The Infinite, in Finite Time

Rayhana Amjad[a], Rob van Glabbeek[*,a,c], Liam O'Connor[b,a]

[a] *University of Edinburgh, Edinburgh, United Kingdom*
[b] *Australian National University, Canberra, Australia*
[c] *University of New South Wales, Sydney, Australia*

Abstract

Linear-time temporal properties, such as those described by Linear-time Temporal Logic, are typically modelled as sets of infinite traces. Yet, in a run-time verification context, such as when testing or monitoring a system, only a finite prefix of the system's behaviour can be observed. For some properties, these finite prefixes may be *definitive*—a yes or no answer can be given without further observation. By enriching the semantics of LTL with these definitive prefixes, we give a proper inductive accounting of the semantics of LTL_3, a multi-valued variant of Linear-time Temporal Logic for run-time verification applications. The semantic descriptions of LTL_3 in previous work are given only in terms of their relationship to conventional LTL. We show that the semantics of LTL and of LTL_3 are isomorphic. In addition, we formalise the formula progression evaluation technique, popularly used in runtime verification contexts, and show its soundness and completeness up to finite traces with respect to our semantics. Then, we turn to linear-time properties more generally: using our theory of definitive prefixes, we re-prove the well-known safety-liveness decomposition theorem, and reconstruct the topology of infinite traces. We define *monitorability* for properties, providing neat topological characterisations for various monitorability classes, and arrange them into a hierarchy. All of our definitions and proofs are mechanised in Isabelle/HOL.

Keywords: Linear-time Temporal Logic, Partial Traces, Semantics, Formula Progression, Safety Properties, Liveness Properties, Guarantee Properties, Topology, Monitorability

[*] *Supported by Royal Society Wolfson Fellowship RSWF\R1\221008*

1. Introduction

In the specification, verification and modelling of reactive and concurrent systems, linear-time properties are a recurring theme. Linear-time properties are sets of *behaviours*: completed, infinite *traces*—sequences of states—describing the execution of a system over time. Logics for the specification of such properties, such as Linear-time Temporal Logic (LTL) [1] are well-studied and widely used, and a variety of algorithmic techniques for model checking and verification exist [2]. In the context of *run-time* verification, testing, or monitoring, however, semantics in terms of behaviours appears insufficient, as the full infinite behaviour of the system cannot be observed: we can only observe a finite sequence of states.

In light of this, we might consider switching to a semantics which includes finite traces as models of temporal properties. There are several variants of LTL which include finite traces as models, the oldest of which, called LTLf, is commonly attributed to Pnueli[1]. These logics concern finite or infinite *completed* traces, but this is also not suitable in the context of run-time verification, as our finite observations are not completed traces, but finite *prefixes* of infinite behaviours. In other words, the behaviour of the system is still infinite, it is only our *observation* of the behaviour that is finite.

Therefore, in this paper we advocate remaining with infinite trace semantics even in a run-time verification context. If we are monitoring for the property "eventually, a good state is reached", then our observation can stop the moment a good state is reached, as the property has been satisfied already and no further observation is needed. Similarly, if we a monitoring for the property "the bad state is never reached", our observation can stop as soon as the bad state is reached, as the property has been irrevocably violated. Thus, there are properties for which *definitive* answers—satisfaction or violation—can be determined after observation of just a finite prefix. We call such prefixes *definitive prefixes*, and sets of infinite traces and their definitive prefixes *definitive sets*. We introduce our theory of definitive prefixes and sets in Section 2, including a proof that definitive sets are determined uniquely by their infinite traces, and thus that definitive sets and linear-time properties are isomorphic.

[1]Such logics are found in many early papers on LTL with Pnueli as a coauthor such as Lichtenstein et al. [3], but Manna & Pnueli [4], which is usually cited, does not mention finite traces at all.

Bauer et al. [5] describe a multi-valued variant of LTL for partial traces called LTL_3 that distinguishes between those formulae that can be definitively said to be true or false from just the partial trace provided, and those formulae which are indeterminate, requiring further states to evaluate definitively. As we shall see in Section 3, the semantics of LTL_3 in the literature are given only in terms of conventional LTL, and Bauer et al. [6] further claim that LTL_3 *cannot* be given an inductive semantics, but this claim applies only to the intensional kind of semantics typically given for LTL. In Section 4 we give a compositional, inductive semantics for LTL_3, in terms of families of definitive sets. By virtue of our isomorphism, we further show that the semantics of conventional LTL and of LTL_3 correspond directly. This correspondence is so direct that LTL_3 can be viewed merely as a different presentation of conventional LTL: They share the same syntax, and their meanings are related by a simple transport.

In Section 5 we turn to *formula progression*, a popular technique for evaluating formulae against a finite trace where the formula is evaluated state-by-state, in a style reminiscent of operational semantics or the Brzozowski derivative. Bauer & Falcone [7] claim without proof that formula progression yields an equivalent semantics to LTL_3. In this paper, we make this statement formally precise, and prove soundness and completeness (modulo a sufficiently powerful simplifier) of the formula progression technique with respect to our semantics.

In Section 6, we investigate various classes of properties and formally describe their monitorability characteristics. Based on our definitive prefixes, we define the *guarantee kernel* of a property, which, when viewed as an interior operator gives a topology equivalent to that of Alpern & Schneider [8]. This enables us to re-prove their classic result, that every property is the intersection of a safety and liveness property, by other means. We then formally characterise various notions of monitorability, including that of Bauer et al. [6] and of Pnueli & Zaks [9], and arrange them into a hierarchy.

Finally, in Section 7, we relate our work to other characterisations of traces and properties, as well as to other multi-valued variants of LTL. All of our work has been mechanised in the Isabelle/HOL proof assistant, proofs of which are available for download [10, 11]. The mechanised proof scripts are intended as a comprehensive library, and, being included in Isabelle's Archive of Formal Proofs, are maintained to ensure they are always compatible with the current stable release of Isabelle. An extended abstract of (part of) this paper appeared as [12]. The present paper contains signifi-

cant additional work expanding on monitorability for linear-time properties, providing topological characterisations for various monitorability classes and demonstrating a hierarchy between them.

2. Definitive Prefix Sets

In this section, we develop a theory of *definitive prefixes*, which we will use in Section 4 to give a semantics to LTL_3, and in Section 6 to define various classes of properties.

Given a set of *atomic propositions*, the directly observable properties of a state of a system are the atomic propositions that hold in that state. For this reason we abstractly model a state simply as a set of atomic propositions. Let Σ be a set of such states. A *trace* t may be finite (in Σ^*) or infinite (in Σ^ω). We denote the set of all traces, i.e. $\Sigma^* \cup \Sigma^\omega$, as Σ^∞. We sometimes address a sequence $a_1 a_2 \cdots$ of atomic propositions as a trace; formally this refers to any sequence $\sigma_1 \sigma_2 \cdots \in \Sigma^\infty$ with $a_i \in \sigma_i$ for all relevant i. Two traces t and u may be concatenated in the obvious way, written as tu. If t is infinite, then $tu = t$. The empty trace is denoted ε. We denote the set of *prefixes* of a trace t as $\downarrow t$:

$$\downarrow t \triangleq \{u \mid \exists v \in \Sigma^\infty.\ t = uv\}$$

We also generalise this notation to sets, so $\downarrow X$ is the set of all prefixes of traces in X. The set of all *extensions* of a trace t is likewise written as $\uparrow t$:

$$\uparrow t \triangleq \{tu \mid u \in \Sigma^\infty\}$$

The set of *definitive prefixes* of a set of traces X is written $\lightning X$. This is the set of all traces for which all extensions are a prefix of a trace in X.

$$\lightning X \triangleq \{t \mid \uparrow t \subseteq \downarrow X\}$$

Equivalently, t is a definitive prefix of X iff X contains all infinite extensions of t: $\lightning X = \{t \mid \uparrow t \cap \Sigma^\omega \subseteq X\}$. Intuitively, this means that $\lightning X$ contains all those traces for which reaching $X \cap \Sigma^\omega$ is in some way *inevitable*, even if it hasn't happened yet. Definitive prefixes are therefore similar to the notion of *good* and *bad* prefixes from Kupferman & Vardi [13], just without any moral judgement (see Section 6.2.1).

A set X of traces is called *definitive* iff $X = \lightning X$. Let $\mathcal{D} \subseteq \mathscr{P}(\Sigma^\infty)$ denote the set of all definitive sets.

For example, the set $\{t \in \Sigma^\infty \mid \exists uv.\ t = u\mathsf{a}v\}$ (i.e., all traces that contain the state a) is definitive, as it contains all its definitive prefixes (i.e., those prefixes that contain a).

For any set of traces X, we have the following straightforwardly from the definitions:

- All definitive prefixes are prefixes, i.e. $\lightning X \subseteq \downarrow X$.
- The set $\lightning X$ itself is definitive, i.e. $\lightning \lightning X = \lightning X$.
- Any extension of a definitive prefix is also a definitive prefix, that is, $\forall t \in \lightning X.\ \uparrow t \subseteq \lightning X$.
- The definitive prefix operator $\lightning$ distributes over intersection, that is, $\lightning(\bigcap_{i\in I} X_i) = \bigcap_{i\in I} \lightning X_i$.

The sets $\varnothing$ and Σ^∞ are both definitive, and the definitive sets are closed under intersection, i.e., for a set of definitive sets S, $\lightning \bigcap S = \bigcap S$. This follows from the distributivity theorem above. The definitive sets are not closed under union, however. To see why, consider when $\Sigma = \{\mathtt{A}, \mathtt{B}\}$ and the set $X_\mathtt{A}$ contains all traces starting with $\mathtt{A}$ and the set $X_\mathtt{B}$ contains all traces starting with $\mathtt{B}$. The sets $X_\mathtt{A}$ and $X_\mathtt{B}$ are both definitive, but their union is not: neither $X_\mathtt{A}$ nor $X_\mathtt{B}$ contain the empty trace ε, but $\varepsilon \in \lightning(X_\mathtt{A} \cup X_\mathtt{B})$, as all extensions of ε (i.e. all non-empty traces) must begin with either $\mathtt{A}$ or $\mathtt{B}$.

2.1. Lattice Properties

Define the *definitive union*, written $\bigcup\!\!\!\!\lightning\, S$ or $X \uplus Y$ in the binary case, as merely the definitive prefixes of the union:

$$\bigcup\!\!\!\!\lightning\, S \triangleq \lightning \bigcup S$$

Theorem 2.1. *The definitive union gives least upper bounds for definitive sets ordered by set inclusion, i.e., for a set $S \subseteq \mathcal{D}$ of definitive sets:*

- *For all $X \in S$, $X \subseteq \bigcup\!\!\!\!\lightning\, S$.*
- *If there is a definitive set Z such that $\forall X \in S.\ X \subseteq Z$, then $\bigcup\!\!\!\!\lightning\, S \subseteq Z$.*

Proof. Follows from definitions. □

Thus, the definitive sets $\mathcal{D}$ ordered by set inclusion form a complete lattice, where the supremum is the definitive union, the infimum is the intersection, the greatest element $\top$ is Σ^∞ and the least element $\bot$ is $\varnothing$.

2.2. Isomorphism to Linear-time Temporal Properties

Theorem 2.2. *Define the lower adjoint* $\mathsf{Pr} : \mathcal{D} \to \mathscr{P}(\Sigma^\omega)$ *as* $\mathsf{Pr}(X) = X \cap \Sigma^\omega$*, and the upper adjoint* $\mathsf{Df} : \mathscr{P}(\Sigma^\omega) \to \mathcal{D}$ *as* $\mathsf{Df}(P) = \lightning P$*. We have, for any definitive set* X *and linear-time temporal property* P*:*

$$\mathsf{Pr}(X) = P \text{ if and only if } X = \mathsf{Df}(P)$$

Proof. Proving each direction separately:

$\Longrightarrow$ It suffices to show that $\mathsf{Df}(\mathsf{Pr}(X)) = X$, i.e. $\lightning(X \cap \Sigma^\omega) = X$. Because any extension of a definitive prefix is also a definitive prefix, and all traces in X are definitive prefixes, every finite trace in X must be a prefix of some infinite trace in X. Thus, the infinite traces in X alone are sufficient to describe all their definitive prefixes, i.e., all traces in X.

