!] \\
= & \exists k \in \mathbb{N} \cdot s_{\geq (k-1)}[\![\circ w \neq w \wedge \circ p]\!] \wedge \forall i \in \{1,..,k-1\} \cdot s_{i-1}[\![\circ w = w]\!] \\
& \text{change of index base in existential quantification} \\
= & \exists k \in \mathbb{N} \cdot s_{\geq k}[\![\circ w \neq w \wedge \circ p]\!] \wedge \forall i \in \{1,..,k\} \cdot s_{i-1}[\![\circ w = w]\!] \\
& \text{change of index base in universal quantification} \\
= & \exists k \in \mathbb{N} \cdot s_{\geq k}[\![\circ w \neq w \wedge \circ p]\!] \wedge \forall i \in \{0,..,k-1\} \cdot s_i[\![\circ w = w]\!] \\
& \text{by semantics of } \mathcal{U} \\
= & s[\![(\circ w = w)\, \mathcal{U}\, (\circ w \neq w \wedge \circ p)]\!]
\end{array}$$

That is,

$$\otimes_w p = (\circ w = w)\, \mathcal{U}\, (\circ w \neq w \wedge \circ p)$$

Now, if we interpret $\updownarrow w$ as $\updownarrow w_0 \vee \updownarrow w_1 \vee \cdots$, we can rewrite this as:

$$\otimes_w p = (\neg \updownarrow w)\, \mathcal{U}\, (\updownarrow w \wedge \circ p)$$

which is clearly closed under stuttering by Property 3.

We now have a much more clear definition of the semantics of the new operator and can easily see that the resulting language is closed under stuttering. This logic, like all the previous ones, is still not as expressive as ours.

## 6. Summary and Conclusion

Closure under stuttering of a formula can be determined automatically using a PSPACE-complete algorithm of [PWW98]. However, the algorithm is not practical for sufficiently large formulas. In this paper we developed a formal theory of closure under stuttering and gave theorems that enable syntactic reasoning about it, introduced the notion of *edges* in LTL formulas, and presented some applications of the closure under stuttering theory. Our theory also allowed simple presentation of attempts at defining languages closed under stuttering found in related work.

This paper concentrated on LTL. However, the notions of edges and closure under stuttering studied here are not restricted to LTL and can be applied to other temporal logic, e.g. the ones based on the continuous flow of time. Extending the theory to encompass past operators and other such concepts present in other variants of temporal logics would also be useful.

Our language of edges is not closed, i.e., it is possible to use this language to state a property which is not closed under stuttering. Extensions of the pattern system of [DAC98] with event-based reasoning can be immediately useful to many users of automata-theoretic model-checkers such as SPIN [Hol97]. Still, we would like to enable *automated* checking of a given formula for closure under stuttering. This can be accomplished by incorporating the theorems presented here into a theorem prover and developing effective proof techniques. We hope that this would go a long way towards making temporal logics more usable.

## Acknowledgments

We thank Rick Hehner, Arie Gurfinkel and anonymous referees for their helpful comments on the earlier drafts of this paper. The research was supported in part by the Natural Science and Engineering Research Council of Canada.

## References

[AL91] M. Abadi and L. Lamport. "The Existence of Refinement Mappings". *Theoretical Computer Science*, 82(2):253–284, 1991.

[BH81] 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.

[BKP86] H. Barringer, R. Kuiper, and Amir Pnueli. A really abstract concurrent model and its temporal logic. In *Proceedings of the 13th ACM Symposium on Principles of Programming Languages*, pages 173–183, 1986.

[BP98] Bernd Biechele and Dimitrie O. Păun. "A Case Study Production Cell with PROMELA/SPIN". In M. Chechik, editor, *Automated Verification: A Collection of Reports.* University of Toronto, Technical Report CSRG-374, April 1998.

[CES86] 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.

[CGMP99] Edmund M. Clarke, Orna Grumberg, Marius Minea, and Doron Peled. "State Space Reduction Using Partial Order Techniques". *STTT*, 2(3):279–287, 1999.

[CPS93] Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. "The Concurrency Workbench: A Semantics Based Tool for the Verification of Concurrent Systems". *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, January 1993.

[DAC98] 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.

[DAC99] 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.

[Ete99] K. Etessami. "Stutter-invariant Languages, $\omega$-automata, and Temporal Logic". In *Proceedings of the 11th International Conference on Computer-Aided Verification (CAV'99)*, volume 1633 of *LNCS*, pages 236–249, 1999.

[Ete00] K. Etessami. "A Note of a Question of Peled and Wilke Regarding Stutter-Invariant LTL". *Information Processing Letters*, 75(6):261–263, 2000.