$\Longleftarrow$ It suffices to show that $\mathsf{Pr}(\mathsf{Df}(P)) = P$, i.e. $\lightning P \cap \Sigma^\omega = P$. Recall that the definitive prefixes of P are all those traces t for which all extensions of t are a prefix of a trace in P. For an infinite trace $t \in P$, the set of extensions $\uparrow t$ is just $\{t\}$, which is surely contained in $\downarrow P$. Therefore, for a linear-time temporal property P (consisting only of infinite traces), we can conclude $P \subseteq \lightning P$. In fact, $\lightning P$ consists of all the (infinite) traces of P as well as possibly some finite prefixes of these. Hence $\lightning P \cap \Sigma^\omega = P$. □

Theorem 2.3. Pr *(and likewise for* Df*) is monotone and preserves least upper and greatest lower bounds, i.e:*

- *If* $A \subseteq B$ *then* $\mathsf{Pr}(A) \subseteq \mathsf{Pr}(B)$
- $\mathsf{Pr}(\bigcap_{i \in I} X_i) = \bigcap_{i \in I} \mathsf{Pr}(X_i)$
- $\mathsf{Pr}(\lightning \bigcup_{i \in I} X_i) = \bigcup_{i \in I} \mathsf{Pr}(X_i)$

Proof. The first two statements follow directly from definitions. Preservation of least upper bounds requires more finesse. As we have already seen in the proof of Theorem 2.2, for any set of traces S, $\mathsf{Pr}(\lightning S) = \mathsf{Pr}(S)$. This means that $\mathsf{Pr}(\lightning \bigcup_{i \in I} X_i) = \mathsf{Pr}(\bigcup_{i \in I} X_i) = \bigcup_{i \in I} \mathsf{Pr}(X_i)$ as required. □

These theorems say that $(\mathsf{Pr}, \mathsf{Df})$ forms a *lattice isomorphism* between definitive sets and linear-time temporal properties.

$$
\begin{array}{lllll}
\text{Formulae} & \varphi, \psi & ::= & \top \mid a \\
 & & \mid & \neg\varphi \\
 & & \mid & \varphi \wedge \psi \\
 & & \mid & \bigcirc \varphi \\
 & & \mid & \varphi \,\mathcal{U}\, \psi \\
\text{Atomic propositions} & a & \in & A \\
\text{States} & \Sigma & = & \mathscr{P}(A) \\
\text{Traces} & t, u & \in & \Sigma^\infty
\end{array}
$$

Abbreviations:

$$
\begin{array}{lcl}
\bot & \triangleq & \neg\top \\
\varphi \vee \psi & \triangleq & \neg(\neg\varphi \wedge \neg\psi) \\
\Diamond \varphi & \triangleq & \top \,\mathcal{U}\, \varphi \\
\varphi \,\mathcal{R}\, \psi & \triangleq & \neg(\neg\varphi \,\mathcal{U}\, \neg\psi) \\
\Box \varphi & \triangleq & \bot \,\mathcal{R}\, \varphi
\end{array}
$$

Figure 1: Syntax of LTL

3. Linear-time Temporal Logic

Figure 1 describes the syntax of LTL formulae and adjacent definitions. LTL extends propositional logic over states with temporal operators to produce a logic over traces — sequences of states.

Our formulation takes conjunction, negation, atomic propositions and the temporal operators *next* ($\bigcirc$) and *until* ($\mathcal{U}$) as primitive, with disjunction and the temporal operators *eventually* ($\Diamond$), *always* ($\Box$) and *release* ($\mathcal{R}$) derived from these primitives.

Figure 2 gives the semantics of conventional LTL, as a satisfaction relation whose models are infinite traces. Here t_0 denotes the first state of a trace t. As we only include future temporal operators, we can advance to the future by dropping initial prefixes from the trace. The notation $t_{|n}$ denotes the trace t without the first n states. If n is greater than the length of t, the result of $t_{|n}$ is the empty trace ε. Note that our *until* operator ($\mathcal{U}$) is *strong*, in that $\varphi \,\mathcal{U}\, \psi$ requires that ψ eventually becomes true at some point in the trace.

Bauer et al. [5] describe LTL_3 as a *three-valued* logic that interprets LTL formulae on finite prefixes to obtain a truth value in $\mathbb{B}_3 = \{\mathtt{T}, \mathtt{F}, \mathtt{?}\}$. For

$$
\begin{array}{lll}
t \vDash \top & & \\
t \vDash a & \text{iff} & a \in t_0 \\
t \vDash \neg\varphi & \text{iff} & t \nvDash \varphi \\
t \vDash \varphi \wedge \psi & \text{iff} & t \vDash \varphi \text{ and } t \vDash \psi \\
t \vDash \bigcirc\varphi & \text{iff} & t_{|1} \vDash \varphi \\
t \vDash \varphi \,\mathcal{U}\, \psi & \text{iff} & \text{there exists } i \text{ s.t. } t_{|i} \vDash \psi \text{ and } \forall j < i.\ t_{|j} \vDash \varphi
\end{array}
$$

Figure 2: Semantics of conventional LTL

a formula φ and a finite prefix t, the truth value $\mathtt{T}$ indicates that φ can be definitively established from t alone, whereas $\mathtt{F}$ indicates that φ can be definitively refuted from t alone. The third value $\mathtt{?}$ indicates that the formula φ can neither be established nor refuted from t alone:

$$
[t \vDash_3 \varphi] = \begin{cases} \mathtt{T} & \text{if } \forall u \in \Sigma^\omega.\ tu \vDash \varphi \\ \mathtt{F} & \text{if } \forall u \in \Sigma^\omega.\ tu \nvDash \varphi \\ \mathtt{?} & \text{otherwise} \end{cases}
$$

Because the truth value $\mathtt{?}$ indicates merely that neither $\mathtt{T}$ nor $\mathtt{F}$ apply, LTL_3 can be better understood as a two-valued *partial logic* [14], where $\mathtt{?}$ indicates the *absence* of a truth value. The logic is partial in the sense that its evaluation is a partial function from possibly-finite prefixes to $\{\mathtt{T}, \mathtt{F}\}$. The value $\mathtt{?}$ is not a third semantic state, but a computational placeholder for when a prefix is not contained within a property's definitive set, and thus no truth value has yet been assigned. In this view, LTL_3 only gives truth values when the trace is *definitive*, i.e., when the answer given will not change regardless of how the trace is extended.

Bauer et al. [5] note that this presentation of LTL_3 is inherently non-inductive, i.e., the answer given for a compound formula cannot be produced by combining the answers for its components. To see why, consider the formula $\varphi = \Diamond a \vee \Diamond \neg a$. Using the semantics above, we have $[\varepsilon \vDash_3 \varphi] = \mathtt{T}$ but each component of φ produces no definitive answer for the empty trace, i.e. $[\varepsilon \vDash_3 \Diamond a] = [\varepsilon \vDash_3 \Diamond \neg a] = \mathtt{?}$. Likewise both components of the formula $\psi = \Diamond b \vee \Diamond \neg c$ produce the answer $\mathtt{?}$ for the empty trace, but unlike φ, we have $[\varepsilon \vDash_3 \psi] = \mathtt{?}$. Therefore, there is no way to combine two $\mathtt{?}$ answers in a disjunction that produces correct answers for both φ and ψ. Because of this, Bauer et al. [5] claim that inductive semantics are *impossible* for LTL_3. As we shall see, however, this claim applies only to the multi-valued semantics

defined above. In our development, which associates sets of traces to each formula, the semantics for a given formula can indeed be compositionally constructed from the semantics of its components.

4. Semantics of LTL and LTL_3

4.1. Answer-indexed Families

We give a semantics to LTL_3 by compositionally assigning to each formula an *answer-indexed family* of definitive sets. In general, an *answer-indexed family* is a function that, given an answer (e.g. a value in $\mathbb{B} = \{\mathtt{T}, \mathtt{F}\}$), produces a set of models (depending on the logic, this could be a set of states, a definitive set, a linear-time temporal property, etc.). This set contains all those models which produce the given answer for the formula in question. In this way, we invert the traditional presentation of multi-valued logics, where the truth value is the output of a satisfaction function, and instead take the desired answer a as an *input* and produce models as an *output*: $a = [\sigma \vDash \varphi]$ becomes $\sigma \in [\![\varphi]\!]\ a$. An answer-indexed family for conventional, infinite-trace LTL therefore produces a linear-time temporal property as an output:

$$\Phi, \Psi \in \mathbb{B} \to \mathscr{P}(\Sigma^\omega)$$

whereas an answer-indexed family for LTL_3 produces a definitive set as an output:

$$\Phi, \Psi \in \mathbb{B} \to \mathcal{D}$$

To begin with, we define an alternative semantics for *conventional* LTL in terms of answer-indexed families of linear-time temporal properties. In this setting, the LTL formula $\Diamond \mathsf{a}$, for example, will have an answer-indexed family that maps $\mathtt{T}$ to the set of traces in which a eventually holds, and $\mathtt{F}$ to the set of traces in which a never holds.

We define various operations on answer-indexed families, one for each kind of LTL constructor:

$$\begin{array}{llll} \textcircled{\scriptstyle\top}\ \mathtt{T} = \Sigma^\omega & (\Phi\ \textcircled{\scriptstyle\wedge}\ \Psi)\ \mathtt{T} = \Phi\ \mathtt{T} \cap \Psi\ \mathtt{T} & (\Phi\ \textcircled{\scriptstyle\vee}\ \Psi)\ \mathtt{T} = \Phi\ \mathtt{T} \cup \Psi\ \mathtt{T} & (\textcircled{\scriptstyle\neg}\ \Phi)\ \mathtt{T} = \Phi\ \mathtt{F} \\ \textcircled{\scriptstyle\top}\ \mathtt{F} = \varnothing & (\Phi\ \textcircled{\scriptstyle\wedge}\ \Psi)\ \mathtt{F} = \Phi\ \mathtt{F} \cup \Psi\ \mathtt{F} & (\Phi\ \textcircled{\scriptstyle\vee}\ \Psi)\ \mathtt{F} = \Phi\ \mathtt{F} \cap \Psi\ \mathtt{F} & (\textcircled{\scriptstyle\neg}\ \Phi)\ \mathtt{F} = \Phi\ \mathtt{T} \end{array}$$

Note that the set operations used for the $\mathtt{F}$ answer are always the duals of the operations used for the $\mathtt{T}$ answer, which means that for conventional LTL, the set produced for the $\mathtt{F}$ answer is always the complement of the set for

$$
\begin{aligned}
[\![\top]\!] &= \textcircled{\scriptsize T} \\
[\![a]\!] &= \lceil a \rfloor \\
[\![\neg\varphi]\!] &= \ominus [\![\varphi]\!] \\
[\![\varphi \wedge \psi]\!] &= [\![\varphi]\!] \mathbin{\textcircled{\scriptsize $\wedge$}} [\![\psi]\!] \\
[\![\varphi \vee \psi]\!] &= [\![\varphi]\!] \mathbin{\textcircled{\scriptsize $\vee$}} [\![\psi]\!] \\
[\![\bigcirc \varphi]\!] &= \circledcirc [\![\varphi]\!] \\
[\![\varphi \,\mathcal{U}\, \psi]\!] &= [\![\varphi]\!] \mathbin{@} [\![\psi]\!]
\end{aligned}
$$

Figure 3: LTL semantics using answer-indexed families

the $\mathtt{T}$ answer. This means that the operator for negation ($\ominus$) can simply swap the places of the set and its complement. This is akin to performing a conversion to negation normal form "just-in-time" as we evaluate a formula. For atomic propositions a, the corresponding answer-indexed family maps $\mathtt{T}$ to the set of all traces that begin with a state containing a, and $\mathtt{F}$ to its complement:

$$
\begin{aligned}
\lceil a \rfloor\ \mathtt{T} &= \{t \mid t \in \Sigma^\omega \wedge a \in t_0\} \\
\lceil a \rfloor\ \mathtt{F} &= \{t \mid t \in \Sigma^\omega \wedge a \notin t_0\}
\end{aligned}
$$

The semantic operator for $\bigcirc\varphi$ formulae prepends one state to all the corresponding traces for φ, analogously to the conventional LTL semantics in Figure 2:

$$
\begin{aligned}
(\circledcirc\, \Phi)\ \mathtt{T} &= \{t \mid t_{|1} \in \Phi\ \mathtt{T}\} \\
(\circledcirc\, \Phi)\ \mathtt{F} &= \{t \mid t_{|1} \in \Phi\ \mathtt{F}\}
\end{aligned}
$$

The $\mathtt{T}$ case of the semantic operator for $\varphi\,\mathcal{U}\,\psi$ formulae is also defined analogously to Figure 2, with the $\mathtt{F}$ case being the complement:

$$
\begin{aligned}
(\Phi\ @\ \Psi)\ \mathtt{T} &= \{t \mid \exists k.(\forall i < k.t_{|i} \in \Phi\ \mathtt{T}) \wedge t_{|k} \in \Psi\ \mathtt{T}\} \\
(\Phi\ @\ \Psi)\ \mathtt{F} &= \{t \mid \forall k.(\exists i < k.t_{|i} \in \Phi\ \mathtt{F}) \vee t_{|k} \in \Psi\ \mathtt{F}\}
\end{aligned}
$$

Finally, we put all of these semantic operators to use in Figure 3, which gives a compositional, inductive semantics to conventional LTL using these operators.

Theorem 4.1 (Equivalence to conventional semantics)**.** *Answer-indexed family LTL semantics assigns the same truth values to a given trace for a given formula as conventional LTL semantics:*

- $(t \vDash \varphi) \iff (t \in [\![\varphi]\!]\ \mathtt{T})$
- $\neg(t \vDash \varphi) \iff (t \in [\![\varphi]\!]\ \mathtt{F})$

Proof. This is proven straightforwardly by induction on φ, justified in the same way as a conversion to negation normal form. □

4.2. The Prepend Operation

To give a semantics to LTL_3, our answer-indexed families will produce definitive sets, rather than linear-time temporal properties. To this end, we will define an auxiliary operation on definitive sets called *prepend*, written $\triangleright X$, which gives all traces whose tails are in X:

$$\triangleright X \triangleq \{t \mid t_{|1} \in X\}$$

Theorem 4.2. *The* prepend *operation is closed for definitive sets. That is, if X is definitive, then $\triangleright X$ is definitive.*

Proof. We must show that $\lightning(\triangleright X) = \triangleright X$ for any definitive set X. Showing each direction separately:

$\Longrightarrow$ Given a definitive prefix $t \in \lightning(\triangleright X)$, we must show that $t \in \triangleright X$. If $t = \varepsilon$, then this implies that $\triangleright X = \Sigma^\infty$ and therefore $t \in \triangleright X$. If $t = \sigma u$, because $\downarrow(\triangleright X) = \triangleright(\downarrow X)$, we can conclude $\uparrow \sigma u \subseteq \triangleright(\downarrow X)$. Taking the tail of both sides, we can see that $\uparrow u \subseteq \downarrow X$ and therefore $u \in X$ as X is definitive. Prepending σ to both sides, we conclude that $\sigma u \in \triangleright X$ as required.

$\Longleftarrow$ Given a prefix $t \in \triangleright X$, we must show that $t \in \lightning(\triangleright X)$. If $t = \varepsilon$, this means that $X = \triangleright X = \lightning(\triangleright X) = \Sigma^\infty$ as X is definitive. If $t = \sigma u$, we know that $u \in X$. As X is definitive, all extensions of u are also in X. Therefore $\triangleright(\uparrow u) \subseteq X$ and thus $\sigma u \in \lightning(\triangleright X)$. □

4.3. Semantics for LTL_3

The semantic operators for LTL_3 resemble that of conventional LTL, except that now we work with definitive sets rather than linear-time temporal properties.

$$\begin{aligned}
\textcircled{\scriptstyle\top}_3\ \mathtt{T} &= \Sigma^\infty & \textcircled{\scriptstyle\top}_3\ \mathtt{F} &= \varnothing \\
(\textcircled{\scriptstyle\neg}_3\,\Phi)\ \mathtt{T} &= \Phi\ \mathtt{F} & (\textcircled{\scriptstyle\neg}_3\,\Phi)\ \mathtt{F} &= \Phi\ \mathtt{T} \\
(\Phi\ \textcircled{\scriptstyle\wedge}_3\ \Psi)\ \mathtt{T} &= \Phi\ \mathtt{T} \cap \Psi\ \mathtt{T} & (\Phi\ \textcircled{\scriptstyle\wedge}_3\ \Psi)\ \mathtt{F} &= \Phi\ \mathtt{F} \uplus \Psi\ \mathtt{F} \\
(\Phi\ \textcircled{\scriptstyle\vee}_3\ \Psi)\ \mathtt{T} &= \Phi\ \mathtt{T} \uplus \Psi\ \mathtt{T} & (\Phi\ \textcircled{\scriptstyle\vee}_3\ \Psi)\ \mathtt{F} &= \Phi\ \mathtt{F} \cap \Psi\ \mathtt{F}
\end{aligned}$$

All of the sets produced by these answer-indexed families are definitive, as Σ^∞ and $\varnothing$ are both definitive sets and definitive sets are closed under intersection and definitive union. Unlike with conventional LTL, the set for the `F` answer is not the complement of the set for the `T` answer, as definitive sets are not closed under complement. The set for `T` contains all traces that are sufficient to definitively satisfy the formula, and the set for `F` contains all traces that are sufficient to definitively refute the formula.

For an atomic proposition a, the set for `T` contains all non-empty traces that begin with a state that satisfies a, and the set for `F` contains all non-empty traces that begin with a state that does not satisfy a. However, if a is trivial, in the sense that *all* or *no* possible states satisfy a, then these sets are not definitive, as the excluded empty trace ε would also be definitive for these sets. Thus, we take the definitive prefixes of these sets to account for this possibility:

$$\begin{aligned}
\ulcorner a \urcorner_3\ \mathtt{T} &= \lightning\{t \mid t \neq \varepsilon \wedge a \in t_0\} \\
\ulcorner a \urcorner_3\ \mathtt{F} &= \lightning\{t \mid t \neq \varepsilon \wedge a \notin t_0\}
\end{aligned}$$

For the $\bigcirc$ operator, we make use of the prepend operator, which by Theorem 4.2 produces definitive sets:

$$\begin{aligned}
(\textcircled{\scriptstyle\bigcirc}_3\,\Phi)\ \mathtt{T} &= \triangleright\ (\Phi\ \mathtt{T}) \\
(\textcircled{\scriptstyle\bigcirc}_3\,\Phi)\ \mathtt{F} &= \triangleright\ (\Phi\ \mathtt{F})
\end{aligned}$$

For the $\mathcal{U}$ operator, we construct our semantics iteratively, building up by repeatedly prepending states. Here the notation f^k indicates the self-

$$
\begin{aligned}
[\![\top]\!]_3 &= \textcircled{\scriptsize T}_3 \\
[\![a]\!]_3 &= \lceil a \rfloor_3 \\
[\![\neg\varphi]\!]_3 &= \ominus_3\ [\![\varphi]\!]_3 \\
[\![\varphi \wedge \psi]\!]_3 &= [\![\varphi]\!]_3\ \textcircled{\scriptsize $\wedge$}_3\ [\![\psi]\!]_3 \\
[\![\varphi \vee \psi]\!]_3 &= [\![\varphi]\!]_3\ \textcircled{\scriptsize $\vee$}_3\ [\![\psi]\!]_3 \\
[\![\bigcirc\varphi]\!]_3 &= \textcircled{\scriptsize $\bigcirc$}_3\ [\![\varphi]\!]_3 \\
[\![\varphi\ \mathcal{U}\ \psi]\!]_3 &= [\![\varphi]\!]_3\ \textcircled{\scriptsize $\mathcal{U}$}_3\ [\![\psi]\!]_3
\end{aligned}
$$

Figure 4: LTL_3 semantics using answer-indexed families

composition of f k times, i.e. $f^0(x) = x$ and $f^{k+1}(x) = f^k(f(x))$:

$$
\begin{aligned}
(\Phi\ \textcircled{\scriptsize $\mathcal{U}$}_3\ \Psi)\ \mathtt{T} &= \bigsqcup_{k\in\mathbb{N}} f^k(\Psi\ \mathtt{T}), \text{ where } f(X) = \rhd X \cap \Phi\ \mathtt{T} \\
(\Phi\ \textcircled{\scriptsize $\mathcal{U}$}_3\ \Psi)\ \mathtt{F} &= \bigcap_{k\in\mathbb{N}} f^k(\Psi\ \mathtt{F}), \text{ where } f(X) = \rhd X \uplus \Phi\ \mathtt{F}
\end{aligned}
$$

Because definitive sets are closed under intersection, definitive union and the prepend operator, we can see that that our $\textcircled{\scriptsize $\mathcal{U}$}$ operator also produces definitive sets by a simple inductive argument on the natural number k. Using all of these operations, we construct an inductive, compositional semantics for LTL_3 in Figure 4.

Theorem 4.3 (Equivalence to original LTL_3 definition)**.** *Let t be a finite prefix and φ be an LTL formula. Then:*

- $t \in [\![\varphi]\!]_3\ \mathtt{T} \iff \forall u \in \Sigma^\omega.\ tu \in [\![\varphi]\!]\ \mathtt{T}$
- $t \in [\![\varphi]\!]_3\ \mathtt{F} \iff \forall u \in \Sigma^\omega.\ tu \in [\![\varphi]\!]\ \mathtt{F}$

Proof. This follows directly from the definition of definitive sets, as $[\![\varphi]\!]_3\ \mathtt{T}$ and $[\![\varphi]\!]_3\ \mathtt{F}$ are both definitive. □

Theorem 4.3 shows that our inductive semantics coincides with the original non-inductive semantics given for LTL_3. If we view our semantics through the lens of the isomorphism in Theorem 2.2, however, we see that this semantics is also equivalent to the semantics of conventional LTL:

Theorem 4.4 (Equivalence to conventional LTL)**.** *For all formulae φ:*

- $\mathsf{Pr}(\llbracket\varphi\rrbracket_3\ \mathtt{T}) = \llbracket\varphi\rrbracket\ \mathtt{T}$
- $\mathsf{Pr}(\llbracket\varphi\rrbracket_3\ \mathtt{F}) = \llbracket\varphi\rrbracket\ \mathtt{F}$

Proof. The two statements are shown simultaneously by induction on φ:

$\varphi = \top$: $\mathsf{Pr}(\Sigma^\infty) = \Sigma^\omega$ by definition.

$\varphi = a$: Because $\mathsf{Pr}(\lightning S) = \mathsf{Pr}(S)$ as seen in the proof of Theorem 2.2, $\mathsf{Pr}(\lceil\!\lfloor a \rfloor\!\rceil_3\ \mathtt{T}) = \lceil\!\lfloor a \rfloor\!\rceil\ \mathtt{T}$ and likewise $\mathsf{Pr}(\lceil\!\lfloor a \rfloor\!\rceil_3\ \mathtt{F}) = \lceil\!\lfloor a \rfloor\!\rceil\ \mathtt{F}$.

$\varphi = \neg\varphi'$: Follows from inductive hypotheses.

$\varphi = \varphi' \wedge \psi'$: Follows from inductive hypotheses as Pr preserves greatest lower and least upper bounds.

$\varphi = \bigcirc\varphi'$: Follows from inductive hypotheses as the prepend operator $\triangleright$ commutes with Pr.