[Heh93] E.C.R. Hehner. *A Practical Theory of Programming.* Texts and Monographs in Computer Science. Springer-Verlag, New York, 1993.

[Hen80] K. Heninger. "Specifying Software Requirements for Complex Systems: New Techniques and Their Applications". *IEEE Transactions on Software Engineering*, SE-6(1):2–12, January 1980.

[HJL96] C. L. Heitmeyer, R. D. Jeffords, and B. G. Labaw. "Automated Consistency Checking of Requirements Specifications". *ACM Transactions on Software Engineering and Methodology*, 5(3):231–261, July 1996.

[HK96] G. J. Holzmann and O. Kupferman. "Not Checking for Closure under Stuttering". In *Proceedings of the Second International SPIN Workshop (SPIN'2)*, pages 17–22, 1996.

[HKL+98] C.L. Heitmeyer, J. Kirby, B. Labaw, M. Archer, and R. Bharadwaj. "Using Abstraction and Model Checking to Detect Safety Violations in Requirements Specifications". *IEEE Transactions on Software Engineering*, 24(11), November 1998.

[HO83] B. T. Hailpern and S. S. Owicki. "Modular Verification of Computer Communication Protocols". *IEEE Transactions on Communication*, 1(COM-31):56–68, 1983.

[Hol97] G.J. Holzmann. "The Model Checker SPIN". *IEEE Transactions on Software Engineering*, 23(5):279–295, May 1997.

[Hol99] Gerard Holzmann. "A Practical Method for Verifying Event-Driven Software". In *Proceedings of the 21st International Conference on Software Engineering (ICSE'99)*, pages 597–607, May 1999.

[Kur96] R. P. Kurshan. "COSPAN". In *Proceedings of CAV'96*, 1996.

[Lam83] Leslie Lamport. Specifying concurrent program modules. *ACM Transactions of Programming Languages and Systems*, 5:190–222, 1983.

[Lam94] Leslie Lamport. "The Temporal Logic of Actions". *ACM Transactions on Programming Languages and Systems*, 16:872–923, May 1994.

[Leu95] S. Leue. "Specifying Real-Time Requirements for SDL Specifications - A Temporal Logic-Based Approach". In *Proceedings of the Fifteenth International Symposium on Protocol Specification, Testing, and Verification PSTV'95*, San Diego, California, 1995.

[McM93] K.L. McMillan. *Symbolic Model Checking.* Kluwer Academic, 1993.

[MM94] 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.

[MP89] Z. Manna and A. Pnueli. "The Anchored Version of the Temporal Framework". In W.P. de Roever J.W. de Bakker and G. Rozenburg, editors, *Models of Concurrency: Linear, Branching and Partial Orders.* Springer-Verlag, 1989.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag, 1992.

[OO90] 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.

[Ost90] J. S. Ostroff. *Temporal Logic of Real-Time Systems.* Advanced Software Development Series. Research Studies Press (John Wiley & Sons), 1990.

[Pnu86] Amir Pnueli. Specification and development of reactive systems. In H. J. Kugler, editor, *Information Processing 86*, pages 845–858. IFIP, North-Holland, 1986.

[Pnu92] Amir Pnueli. System specification and refinement in temporal logic. In Rudrapatna Shyamasundar, editor, *Foundations of Software Technology and Theoretical*

*Computer Science: 12th Conference*, number 652 in Lecture Notes in Computer Science, Berlin, December 1992. Springer-Verlag.

[P̆99] Dimitrie O. Păun. Closure under stuttering in temporal formulas. Master's thesis, Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3G4, CANADA, 1999. April.

[PW97] D. Peled and Th. Wilke. "Stutter-Invariant Temporal Properties are Expressible without the Next-Time Operator". *Information Processing Letters*, 63:243–246, 1997.

[PWW98] D. Peled, Th. Wilke, and P. Wolper. "An Algorithmic Approach for Checking Closure Properties of Temporal Logic Specifications and $\omega$-regular languages". *Theoretical Computer Science*, 195(2):183–203, March 1998.