$\varphi = \varphi'\ \mathcal{U}\ \psi'$: Because Pr commutes with $\triangleright$ and preserves least upper and greatest lower bounds, we can show that $\mathsf{Pr}(\bigsqcup_{k\in\mathbb{N}} f^k(\llbracket\psi'\rrbracket_3\ \mathtt{T}))$ where $f(X) = \triangleright X \cap \llbracket\varphi\rrbracket_3\ \mathtt{T}$, is equal to $\bigcup_{k\in\mathbb{N}} g^k(\mathsf{Pr}(\llbracket\psi'\rrbracket_3\ \mathtt{T}))$ where $g(X) = \triangleright X \cap \mathsf{Pr}(\llbracket\varphi'\rrbracket_3\ \mathtt{T})$ by induction on on the natural number k. By the inductive hypotheses, this is equal to $\bigcup_{k\in\mathbb{N}} g^k(\llbracket\psi'\rrbracket\ \mathtt{T})$ where $g(X) = \triangleright X \cap \llbracket\varphi'\rrbracket\ \mathtt{T}$. This can be shown by another simple induction to be equal to the original definition in the conventional LTL semantics $\{t \mid \exists k.(\forall i < k. t_{|i} \in \llbracket\varphi'\rrbracket\ \mathtt{T}) \wedge t_{|k} \in \llbracket\psi'\rrbracket\ \mathtt{T}\}$. The cases for the $\mathtt{F}$ answer are proved similarly. □

Because of this equivalence theorem, we can now express the relationship between the set for $\mathtt{T}$ and the set for $\mathtt{F}$ in our LTL_3 semantics. In LTL_3, while the two sets do not overlap, they are not perfect complements of each other as they were in conventional LTL, as definitive sets are not closed under complement. Instead, the $\mathtt{F}$ set is the definitive set corresponding to the *linear-time temporal property* containing all infinite traces not in the $\mathtt{T}$ set.

Theorem 4.5 (Excluded Middle)**.** *For all formulae φ:*

$$\llbracket\varphi\rrbracket_3\ \mathtt{T} = \lightning(\Sigma^\omega \setminus \llbracket\varphi\rrbracket_3\ \mathtt{F}) \qquad \textit{and} \qquad \llbracket\varphi\rrbracket_3\ \mathtt{F} = \lightning(\Sigma^\omega \setminus \llbracket\varphi\rrbracket_3\ \mathtt{T})$$

Proof. It is a straightforward consequence of Theorem 4.1 that $[\![\varphi]\!]\ \mathtt{T} = \Sigma^\omega \setminus [\![\varphi]\!]\ \mathtt{F}$ (*). Then:

$$\begin{aligned}
[\![\varphi]\!]_3\ \mathtt{T} &= \mathsf{Df}(\mathsf{Pr}([\![\varphi]\!]_3\ \mathtt{T})) && \text{(Theorem 2.2)}\\
&= \mathsf{Df}([\![\varphi]\!]\ \mathtt{T}) && \text{(Theorem 4.4)}\\
&= \mathsf{Df}(\Sigma^\omega \setminus [\![\varphi]\!]\ \mathtt{F}) && \text{(*)}\\
&= \mathsf{Df}(\Sigma^\omega \setminus \mathsf{Pr}([\![\varphi]\!]_3\ \mathtt{F})) && \text{(Theorem 4.4)}\\
&= \mathsf{Df}(\Sigma^\omega \setminus ([\![\varphi]\!]_3\ \mathtt{F} \cap \Sigma^\omega)) && \text{(Definition of Pr)}\\
&= \lightning(\Sigma^\omega \setminus [\![\varphi]\!]_3\ \mathtt{F}) && \text{(Definition of Df)}
\end{aligned}$$

□

5. Formula Progression

Formula progression is a technique first introduced by Kabanza et al. [15, 16] that evaluates a formula stepwise against states in a style reminiscent of operational semantics or the Brzozowski derivative. This technique was used by O'Connor & Wickström [17] as the basis for their testing algorithm, and by Bauer & Falcone [7] for decentralised monitoring of component-based systems. Figure 5 gives an overview of formula progression rules for LTL. The judgement $\varphi \xrightarrow{\sigma} \psi$ states that, to prove φ, it suffices to prove ψ for the tail of our trace if the head of our trace is $\sigma \in \Sigma$. Note that these rules are total and syntax-directed on the left-hand formula φ. This means that these rules taken together constitute a definition of a total function that takes φ and σ as input and produces ψ as output. We generalise this notation to finite prefixes, so that for a finite trace $t = \sigma_0 \dots \sigma_n$, the notation $\varphi \xrightarrow{t} \psi$ just means $\varphi \xrightarrow{\sigma_0} \cdots \xrightarrow{\sigma_n} \psi$. Repeated application of these rules, however, can lead to exponential blowup in the size of the formula. While both O'Connor & Wickström [17] and Bauer & Falcone [7] report that interleaving this progression with formula simplification at each step keeps the formulae tractable for most practical use cases, Roşu & Havelund [18] warn that pathological exponential cases still exist.

Bauer & Falcone [7] state that formula progression can serve as an *alternative semantics* for LTL_3 on finite traces, where a formula φ is considered definitively true for a finite trace t iff $\varphi \xrightarrow{t} \top$, definitively false iff $\varphi \xrightarrow{t} \bot$, and is unknown otherwise. While it goes unmentioned in their paper, here the implicit simplification steps are not just a performance optimisation, but are vital to ensure that the semantics given via formula progression is complete with respect to the standard LTL_3 semantics. To see why, consider the formula $\Diamond a$. Let σ_a be a state where $a \in \sigma_a$. Then the formula $\Diamond a$ should be

$$\boxed{\varphi \xrightarrow{\sigma} \psi}$$

$$\frac{}{\top \xrightarrow{\sigma} \top} \qquad \frac{a \in \sigma}{a \xrightarrow{\sigma} \top} \qquad \frac{a \notin \sigma}{a \xrightarrow{\sigma} \bot} \qquad \frac{}{\bigcirc \varphi \xrightarrow{\sigma} \varphi} \qquad \frac{\varphi \xrightarrow{\sigma} \varphi' \qquad \psi \xrightarrow{\sigma} \psi'}{\varphi \vee \psi \xrightarrow{\sigma} \varphi' \vee \psi'}$$

$$\frac{\varphi \xrightarrow{\sigma} \varphi'}{\neg\varphi \xrightarrow{\sigma} \neg\varphi'} \qquad \frac{\varphi \xrightarrow{\sigma} \varphi' \qquad \psi \xrightarrow{\sigma} \psi'}{\varphi \wedge \psi \xrightarrow{\sigma} \varphi' \wedge \psi'} \qquad \frac{\psi \xrightarrow{\sigma} \psi' \qquad \varphi \xrightarrow{\sigma} \varphi'}{\varphi \,\mathcal{U}\, \psi \xrightarrow{\sigma} \psi' \vee (\varphi' \wedge (\varphi \,\mathcal{U}\, \psi))}$$

Figure 5: Rules for formula progression

considered definitively true for the trace consisting of just σ_a. The formula generated by our formula progression rules, however, would be $\top \vee (\top \wedge \Diamond a)$, which yields the desired formula $\top$ only after logical simplifications are applied. While in this case, the simplifications required are just identities of propositional logic, in general such straightforward simplifications alone are insufficient. For example, consider the formula $(\bigcirc a) \vee (\Diamond \neg a)$. According to the semantics of LTL_3 presented above, this formula should be considered definitively true for the empty trace ε, as it is a tautology. Temporally local simplifications such as those used by O'Connor & Wickström [17], however, would not be able to determine that this formula is a tautology until after one state has been observed. Therefore, in order for formula progression to align correctly with the semantics of LTL_3, the simplification must transform *all* tautologies into $\top$ and *all* absurdities into $\bot$. A simple, although slow way to implement such a simplifier would be to convert both the formula and its negation into Büchi automata, and perform cycle detection to check for emptiness. For our development, we abstract away from such syntactic simplification procedures by working only on the level of our model-based semantics. As can be seen in our Theorem 5.1 given below, we do not seek a specific syntactic tautology $\top$ or absurdity $\bot$, but rather refer to any formula with trivial semantics. A purely syntactic characterisation, by contrast, would require a full accounting of the simplification procedure, which is outside the scope of our development here.

The rules given in Figure 5 operate on one state at a time, whereas our semantics are on the level of entire traces. Therefore, in order to show soundness and completeness (for finite traces) of our formula progression rules with respect to our semantics, we must first prove two lemmas which relate a single step of formula progression to our semantics.

The first lemma states that for one step of formula progression $\varphi \xrightarrow{\sigma} \varphi'$, prepending σ to the traces that satisfy/refute the *output* formula φ' yields traces that satisfy (resp. refute) the *input* formula φ.

Lemma 5.1. *Let φ and φ' be formulae and σ be a state such that $\varphi \xrightarrow{\sigma} \varphi'$. Then:*

- $\triangleright([\![\varphi']\!]_3\ \mathtt{T}) \cap \{t \mid t_0 = \sigma\} \subseteq [\![\varphi]\!]_3\ \mathtt{T}$
- $\triangleright([\![\varphi']\!]_3\ \mathtt{F}) \cap \{t \mid t_0 = \sigma\} \subseteq [\![\varphi]\!]_3\ \mathtt{F}$

Proof. The two statements are shown simultaneously by structural induction on the formula φ (which, as our rules are syntax directed, uniquely determines the output formula φ'). The base cases for $\varphi = \top$ and $\varphi = a$ as well as the inductive cases for the next operator $\bigcirc$ follow directly from definitions. Of the other inductive cases, the cases for conjunction and disjunction require the use of the distributive properties of the lattice of definitive sets, as well as the fact that the prepend operator $\triangleright$ distributes over intersection and definitive union. The cases for negation follow directly from the inductive hypotheses, whereas the cases for the until operator $\mathcal{U}$ require unfolding of the big unions and intersections in the definition of the semantic operator $@_3$ by one step. □

The second lemma states that those traces that satisfy the *input* formula φ and begin with the state σ will have tails that satisfy the *output* formula φ'.

Lemma 5.2. *Let φ and φ' be formulae and σ be a state such that $\varphi \xrightarrow{\sigma} \varphi'$. Then:*

- $[\![\varphi]\!]_3\ \mathtt{T} \cap \{t \mid t_0 = \sigma\} \subseteq \triangleright([\![\varphi']\!]_3\ \mathtt{T})$
- $[\![\varphi]\!]_3\ \mathtt{F} \cap \{t \mid t_0 = \sigma\} \subseteq \triangleright([\![\varphi']\!]_3\ \mathtt{F})$

Proof. As with Theorem 5.1, the two statements are shown simultaneously by structural induction on the formula φ. The base cases and the cases for the next operator $\bigcirc$ are shown just by unfolding definitions, the cases for conjunction and disjunction are shown by use of distributive properties including those of the prepend operator $\triangleright$, negation proceeds directly from the induction hypotheses, and the until operator $\mathcal{U}$ requires unfolding of the semantic operator $@_3$ by one step. □

By combining these two lemmas, we can inductively prove a theorem that relates formula progression to our semantics on the level of entire finite traces. This resembles the informal definition of formula progression semantics given by Bauer & Falcone [7], but with the syntactic requirement that the ultimate formula be $\top$ or $\bot$ replaced by a semantic requirement that it has trivial semantics.

Theorem 5.1. *Let $t \in \Sigma^*$ be a finite trace. Then, for all formulae φ and φ' where $\varphi \xrightarrow{t} \varphi'$:*

- *$t \in [\![\varphi]\!]_3\ \mathtt{T}$ if and only if $[\![\varphi']\!]_3\ \mathtt{T} = \Sigma^\infty$.*
- *$t \in [\![\varphi]\!]_3\ \mathtt{F}$ if and only if $[\![\varphi']\!]_3\ \mathtt{F} = \Sigma^\infty$.*

Proof. By induction on the length of the trace t (where φ and φ' are kept arbitrary). The second statement for $\mathtt{F}$ is proved identically to the first for $\mathtt{T}$, so we present the proof only for $\mathtt{T}$ here.

Base Case ($t = \varepsilon$) It suffices to show that $\varepsilon \in [\![\varphi]\!]_3\ \mathtt{T}$ iff $[\![\varphi]\!]_3\ \mathtt{T} = \Sigma^\infty$. Because $[\![\varphi]\!]_3\ \mathtt{T}$ is a definitive set, and any extension of a definitive prefix is also a definitive prefix, as ε is in $[\![\varphi]\!]_3\ \mathtt{T}$, we can conclude that all traces (i.e. extensions of ε) are in $[\![\varphi]\!]_3\ \mathtt{T}$. The reverse direction of the iff is straightforward.

Inductive Case ($t = \sigma u$) We know that $\varphi_0 \xrightarrow{\sigma} \varphi \xrightarrow{u} \varphi'$ and have the inductive hypothesis that $u \in [\![\varphi]\!]_3\ \mathtt{T} \iff [\![\varphi']\!]_3\ \mathtt{T} = \Sigma^\infty$. We must show that $\sigma u \in [\![\varphi_0]\!]_3\ \mathtt{T} \iff [\![\varphi']\!]_3\ \mathtt{T} = \Sigma^\infty$. Therefore, by the inductive hypothesis, it suffices to show $\sigma u \in [\![\varphi_0]\!]_3\ \mathtt{T} \iff u \in [\![\varphi]\!]_3\ \mathtt{T}$. Showing each direction separately:

$\Longrightarrow$ By Theorem 5.2 we can conclude that $\sigma u \in \rhd([\![\phi]\!]_3\ \mathtt{T})$ and thus that $u \in [\![\varphi]\!]_3\ \mathtt{T}$ by the definition of the prepend operator $\rhd$.

$\Longleftarrow$ By the definition of the prepend operator $\rhd$ we can conclude that $\sigma u \in \rhd([\![\varphi]\!]_3\ \mathtt{T})$ and thus that $\sigma u \in [\![\varphi_0]\!]_3\ \mathtt{T}$ by Theorem 5.1. $\square$

Theorem 5.1 is both a *soundness* and *completeness* proof for formula progression semantics with respect to our model-based semantics, up to finite traces. Soundness here means that a formula φ will only evaluate in formula progression to a tautology for a trace t when t is in $[\![\varphi]\!]_3\ \mathtt{T}$, and likewise will only evaluate to an absurdity when t is in $[\![\varphi]\!]_3\ \mathtt{F}$. This is the $\Longleftarrow$ direction of

the iff in Theorem 5.1. Completeness (or adequacy) up to finite traces means that all finite prefixes that definitively confirm the formula will evaluate in formula progression to a tautology, and all finite prefixes that definitively refute the formula will evaluate to an absurdity. This is the $\Longrightarrow$ direction of the iff in Theorem 5.1.

The progressed formula evaluating to a tautology or an absurdity should be understood purely as a semantic description of when progression has reached a formula with trivial semantics, rather than as an implementation prescription for an explicit universality or emptiness check. Any practical implementation must therefore balance completeness with respect to this characterisation against efficiency. This trade-off can be addressed either by employing complete but potentially expensive procedures, or by adopting tractable but incomplete syntactic simplification techniques [17].

6. Properties and Monitorability

6.1. Linear-time Temporal Properties

Linear-time temporal properties can be broadly categorised into *safety* properties, which state that something "bad" does not happen during execution, and *liveness* properties, which state that something "good" eventually occurs during execution [19]. Alpern & Schneider [8] formalise Lamport's informal concepts of safety and liveness properties by defining a safety property as one that can be definitely refuted by a finite prefix of a trace, and a liveness property P as one such that any finite trace can be extended to a member of P. They define a topological space where safety properties are those sets that are (limit-)closed and liveness properties are those sets that are dense. We also see in later work [13, 20] the concept of *guarantee* (or *co-safety*) properties and *morbidity* (or *co-liveness*) properties, the complements of safety and liveness properties respectively. A *guarantee* property can always be definitively *confirmed* by a finite prefix of a trace,[2] whereas any finite prefix can be extended in such a way as to *refute* a given *morbidity* property.

Our definitive sets include those finite prefixes that can confirm (or refute) the property, enabling us to express these insights about finite prefixes

[2]Thus, *guarantee* could be seen as an alternative formalisation of Lamport's informal concept of a liveness property [19], different from the standard formalisation of Alpern & Schneider [8]. This formalisation of liveness is employed, e.g., by van Glabbeek [21].

directly. We shall use this to define a new operator on properties, which we call the *guarantee kernel*, which, when viewed as an interior operator, gives rise to an equivalent topology to that of Alpern & Schneider [8], enabling us to re-prove their classic result—that every property is the intersection of a safety and liveness property—by other means. These definitions also enable precise definitions of *monitorability* classes, which we discuss in Section 6.2.

6.1.1. Guarantee and Safety Properties

Definition 6.1 (Guarantee kernel)**.** The *guarantee kernel* of a property P, written $\underline{P}$, is the set of all infinite traces that are extensions of finite definitive prefixes of P:

$$\underline{P} \triangleq \uparrow(\lightning P \cap \Sigma^*) \cap \Sigma^\omega$$

Intuitively, traces in the guarantee kernel $\underline{P}$ are those traces $t \in P$ for which membership in P can be confirmed after observing merely a finite prefix of t. This means that $\underline{P}$ is the largest subset of P that is a guarantee property. Thus, we can characterise guarantee properties using the guarantee kernel:

$$P \text{ is a guarantee property} \iff \underline{P} = P$$

The guarantee kernel is a *kernel operator*, that is, it is monotone ($P \subseteq R$ implies $\underline{P} \subseteq \underline{R}$), it is idempotent ($\underline{\underline{P}} = \underline{P}$), and it is a subset of the original property ($\underline{P} \subseteq P$).

Theorem 6.1. *The guarantee kernel distributes over binary intersection, i.e.,* $\underline{P} \cap \underline{R} = \underline{P \cap R}$.

Proof. Showing each direction separately:

$\underline{P} \cap \underline{R} \subseteq \underline{P \cap R}$: Let u be an infinite trace in $\underline{P} \cap \underline{R}$. Then there must exist a definitive finite prefix for P, $t_P \in \downarrow u$. Similarly there must exist a definitive finite prefix for R, $t_R \in \downarrow u$. Because both t_P and t_R are prefixes of u, either $t_P \in \downarrow t_R$ or $t_R \in \downarrow t_P$. If $t_P \in \downarrow t_R$, then t_R must also be a definitive prefix for P as definitive sets are closed under extension, so t_R is definitive finite prefix for $P \cap R$ and thus $u \in \underline{P \cap R}$. Likewise for t_P and R when $t_R \in \downarrow t_P$.

$\underline{P} \cap \underline{R} \supseteq \underline{P \cap R}$: Straightforward as the guarantee kernel is monotone. □

Counterexample 6.1 (Guarantee kernel does not distribute over union)**.** Consider the properties $\Box\Diamond a$ and $\Diamond\Box\neg a$. The guarantee kernel of each is $\varnothing$, but their union is trivially true, thus having a guarantee kernel of Σ^ω.

As a safety property is just the complement of a guarantee property, we can characterise safety properties analogously[3]:

$$P \text{ is a safety property} \iff \underline{P^{\complement}} = P^{\complement}$$

Definition 6.2 (Safety closure)**.** The *safety closure* of a property P, written $\overline{P}$, is the set of all infinite traces which cannot be shown *not* to be in P by examining only a finite prefix:

$$\overline{P} \triangleq (\underline{P^{\complement}})^{\complement}$$

It follows that $\overline{P}$ is the smallest superset of P that is a safety property. Thus, we can restate our above characterisation of safety properties as:

$$P \text{ is a safety property} \iff \overline{P} = P$$

As the dual of the guarantee kernel, the safety closure is a *closure operator*. That is, it is monotone ($P \subseteq R$ implies $\overline{P} \subseteq \overline{R}$), it is idempotent ($\overline{\overline{P}} = \overline{P}$), and it is a superset of the original property ($\overline{P} \supseteq P$). This operator coincides with the closure operator used to define the topology in Alpern & Schneider [8]. Thus, as with Alpern & Schneider, safety properties are closed sets in a topology, and therefore guarantee properties are the open sets, and the guarantee kernel is the *interior operator* for the topology.

Theorem 6.2 (The space of properties is metrisable)**.** *Let P be a property, with the metric:*

$$d(t,u) = \begin{cases} 0 & \text{if } t = u \\ 2^{-\sup\{\ell \mid \forall i < \ell.\, t_i = u_i\}} & \text{if } t \neq u \end{cases}$$

Then P is a guarantee property iff P is open in the metric topology, i.e. $\forall t \in P.\ \exists \varepsilon > 0.\ \{u \mid d(t,u) < \varepsilon\} \subseteq P$.

Proof. $\Longrightarrow$ Assume P is a guarantee property. Let $t \in P$. Then, because P is guarantee, there must exist some finite $p \in \downarrow t$ that is definitive for P, i.e. $\uparrow p \cap \Sigma^\omega \subseteq P$. Set $r = |p| - 1$ and $\varepsilon = 2^{-r}$. Then the metric ball of radius ε, i.e. $\{u \mid d(t,u) < \varepsilon\}$, is the set of all infinite extensions of p, i.e $\uparrow p \cap \Sigma^\omega$, because $d(t,u) < 2^{-r}$ iff t and u agree for a prefix of length $r+1$ — that is, p. Since t was arbitrary, this shows that $\forall t \in P.\ \exists \varepsilon > 0.\ \{u \mid d(t,u) < \varepsilon\} \subseteq P$ as required.

[3]Throughout this paper, the complement notation is only ever applied to *properties*, not definitive sets. So $P^{\complement} \triangleq \Sigma^\omega \setminus P$.

$\Longleftarrow$ Assume P is open in the metric topology, i.e. that $\forall t \in P.\ \exists \varepsilon > 0.\ \{u \mid d(t,u) < \varepsilon\} \subseteq P$. We shall show that P is a guarantee property, by showing that $P \subseteq \underline{P}$. Assume $t \in P$. Let $\varepsilon > 0$ be such that $\{u \mid d(t,u) < \varepsilon\} \subseteq P$. By the Archimedean property, there must be some natural number r such that $2^{-r} < \varepsilon$. The ball of radius 2^{-r}, i.e. $\{u \mid d(t,u) < 2^{-r}\}$ is therefore contained within the ball of radius ε, which, by our openness assumption, must in turn be contained within P. Because $d(t,u) < 2^{-r}$ iff t and u agree for a prefix of length $r+1$, let $p \in \downarrow t$ be a prefix of t of length $r+1$. Then the ball of radius 2^{-r} is exactly $\uparrow p \cap \Sigma^\omega$. Therefore, $\uparrow p \cap \Sigma^\omega \subseteq \{u \mid d(t,u) < \varepsilon\} \subseteq P$, i.e. $p \in \lightning P$. This shows, as t was arbitrary, that all $t \in P$ are the extension of some finite definitive $p \in \lightning P$, and therefore that $t \in \underline{P}$. Therefore $P \subseteq \underline{P}$ and P is a guarantee property. □

As a consequence, the collection of guarantee properties over Σ constitutes a topology: it contains $\varnothing$ and Σ^ω, and is closed under arbitrary unions and finite intersections.

6.1.2. Liveness and Morbidity

Liveness properties are those that can never be definitively refuted by a finite prefix. Thus a property P represents a liveness property iff all finite prefixes are prefixes of traces in P, i.e., $\Sigma^* \subseteq \downarrow P$. Equivalently, liveness properties are *dense*: P is a liveness property iff $\overline{P} = \Sigma^\omega$.

As morbidity properties are the complement of liveness properties, we can say P represents a morbidity property iff all finite prefixes are prefixes of traces in its complement, i.e., $\Sigma^* \subseteq \downarrow(P^\complement)$. Equivalently, morbidity properties have an empty interior: P is a morbidity property iff $\underline{P} = \varnothing$.