## A. Axioms

In this section we give a few laws of boolean logic used throughout this paper. Here, assume $a$, $b$, $c$ and $d$ are boolean variables.

| | |
|---|---|
| $a \Rightarrow b = \neg a \vee b$ | material implication |
| $a \wedge \bot = \bot$ | base |
| $a \vee \top = \top$ | base |
| $a \Rightarrow \top$ | base |
| $\bot \Rightarrow a$ | base |
| $a \wedge (a \vee b) = a$ | absorption |
| $a \vee (a \wedge b) = a$ | absorption |
| $(a \wedge b) \Rightarrow c = a \Rightarrow (b \Rightarrow c)$ | partition |
| $(a \wedge b) \Rightarrow c = a \Rightarrow (\neg b \vee c)$ | partition |
| $\neg\neg a = a$ | double negation |
| $a \wedge \neg a = \bot$ | contradiction |
| $a \wedge (a \Rightarrow b) = a \wedge b$ | discharge |
| $a \Rightarrow (a \wedge b) = a \Rightarrow b$ | discharge |
| $\neg(a \wedge b) = \neg a \vee \neg b$ | duality |
| $\neg(a \vee b) = \neg a \wedge \neg b$ | duality |
| $\top \wedge a = a$ | identity |
| $\bot \vee a = a$ | identity |
| $\top \Rightarrow a = a$ | identity |
| $(a \Rightarrow b) \wedge (c \Rightarrow d) \Rrightarrow a \wedge c \Rightarrow b \wedge d$ | confutation |
| $(a \Rightarrow b) \wedge (c \Rightarrow d) \Rrightarrow a \vee c \Rightarrow b \vee d$ | confutation |
| $a \Rightarrow a$ | reflectivity |
| $a = a = \top$ | reflectivity |

Distributive Laws:

$a \wedge (b \wedge c) = (a \wedge b) \wedge (a \wedge c)$
$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$
$a \vee (b \vee c) = (a \vee b) \vee (a \vee c)$
$a \vee (b \Rightarrow c) = (a \vee b) \Rightarrow (a \vee c)$
$a \vee (b = c) = (a \vee b) = (a \vee c)$

$a \Rightarrow (b \wedge c) = (a \Rightarrow b) \wedge (a \Rightarrow c)$
$a \Rightarrow (b \vee c) = (a \Rightarrow b) \vee (a \Rightarrow c)$
$a \Rightarrow (b \Rightarrow c) = (a \Rightarrow b) \Rightarrow (a \Rightarrow c)$
$a \Rightarrow (b = c) = (a \Rightarrow b) = (a \Rightarrow c)$

## B. Laws of Temporal Logic

In this section we give a number of laws pertaining to temporal operators. Since most of these laws are common in the literature, we give them without proofs.

Absorption Laws:

| | |
|---|---|
| $\circ\top = \top$ | $\circ\bot = \bot$ |
| $\Box\top = \top$ | $\Box\bot = \bot$ |
| $\Diamond\top = \top$ | $\Diamond\bot = \bot$ |
| $A\,\mathcal{U}\,\top = \top$ | $A\,\mathcal{U}\,\bot = \bot$ |

Duality laws:

| | |
|---|---|
| $\neg\Box A = \Diamond\neg A$ | $\neg\circ A = \circ\neg A$ |
| $\neg\Diamond A = \Box\neg A$ | |

Distributive laws:

| | |
|---|---|
| $\circ\Box A = \Box\circ A$ | $(A \wedge B)\,\mathcal{U}\,C = (A\,\mathcal{U}\,C) \wedge (B\,\mathcal{U}\,C)$ |
| $\circ\Diamond A = \Diamond\circ A$ | $A\,\mathcal{U}\,(B \vee C) = (A\,\mathcal{U}\,B) \vee (A\,\mathcal{U}\,C)$ |
| $\circ(A\,\mathcal{U}\,B) = \circ A\,\mathcal{U}\,\circ B$ | $\Box\Diamond(A \vee B) = \Box\Diamond A \vee \Box\Diamond B$ |
| $\circ(A \Rightarrow B) = \circ A \Rightarrow \circ B$ | $\Diamond\Box(A \wedge B) = \Diamond\Box A \wedge \Diamond\Box B$ |
| $\circ(A \wedge B) = \circ A \wedge \circ B$ | $\circ\forall x \in D \cdot A = \forall x \in D \cdot \circ A$ |
| $\circ(A \vee B) = \circ A \vee \circ B$ | $\circ\exists x \in D \cdot A = \exists x \in D \cdot \circ A$ |
| $\Box(A \wedge B) = \Box A \wedge \Box B$ | $\Box\forall x \in D \cdot A = \forall x \in D \cdot \Box A$ |
| $\Diamond(A \vee B) = \Diamond A \vee \Diamond B$ | $\Diamond\exists x \in D \cdot A = \exists x \in D \cdot \Diamond A$ |

Idempotent laws:

| | |
|---|---|
| $\Box\Box A = \Box A$ | $A\,\mathcal{U}\,(A\,\mathcal{U}\,B) = A\,\mathcal{U}\,B$ |
| $\Diamond\Diamond A = \Diamond A$ | $(A\,\mathcal{U}\,B)\,\mathcal{U}\,B = A\,\mathcal{U}\,B$ |
| $A\,\mathcal{U}\,A = A$ | |