Theorem 6.3 (Alpern & Schneider, redux)**.** *Every property is the union of a guarantee and a morbidity property.*

Proof. For a given property P, we can decompose it straightforwardly as follows:

$$P = \underbrace{\underline{P}}_{\text{guarantee}} \cup \underbrace{(P \setminus \underline{P})}_{\text{morbidity}}$$

$\underline{P}$ is clearly a guarantee property. It remains to show that $P \setminus \underline{P}$ is a morbidity

property. We shall do this by showing that its interior is empty, i.e.:

$$\begin{aligned}
\underline{P \setminus \underline{P}} &= \underline{P \cap (\underline{P})^{\complement}} \\
&= \underline{\underline{P} \cap (\underline{P})^{\complement}} && \text{(Theorem 6.1)} \\
&\subseteq \underline{P} \cap (\underline{P})^{\complement} && \text{(g.k. is a subset)} \\
&= \varnothing
\end{aligned}$$

□

As we have proved that every property is the union of a guarantee and a morbidity property, we can then show that every property is the intersection of a safety and liveness property, the famous result of Alpern & Schneider [8], simply by taking the complement.

6.2. Monitorability

There are some properties for which LTL_3 will *always* give a `?` answer. For example, take the standard *request/acknowledge* format:

$$\varphi = \Box(r \Rightarrow \Diamond a)$$

which states that all requests (r) must eventually be acknowledged (a). For every finite prefix u, we have $ur^\omega \in [\![\varphi]\!]_3$ `F` and $ua^\omega \in [\![\varphi]\!]_3$ `T`. As the `F` and `T` answers are non-overlapping (Theorem 4.5), u cannot be a definitive prefix. Therefore, *all* finite prefixes are not definitive, so no run-time monitor will ever be useful for this property. In order for run-time monitoring to be useful, temporal properties must be *monitorable*. Exactly what *monitorable* means, however, is a murky question, and several possible answers exist in the literature. We formally characterise and compare several of these definitions below. Our findings are summarised in Figure 6. In this diagram, an arrow indicates a *strict* inequality. For example, all safety properties are bad monitorable (see Theorem 6.4), but not all bad monitorable properties are safety properties.

6.2.1. Good and bad monitorability

Kupferman & Vardi [13] define the *bad prefixes* of a property $P \subseteq \Sigma^\omega$ as those prefixes that cannot be extended to a trace that is in P, and further define *good prefixes* as those for whom all infinite extensions are in P. In our notation, the good prefixes of P are exactly the definitive prefixes $\lightning P$, and the bad prefixes are the definitive prefixes of the complement $\lightning P^{\complement}$. From this, we define a property P as *good monitorable* iff there exists a good finite prefix of P, *or* there are no traces in P. Similarly, a property P is *bad monitorable* iff there exists a bad finite prefix of P, *or* if every trace is in P.

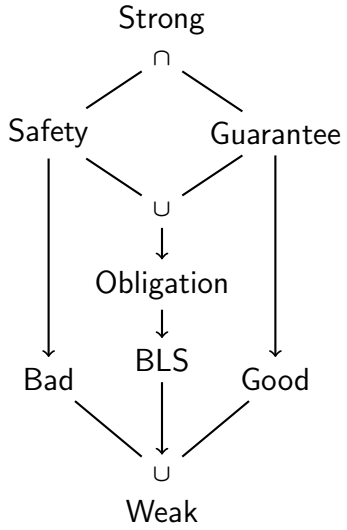

Figure 6: The monitorability lattice

Definition 6.3 (Good and bad monitorability)**.**

$$P \text{ is good monitorable} \longleftrightarrow \not\downarrow P \cap \Sigma^* \neq \varnothing \vee P = \varnothing \longleftrightarrow \underline{P} \neq \varnothing \vee P = \varnothing$$

$$P \text{ is bad monitorable} \longleftrightarrow \not\downarrow P^{\complement} \cap \Sigma^* \neq \varnothing \vee P = \Sigma^\omega \longleftrightarrow \overline{P} \neq \Sigma^\omega \vee P = \Sigma^\omega$$

Theorem 6.4. *All safety properties are bad monitorable, and all guarantee properties are good monitorable.*

Proof. Consider a guarantee property P. If $P = \varnothing$ then it is trivially good monitorable. Otherwise, let t be an infinite trace in P. Because P is a guarantee property, $\underline{P} = P$, and thus t must be the extension of some finite good prefix $p \in \not\downarrow P$. Thus P is good monitorable. For safety properties and bad monitorability, we take the complement. □

6.2.2. Weak monitorability

Next we define the largest class of properties for which monitors can be of any use, *weak monitorability*. A property is weak monitorable if there exists at least one definitive prefix, good or bad. This definition of monitorability is often attributed to Pnueli & Zaks [9], and is also called "weak monitorability" by Chen et al. [22].

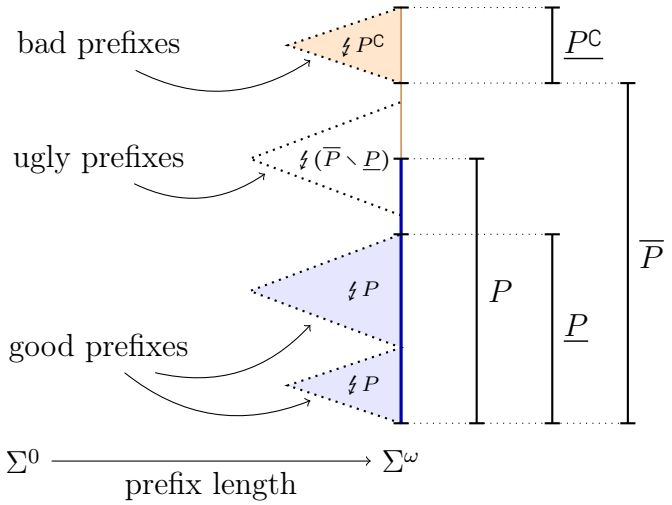

Figure 7: The connection between Kupferman & Vardi and Bauer's prefix characterisations and our kernel and closure operators

Definition 6.4. Weak Monitorability

$$P \text{ is weak monitorable} \iff \underline{P} \cup \underline{P^{\complement}} \neq \varnothing \iff \overline{P} \setminus \underline{P} \neq \Sigma^\omega$$

We call the set $\overline{P} \setminus \underline{P}$ the *unmonitorable traces* (topologically speaking, the *frontier*) of P, because these are precisely those infinite traces for which definitive answers cannot be given after observing only a finite prefix.

As can be seen in Figure 6, the class of weak monitorable properties is just the union of the classes of good monitorable and bad monitorable properties.

Weak monitorability is closed under negation, but not union or intersection. For example, take properties $P = \Box\Diamond b \vee \bigcirc a$ and $Q = \Box\Diamond b \vee \bigcirc(\neg a)$. Both P and Q are weak monitorable, but $P \wedge Q$ is not monitorable.

6.2.3. Ugly prefixes

Bauer et al. [6] further define *ugly prefixes* as those finite prefixes that cannot be finitely extended into good nor bad prefixes. Note that the good, bad, and ugly prefixes do not constitute a complete classification of all finite prefixes. For example, the finite prefix $ppp\ldots$ is neither good nor bad for $p \,\mathcal{U}\, q$, but it is not ugly either, as it can be extended with q giving a good prefix, or with $\varnothing$ giving a bad prefix. Here, $\varnothing$ is the state satisfying neither p nor q.

Theorem 6.5 (Frontier view of ugly traces)**.** *The ugly prefixes of a property P are the finite definitive prefixes of the unmonitorable traces of P:*

$$t \text{ is ugly} \longleftrightarrow t \in \not\downarrow(\overline{P} \setminus \underline{P}) \cap \Sigma^*$$

Proof. Showing each direction separately:

$\Longrightarrow$ If t is ugly then all finite extensions of t are not definitive for P or $P^\complement$. Let w be any infinite extension of t. If $w \in \underline{P} \cup \underline{P^\complement}$, then there must be some finite prefix of w that is definitive for P or for $P^\complement$. As such a prefix cannot exist, w is in neither $\underline{P}$ nor $\underline{P^\complement}$. As w is arbitrary, all infinite extensions of t are in $(\underline{P} \cup \underline{P^\complement})^\complement = \overline{P} \setminus \underline{P}$. Thus $t \in \not\downarrow(\overline{P} \setminus \underline{P}) \cap \Sigma^*$.

$\Longleftarrow$ Given a finite prefix $t \in \not\downarrow(\overline{P} \setminus \underline{P})$, we must show that t is ugly. Let $u \in \uparrow t$ be any finite extension of t. As definitive sets are closed under extension, $u \in \not\downarrow(\overline{P} \setminus \underline{P})$. As the guarantee kernel of P, the guarantee kernel of $P^\complement$, and the frontier $\overline{P} \setminus \underline{P}$ are all disjoint, and u is definitive for the frontier, u cannot be definitive for P (i.e. good) nor for $P^\complement$ (i.e bad). Therefore, t has no good nor bad finite extensions, and thus t is an ugly prefix. □

6.2.4. BLS monitorability

Bauer et al. [6] defines P as monitorable iff there are no (finite) ugly prefixes. Intuitively, if a property is BLS (Bauer-Leucker-Schallhart) monitorable, then no matter what prefix has already been observed, a definitive answer is always possible, in the sense of a liveness property. Due to Theorem 6.5, this definition is equivalent to the following:

Definition 6.5 (BLS monitorability)**.**

$$P \text{ is BLS monitorable} \longleftrightarrow \underline{\overline{P} \setminus \underline{P}} = \varnothing$$

Equivalently, P is BLS monitorable iff the unmonitorable traces of P are a morbidity property. Topologically, P is BLS monitorable iff the *frontier* of P has an *empty interior*.[4] This topological characterisation was first established by Diekert and Leucker [23], and is independently reconstructed here.

[4]Because the frontier of P is always closed, we could here say "nowhere dense" instead of "empty interior".

Theorem 6.6. *Every BLS monitorable property is weak monitorable.*

Proof. Let P be BLS monitorable, i.e., $\underline{\overline{P} \setminus \underline{P}} = \varnothing$. If P were not weak monitorable, then $\overline{P} \setminus \underline{P} = \Sigma^\omega$. As $\underline{\Sigma^\omega} = \Sigma^\omega \neq \varnothing$, this is a contradiction. Thus P must be weak monitorable. □

Counterexample 6.2 (Weak $\supsetneq$ BLS)**.** Consider $\Box\Diamond a \vee \bigcirc b$. The prefix *bbbb* is good, so this property is weak monitorable, however it is not BLS monitorable as the prefix *aaaa* is ugly.

Theorem 6.7 (Closure properties for BLS monitorability)**.** *BLS monitorability is closed under complement, finite intersection and finite union.*

Proof. Complement: Let P be a BLS monitorable property, i.e. $\underline{\overline{P} \setminus \underline{P}} = \varnothing$. Then $\underline{\overline{P^{\mathsf{C}}} \setminus \underline{P^{\mathsf{C}}}} = \underline{\underline{P^{\mathsf{C}}}^{\mathsf{C}} \cap \underline{P^{\mathsf{C}}}^{\mathsf{C}}} = \underline{\underline{P}^{\mathsf{C}} \cap \underline{P^{\mathsf{C}}}^{\mathsf{C}}} = \underline{\underline{P}^{\mathsf{C}} \cap \overline{P}} = \underline{\overline{P} \setminus \underline{P}} = \varnothing$.

Intersection: Let P and Q be BLS monitorable properties and let w be a finite prefix. We must show that w is not ugly for $P \cap Q$. In other words, we must show that w has a finite extension wu that is either good or bad for $P \cap Q$.

As P is BLS monitorable, w is not ugly for P. Hence there exists a finite extension wv that is either good or bad for P. If wv is bad for P, it is also bad for $P \cap Q$, and we can take $u \coloneqq v$ and be done. So assume wv is good for P. As Q is BLS monitorable, wv has a finite extension wvx which is either good or bad for Q. If wvx is bad for Q, then it is also bad for $P \cap Q$, and we can take $u \coloneqq vx$ and be done. So assume wvx is good for Q. As wvx is an extension of wv, it is also good for P. Thus it is good for $P \cap Q$ and we can take $u \coloneqq vx$.

Union: Because BLS monitorability is closed under intersection and complement, it is also closed under union by de Morgan's laws. □

Counterexample 6.3 (Failure of closure over infinite intersection)**.** Consider the following family of properties:

$$P_i = \{t \in \Sigma^\omega \mid t \text{ contains at least } i \text{ states labelled } a\}$$

Each P_i is BLS monitorable, as appending i a-states to any finite prefix will yield a good prefix. The infinite intersection $\bigcap_i P_i$, however, requires an a-state to occur infinitely often, which is not BLS monitorable.

Theorem 6.8. *All safety and guarantee properties are BLS monitorable.*[5]

Proof. Let P be a safety property, i.e. $\overline{P} = P$. Then

$$\begin{aligned}
\overline{\underline{P} \setminus \underline{P}} &= \underline{P \setminus \underline{P}} && (P \text{ is safety prop.}) \\
&= \underline{P \cap \underline{P}^{\mathsf{C}}} \\
&= \underline{P} \cap \underline{\underline{P}^{\mathsf{C}}} && \text{(Theorem 6.1)} \\
&\subseteq \underline{P} \cap \underline{P}^{\mathsf{C}} && \text{(g.k. is subset)} \\
&= \varnothing
\end{aligned}$$

For guarantee properties, we take the complement. □

Counterexample 6.4 ($\mathsf{BLS} \supsetneq \mathsf{Safety} \vee \mathsf{Guarantee}$)**.** Consider the following property X, originally due to Bauer et al. [6]:

$$(p \vee q)\ \mathcal{U}\ r \vee \Box p$$

This property is neither a safety property (the trace q^ω violates X but $q^\omega \in \overline{X}$, i.e. this violation is not observable from a finite prefix) nor a guarantee property (the trace p^ω satisfies X but $p^\omega \notin \underline{X}$, i.e. this satisfaction is not observable from a finite prefix). However, this property is BLS monitorable, as any finite trace can be extended to one that satisfies or violates the property.

6.2.5. Obligation Properties

Counterexample 6.4 above is neither a safety nor a guarantee property, but it is the *union* of a safety and a guarantee property. Manna & Pnueli [24] define *obligation properties* as Boolean combinations of safety and guarantee properties:

Definition 6.6 (Obligation Properties, per [24])**.** Inductively:

- All guarantee properties are obligation properties.
- All safety properties are obligation properties.
- If A and B are obligation properties, then so is $A \cap B$.

[5]This also follows from a well known theorem about Hausdorff spaces that the frontier of any closed or open set has an empty interior.

- If A and B are obligation properties, then so is $A \cup B$.

Theorem 6.9. *All obligation properties are BLS monitorable.*

Proof. Corollary of Theorem 6.8 and our closure properties in Theorem 6.7. □

Counterexample 6.5 (BLS $\supsetneq$ Obligation)**.** Let X be any property over $\Sigma = \{a, b\}$ that is not an obligation property. X' is the property over $\Sigma = \{a, b, c\}$ given by:

$$X' = X \cup \{t \mid c \text{ occurs in } t\}$$

X' is BLS monitorable, as any finite prefix w can be extended with just one c to give a good prefix for X' and is therefore non-ugly. But, X' cannot be an obligation property, because if it were, we could show that $X' \setminus \{t \mid c \text{ occurs in } t\} = X$ is an obligation property, which is a contradiction.

6.2.6. Strong monitorability

BLS monitorability allows for a property to have unmonitorable traces, but they must not be inevitable from any finite prefix. We could imagine a stronger kind of monitorability, which demands that the property contains no unmonitorable traces at all:

Definition 6.7 (Strong monitorability)**.**

$$P \text{ is strong monitorable} \iff \underline{P} \cup \underline{P^C} = \Sigma^\omega \iff \overline{P} \setminus \underline{P} = \varnothing$$

Topologically, the strong monitorable properties are the sets with an empty frontier. Intuitively, a property P is strong monitorable if for *all* traces that satisfy or violate P, the satisfaction or violation of P can be determined after only a finite observation. This corresponds to the *complete monitorability* of Aceto et al. [25].

Theorem 6.10. *Strong monitorable properties are the intersection of safety properties and guarantee properties.*[6]

[6] This is an instance of the general topology theorem that all sets with an empty frontier are clopen.

Proof. Let P be a strong monitorable property, that is, $\overline{P} \setminus \underline{P} = \varnothing$. Then $\overline{P} = \overline{P} \setminus \underline{P} \cup \underline{P} = \underline{P}$. As $\underline{P} \subseteq P \subseteq \overline{P}$, this means that $\overline{P} = \underline{P} = P$. Therefore P is both a safety and guarantee property. Similarly let P be both a safety and a guarantee property, i.e. $\overline{P} = \underline{P} = P$. Therefore clearly $\overline{P} \setminus \underline{P} = \varnothing$, and thus P is strong monitorable. □

As a corollary, strong monitorable properties are closed under complement, intersection, and union.

7. Related Work

7.1. Finfinite Semantics

Aceto et al. [26] define monitoring systems essentially as two extension-closed predicates on Σ^∞, **acc** and **rej**. Because they are extension closed, judgements are *irrevocable*, in the sense that if a monitor **acc**epts a finite trace, it must **acc**ept all extensions of that trace (and similarly for **rej**). This is highly reminiscent of the two sets we use for our LTL_3 semantics (which are definitive and therefore also extension-closed), however these monitors are in the context of *finfinite semantics*, where properties can consist of both finite and infinite traces. Falcone et al. [27] provide definitions of monitorability in a finfinite setting which they claim correspond to safety and guarantee properties, but, as Aceto et al. [26] show, this claim is false. Aceto et al. [26] then present a hierarchy of monitorability classes that, at first glance, highly resembles that of Figure 6, but finfinite semantics differ in significant ways from our infinite trace semantics. In a finfinite context, the property $\bigcirc\bot$, using a weak interpretation of the $\bigcirc$ operator, is not empty. Similarly, the property $\bigcirc\top$, using a strong interpretation of the $\bigcirc$ operator, is not trivially true. Under finfinite semantics, monitors cannot make the progress assumption that a given finite observation will eventually be extended with another state. This significantly changes the characteristics of our monitorability classes. For example, as Aceto et al. [25] show, under finfinite semantics the only strong monitorable properties (what Aceto et al. refer to as *complete monitorability*) are $\varnothing$ and Σ^∞, but in infinite semantics there are non-trivial strong monitorable properties. Similarly, as noted by Aceto et al. [26], when Σ contains only one state σ, the property $\{\sigma^\omega\}$, which would be trivially true and thus strong monitorable in our infinite semantics, is not even weak monitorable (what Aceto et al. [26] call $\exists PZ$*-monitorable*) under finfinite semantics. We conjecture that finfinite semantics form another dimension to

our lattice in Figure 6, where the finfinite semantics classes are generally strictly smaller than the infinite semantics classes, but we leave detailed examination of finfinite semantics for future work.

7.2. Syntactic Treatments

The topological characterisations developed in this work are purely semantic and apply to any property $L \subseteq \Sigma^\omega$. A natural progression for future work is to consider the syntactic characterisation of these monitorable fragments when restricted to the class of ω-regular languages, analogous to the characterisation of the safety-progress hierarchy classes in LTL [24] or the work of Aceto et al. [25] on monitorability over finfinite semantics using subsets of the modal μ-calculus.

Such syntactic treatments would facilitate the development of algorithms for the effective construction of monitors for specific formulae, and would allow for evaluating the completeness of existing symbolic monitoring algorithms against our semantic definitions. Early investigations into monitor construction for LTL used coinductive techniques [28] and term-rewriting [29] to bridge the gap between finite observations and infinite requirements. More recently, Francalanza and Cini [30] introduced a local proof system for explainable monitoring based on dual-sided proof relations ($s \vdash^+ \varphi$ and $s \vdash^- \varphi$), which have a clear semantic analogue in our answer-indexed families. Such symbolic approaches have also been extended to real-time properties in Metric Temporal Logic [31].

7.3. Alternative Property Classes

Havelund & Peled [32] provide a different set of property classes to the conventional hierarchy. They define six classes, AFR, AFS, SFR, SFS, NFR, and NFS, indicating if the property is Always, Sometimes, or Never Finitely Refutable or Satisfiable. Safety properties coincide with AFR, guarantee properties with AFS, liveness properties with NFR, and morbidity properties with NFS. Good monitorable properties are SFS$\cup$AFS, and bad monitorable properties are SFR$\cup$AFR. The classes of weak monitorable properties and of strong monitorable properties are both Boolean combinations of other classes, so they can also be expressed in terms of Havelund & Peled's framework similarly. BLS monitorability does not cleanly map into Havelund & Peled's framework, and indeed they show that several of their classes contain both BLS monitorable and BLS unmonitorable properties. Havelund & Peled do not characterise any monitorability class other than BLS monitorability.

7.4. Revocable Semantics

As previously mentioned, all *obligation* properties [24] are BLS monitorable. If we move further up the safety-progress hierarchy [24, 33], however, we quickly encounter completely unmonitorable properties, such as our *request/acknowledge* format example $\Box(r \Rightarrow \Diamond a)$.

Bauer et al. [34, 6] propose relaxing the irrevocability requirement of LTL_3 by giving a dialect of LTL, called RV-LTL, where answers may be changed as the observed prefix is extended with more states. This allows answers to be given even for unmonitorable properties, although with no guarantees of soundness for presumptive answers. More precisely, RV-LTL is an ad-hoc layering of LTL_3 on top of Pnueli's LTL for finite traces (here notated $\models_{\mathsf{F}}$). Where LTL_3 would give the ? answer, RV-LTL instead gives a *presumptive* answer ($\top^{\mathsf{p}}$ or $\bot^{\mathsf{p}}$) based on the answer obtained from Pnueli's finite LTL:

$$[u \models \varphi]_{\mathsf{RV}} = \begin{cases} \top & \text{if } [u \models \varphi]_3 = \top \\ \bot & \text{if } [u \models \varphi]_3 = \bot \\ \top^{\mathsf{p}} & \text{if } [u \models \varphi]_3 = ? \text{ and } u \models_{\mathsf{F}} \varphi \\ \bot^{\mathsf{p}} & \text{if } [u \models \varphi]_3 = ? \text{ and } u \not\models_{\mathsf{F}} \varphi \end{cases}$$

Intuitively, after a finite prefix u, a definitive answer ($\top$ or $\bot$) is unchangeable no matter how the prefix is extended, whereas a presumptive answer ($\top^{\mathsf{p}}$ or $\bot^{\mathsf{p}}$) only applies if execution is stopped at that point.

If the property in question is a safety property, then the only presumptive answer possible is $\top^{\mathsf{p}}$, and likewise for guarantee properties and $\bot^{\mathsf{p}}$. This means that for properties at the bottom of the safety-progress hierarchy [24], LTL_3 is sufficient, as the single ? answer can be interpreted as $\top^{\mathsf{p}}$ or $\bot^{\mathsf{p}}$ respectively. However, as noted by Bauer et al. [6], for an obligation property such as Counterexample 6.4, which is neither a safety nor guarantee property, both $\top^{\mathsf{p}}$ and $\bot^{\mathsf{p}}$ answers are possible.

Like LTL_3 previously, the semantics of RV-LTL is presented only in terms of other logics. We believe that an inductive semantics can be designed along similar principles to that of LTL_3 given in the present paper, where our answer indexed-families instead produce four sets, two of which are definitive, rather than the two definitive sets we provide for LTL_3.