Iteration laws:

| | |
|---|---|
| $\Box A = A \wedge \circ\Box A$ | $\Diamond A = A \vee \circ\Diamond A$ |
| | $A\,\mathcal{U}\,B = B \vee (A \wedge \circ(A\,\mathcal{U}\,B))$ |

Semidistributive laws:

| | |
|---|---|
| $\Box(A \vee B) \Leftarrow \Box A \vee \Box B$ | $\Box(A \Rightarrow B) \Rightarrow \Box A \Rightarrow \Box B$ |
| $\Diamond(A \wedge B) \Rightarrow \Diamond A \wedge \Diamond B$ | $\Diamond(A \Rightarrow B) \Leftarrow \Diamond A \Rightarrow \Diamond B$ |

Nullpotency laws:

| | |
|---|---|
| $\Diamond\Box\Diamond A = \Box\Diamond A$ | $\circ\Box\Diamond A = \Box\Diamond A$ |
| $\Box\Diamond\Box A = \Diamond\Box A$ | $\circ\Diamond\Box A = \Diamond\Box A$ |

Other laws:

| | |
|---|---|
| $\Box A \Rightarrow A$ | specialization |
| $\Diamond A \Leftarrow A$ | generalization |
| $\Box A \Rightarrow \circ A$ | specialization |
| $\Diamond A \Leftarrow \circ A$ | generalization |

| | |
|---|---|
| $\Box A \Rrightarrow \Diamond A$ | specialization |
| $\Diamond\Box A \Rrightarrow \Box\Diamond A$ | semicommutativity |
| $\top\, \mathcal{U}\, A \equiv \Diamond A$ | eventually |
| $\bot\, \mathcal{U}\, A \equiv A$ | identity |
| $\Diamond A \Rrightarrow \neg A\, \mathcal{U}\, A$ | |

## C. Other Laws

In this section we list laws of sequences and quantifiers used throughout the paper. Here we assume that $s$ is a sequence, $A$ is a boolean expression, and $a$ is a variable or a constant. $''a''$ indicates the string with one value $''a''$.

| | |
|---|---|
| $\ll F \gg \equiv$ <br> $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s[\![F]\!] = s^{\mp\omega}[\![F]\!]$ | definition of cus |
| $\omega \in \Omega^* \Rrightarrow$ <br> $s^{\mp\omega} = \omega_0 \times s_0; \omega_1 \times s_1; ...; \omega_i \times s_i; ...$ | stuttering |
| $s^{\mp 0} = s_0; s$ | stuttering 0th step only |
| $\omega \in \Omega_{\{1,2\}} \Rrightarrow s^{\mp\omega>0} = s_0; \omega_1 \times s_1; ...; \omega_i \times s_i; ...$ | not stuttering 0th step |
| $(\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s[\![a]\!] = s^{\mp\omega}[\![a]\!]) \equiv$ <br> $(\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega^* \cdot s_0(a) = s_0^{\mp\omega}(a))$ | interp |
| $\forall s \in \mathbb{S} \cdot \forall \omega \in \Omega_{\{1,2\}} \cdot s_{\geq 1}^{\mp\omega>0}[\![X]\!] = (s_{\geq 1})^{\mp\omega>0}[\![X]\!]$ | 1st state law |
| $(i > j) \Rrightarrow s_j = (s_{0;..i})_j$ | index |

Splitting laws – for any fixed domain

$\forall \nu \cdot x \wedge y \equiv (\forall \nu \cdot x) \wedge (\forall \nu \cdot y)$
$\exists \nu \cdot x \wedge y \Rrightarrow (\exists \nu \cdot x) \wedge (\exists \nu \cdot y)$
$\forall \nu \cdot x \vee y \Lleftarrow (\forall \nu \cdot x) \vee (\forall \nu \cdot y)$
$\exists \nu \cdot x \vee y \equiv (\exists \nu \cdot x) \vee (\exists \nu \cdot y)$
$\forall \nu \cdot x \Rightarrow y \Rrightarrow (\forall \nu \cdot x) \Rightarrow (\forall \nu \cdot y)$
$\forall \nu \cdot x \Rightarrow y \Rrightarrow (\exists \nu \cdot x) \Rightarrow (\exists \nu \cdot y)$
$\forall \nu \cdot x = y \Rrightarrow (\forall \nu \cdot x) = (\forall \nu \cdot y)$
$\forall \nu \cdot x = y \Rrightarrow (\exists \nu \cdot x) = (\exists \nu \cdot y)$