As noted by O'Connor & Wickström [17], Pnueli's finite LTL is a logic of finite *completed* traces, so the decision to judge partial traces as completed for the purpose of giving presumptive answers in RV-LTL is ad-hoc and

can produce seemingly arbitrary answers for properties higher in the safety-progress hierarchy. For example, consider a system where a flashing light consistently alternates between On and Off states:

$$\mathsf{On}\ \mathsf{Off}\ \mathsf{On}\ \mathsf{Off}\ \cdots$$

A simple property that we might wish to monitor for this system is that the light is On infinitely often:

$$\Box\Diamond\mathsf{On}$$

As this formula nests $\Box$ and $\Diamond$ operators, it is definitive in neither positive nor negative cases and will only give presumptive answers. But the presumptive answer given in RV-LTL depends only on the very last observed status of the light. For a trace where the light continuously alternates off and on, as above, we might intuitively say that presumptive answer ought to be true, but this formula would be considered presumptively false if our observation happens to end in a state where the light is off. Thus, the truth value obtained for this formula is rather sensitive to the point at which our finite observation ceases.

7.5. Branching Time Monitoring

The vast majority of research on monitoring and run-time verification concerns linear-time semantics, but we can also consider monitoring for branching-time properties. Aceto et al. [25] compare monitors for linear-time and branching-time semantics. Francalanza et al. [35, 36] provide an extensive investigation into branching time monitoring. Manolios & Trefler [37] and Bouajjani et al. [38] examine safety and liveness properties analogously to Alpern & Schneider [8] but in a branching-time setting. Branching-time analogues of our monitorability classes are future work.

8. Conclusion

We have presented a comprehensive semantic foundation for run-time verification with linear-time infinite trace semantics. By identifying those prefixes that are definitive for a property, we have given an inductive, model-based semantic accounting of LTL_3 in terms of answer-indexed families of definitive sets, and in the process shown that LTL_3 is more accurately described as a more detailed presentation of conventional LTL, rather than a distinct logic in its own right. We have formalised the popular formula

progression technique used in runtime verification and testing scenarios, and proved it sound and complete with respect to our semantics. Using definitive prefixes as a basis, we have elegantly characterised various classes of properties and their monitorability characteristics in a distinctly topological flavour, in the vein of Alpern & Schneider [8]. All of our work has been mechanised in over 2900 lines of Isabelle/HOL proof script.

Our theory of definitive prefixes might provide a semantic foundation for other logics of partial traces, such as the LTL$^{\pm}$ of Eisner et al. [39], QuickLTL from O'Connor & Wickström [17], or the aforementioned RV-LTL [6]. Our answer-indexed families may also be applicable to other multi-valued logics. Examples include rLTL [40] and the five-valued logic of Chai et al. [41]. Our monitorability characterisations are likely to have analogues in other semantic contexts such as when monitoring for branching-time properties or hyperproperties.

References

[1] Z. Manna, A. Pnueli, The Temporal Logic of Reactive and Concurrent Systems, Springer, 1992. doi:10.1007/978-1-4612-0931-7.

[2] M. Y. Vardi, P. Wolper, Reasoning about infinite computations, Information and Computation 115 (1994) 1–37. doi:10.1006/inco.1994.1092.

[3] O. Lichtenstein, A. Pnueli, L. Zuck, The glory of the past, in: Logics of Programs, Springer, 1985, pp. 196–218. doi:10.1007/3-540-15648-8_16.

[4] Z. Manna, A. Pnueli, Temporal Verification of Reactive Systems: Safety, Springer, 1995. doi:10.1007/978-1-4612-4222-2.

[5] A. Bauer, M. Leucker, C. Schallhart, Runtime verification for LTL and TLTL, ACM Transactions on Software Engineering Methodology 20 (2011). doi:10.1145/2000799.2000800.

[6] A. Bauer, M. Leucker, C. Schallhart, Comparing LTL semantics for runtime verification, Journal of Logic and Computation 20 (2010) 651–674. doi:10.1093/logcom/exn075.

[7] A. Bauer, Y. Falcone, Decentralised LTL monitoring, in: FM 2012: Formal Methods, Springer, 2012, pp. 85–100. doi:10.1007/978-3-642-32759-9_10.

[8] B. Alpern, F. B. Schneider, Defining liveness, Information Processing Letters 21 (1985) 181–185. doi:10.1016/0020-0190(85)90056-0.

[9] A. Pnueli, A. Zaks, PSL model checking and run-time verification via testers, in: FM 2006: Formal Methods, Springer, 2006, pp. 573–586. doi:10.1007/11813040_38.

[10] R. Amjad, R. J. van Glabbeek, L. O'Connor, The infinite, in finite time, 2025. https://github.com/rayhanayasmin/the_infinite_in_finite_time, Formal proof development.

[11] R. Amjad, R. J. van Glabbeek, L. O'Connor, Definitive set semantics for LTL3, Archive of Formal Proofs (2024). https://isa-afp.org/entries/LTL3_Semantics.html, Formal proof development, not yet up to date with the repository linked above.

[12] R. Amjad, R. J. van Glabbeek, L. O'Connor, Semantics for linear-time temporal logic with finite observations, in: G. Caltais, C. Di Giusto (Eds.), Proceedings Combined 31st International Workshop on Expressiveness in Concurrencyand 21st Workshop on Structural Operational Semantics, Calgary, Canada, 9th September 2024, volume 412 of *Electronic Proceedings in Theoretical Computer Science*, Open Publishing Association, 2024, pp. 35–50. doi:10.4204/EPTCS.412.4.

[13] O. Kupferman, M. Y. Vardi, Model checking of safety properties, Formal Methods in System Design 19 (2001) 291–314. doi:10.1023/A:1011254632723.

[14] S. Blamey, Partial logic, in: Handbook of Philosophical Logic, Springer, 2002, pp. 261–353. doi:10.1007/978-94-017-0458-8_5.

[15] F. Bacchus, F. Kabanza, Using Temporal Logic to Control Search in a Forward Chaining Planner, IOS Press, 1996, p. 141–153. doi:10.1016/S0004-3702(99)00071-5.

[16] F. Kabanza, S. Thiébaux, Search control in planning for temporally extended goals, in: International Conference on Automated Planning and Scheduling, AAAI, 2005, pp. 130–139.

[17] L. O'Connor, O. Wickström, Quickstrom: Property-based acceptance testing with LTL specifications, in: Programming Language Design and Implementation, PLDI 2022, ACM, 2022, p. 1025–1038. doi:10.1145/3519939.3523728.

[18] G. Roşu, K. Havelund, Rewriting-based techniques for runtime verification, Automated Software Engineering 12 (2005) 151–197. doi:10.1007/s10515-005-6205-y.

[19] L. Lamport, Proving the correctness of multiprocess programs, IEEE Transactions on Software Engineering 3 (1977) 125–143. doi:10.1109/TSE.1977.229904.

[20] T. A. Henzinger, N. Mazzocchi, N. E. Saraç, Quantitative safety and liveness, in: Foundations of Software Science and Computation Structures, Springer, 2023, pp. 349–370. doi:10.1007/978-3-031-30829-1_17.

[21] R. J. van Glabbeek, The coarsest precongruences respecting safety and liveness properties, in: C. S. Calude, V. Sassone (Eds.), Proceedings 6th IFIP TC 1/WG 2.2 International Conference on Theoretical Computer Science (TCS'10); held as part of the *World Computer Congress* 2010, Brisbane, Australia, September 20-23, 2010, volume 323 of *IFIP*, Springer, 2010, pp. 32–52. doi:10.1007/978-3-642-15240-5_3.

[22] Z. Chen, Y. Wu, O. Wei, B. Sheng, Deciding weak monitorability for runtime verification, in: International Conference on Software Engineering: Companion Proceedings, ICSE '18, ACM, 2018, p. 163–164. doi:10.1145/3183440.3195077.

[23] V. Diekert, M. Leucker, Topology, monitorable properties and runtime verification, Theoretical Computer Science 537 (2014) 29–41. doi:10.1016/j.tcs.2014.02.052.

[24] Z. Manna, A. Pnueli, A hierarchy of temporal properties, in: Principles of Distributed Computing, ACM, 1990, p. 377–410. doi:10.1145/93385.93442.

[25] L. Aceto, A. Achilleos, A. Francalanza, A. Ingólfsdóttir, K. Lehtinen, Adventures in monitorability: from branching to linear time and back again, Proceedings of the ACM on Programming Languages 3 (2019) 1–29. doi:10.1145/3290365.

[26] L. Aceto, A. Achilleos, A. Francalanza, A. Ingólfsdóttir, K. Lehtinen, An operational guide to monitorability, in: Software Engineering and Formal Methods, Springer, 2019, pp. 433–453. doi:10.1007/978-3-030-30446-1_23.

[27] Y. Falcone, J.-C. Fernandez, L. Mounier, What can you verify and enforce at runtime?, International Journal on Software Tools for Technology Transfer 14 (2012) 349–382. doi:10.1007/s10009-011-0196-8.

[28] K. Sen, G. Rosu, G. Agha, Generating optimal linear temporal logic monitors by coinduction, in: Proceedings of the 8th Asian Computing Science Conference on Advances in Computing Science, (ASIAN'03), Springer, 2003, pp. 260–275. doi:10.1007/978-3-540-40965-6_17.

[29] M. Geilen, On the Construction of Monitors for Temporal Logic Properties, in: Runtime Verification, volume 55 of *Electronic Notes in Theoretical Computer Science*, Elsevier, 2001, pp. 181–199. doi:10.1016/s1571-0661(04)00252-x.

[30] A. Francalanza, C. Cini, Computer says no: Verdict explainability for runtime monitors using a local proof system, Journal of Logical and Algebraic Methods in Programming 119 (2021). doi:10.1016/j.jlamp.2020.100636.

[31] L. Lima, A. Herasimau, M. Raszyk, D. Traytel, S. Yuan, Explainable online monitoring of metric temporal logic, in: Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2023, pp. 473–491. doi:10.1007/978-3-031-30820-8_28.

[32] K. Havelund, D. Peled, Runtime verification: From propositional to first-order temporal logic, in: C. Colombo, M. Leucker (Eds.), Runtime Verification, Springer, 2018, pp. 90–112. doi:10.1007/978-3-030-03769-7_7.

[33] E. Chang, Z. Manna, A. Pnueli, The safety-progress classification, in: Logic and Algebra of Specification, Springer, 1993, pp. 143–202. doi:10.1007/978-3-642-58041-3_5.

[34] A. Bauer, M. Leucker, C. Schallhart, The good, the bad, and the ugly, but how ugly is ugly?, in: Runtime Verification, Springer, 2007, pp. 126–138. doi:10.1007/978-3-540-77395-5_11.

[35] A. Francalanza, L. Aceto, A. Achilleos, D. P. Attard, I. Cassar, D. Della Monica, A. Ingólfsdóttir, A foundation for runtime monitoring, in: S. Lahiri, G. Reger (Eds.), Runtime Verification, Springer, 2017, pp. 8–29. doi:10.1007/978-3-319-67531-2_2.

[36] A. Francalanza, L. Aceto, A. Ingolfsdottir, Monitorability for the Hennessy–Milner logic with recursion, Formal Methods in System Design 51 (2017) 87–116. doi:10.1007/s10703-017-0273-z.

[37] P. Manolios, R. Trefler, Safety and liveness in branching time, in: Symposium on Logic in Computer Science, IEEE Computer Society, 2001, p. 0366. doi:10.1109/LICS.2001.932512.

[38] A. Bouajjani, J. C. Fernandez, S. Graf, C. Rodriguez, J. Sifakis, Safety for branching time semantics, in: Automata, Languages and Programming, Springer, 1991, pp. 76–92. doi:10.1007/3-540-54233-7_126.

[39] C. Eisner, D. Fisman, J. Havlicek, Y. Lustig, A. McIsaac, D. Van Campenhout, Reasoning with temporal logic on truncated paths, in: Computer Aided Verification, Springer, 2003, pp. 27–39. doi:10.1007/978-3-540-45069-6_3.

[40] P. Tabuada, D. Neider, Robust Linear Temporal Logic, in: 25th EACSL Annual Conference on Computer Science Logic, CSL 2016, volume 62 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CSL.2016.10.

[41] M. Chai, B.-H. Schlingloff, Online monitoring of distributed systems with a five-valued LTL, in: IEEE 44th International Symposium on Multiple-Valued Logic, 2014, pp. 226–231. doi:10.1109/ISMVL.2014.47.