A Lattice-Theoretic Characterization of Safety and Liveness

Panagiotis Manolios
Georgia Institute of Technology
College of Computing, CERCS Lab
801 Atlantic Drive
Atlanta, Georgia, 30332, USA
http://www.cc.gatech.edu/~manolios
manolios@cc.gatech.edu

Richard Trefler
University of Waterloo
School of Computer Science
200 University Avenue West
Waterloo, Ontario, N2L 3G1, Canada
http://se.uwaterloo.ca/~trefler
trefler@cs.uwaterloo.ca

ABSTRACT

The distinction between safety and liveness properties is due to Lamport who gave the following informal characterization. Safety properties assert that nothing bad ever happens while liveness properties assert that something good happens eventually. In a well-known paper Alpern and Schneider gave a topological characterization of safety and liveness for the linear time framework. Gumm has stated these notions in the more abstract setting of $\vee$-complete Boolean algebras. Recently, we characterized safety and liveness for the branching time framework and found that neither the topological characterization nor Gumm's characterization were general enough for our needs. We present a lattice-theoretic characterization that allows us to unify previous results on safety and liveness, including the results for the linear time and branching time frameworks and for ω-regular string and tree languages.

Categories and Subject Descriptors

F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs

1. INTRODUCTION

Reactive systems are non-terminating systems engaged in ongoing interaction with an environment [9]; examples include network protocols, operating systems, on-board controllers, cache coherence protocols, distributed databases, etc. In a seminal paper, Lamport grouped linear time properties of reactive systems into three categories: safety properties, liveness properties, and properties that are neither [13]. Informally, safety properties assert that nothing bad ever happens, whereas liveness properties assert that something good happens eventually.

Understanding the distinction between safety and liveness —which now appears in introductory textbooks on distributed computing— has been an ongoing research endeavor that has yielded numerous results, some of which we now recount.

Alpern and Schneider gave a formal semantic characterization of safety and liveness in the linear time framework, where properties and the semantics of programs are sets of infinite strings [3]. They gave a topological characterization where safety properties are closed sets and liveness properties are dense sets and they proved that every linear time property can be expressed as the conjunction of a liveness property and a safety property. In a subsequent paper they showed that given a Büchi automaton, $\mathcal{B}$, over infinite strings (which recognizes an ω-regular language), it is possible to decompose $\mathcal{B}$ into automata $\mathcal{B}_S$ and $\mathcal{B}_L$ such that the set of strings accepted by $\mathcal{B}_S$ is a safety property, the set of strings accepted by $\mathcal{B}_L$ is a liveness property, and the set of strings accepted by $\mathcal{B}$ is equal to the intersection of the set of strings accepted by $\mathcal{B}_S$ and $\mathcal{B}_L$ [4].

Sistla characterized safety and liveness for temporal logic formulas [21], an important problem due to the wide use of temporal logic (which was proposed for specifying and verifying reactive systems by Pnueli [17]).

Recently, we have extended the Alpern and Schneider linear time characterization of safety and liveness properties to branching time, where properties and the semantics of programs are sets of infinite trees [14]. The branching time framework includes process algebra [15] and temporal logics such as CTL [5] (which is used by many model checkers and is of great practical importance), CTL* [7], and the μ-calculus [11]. We showed that any branching time property can be decomposed into the conjunction of a safety and liveness property. In the case where a set of trees is given implicitly as a Rabin tree automaton, $\mathcal{B}$, we showed that it is possible to compute the Rabin automata corresponding to the closures of the language of $\mathcal{B}$. This allows us to effectively compute $\mathcal{B}_S$ and $\mathcal{B}_L$ such that the language of $\mathcal{B}_S$ is safe, the language of $\mathcal{B}_L$ is live, and the language of $\mathcal{B}$ is the intersection of the languages of $\mathcal{B}_S$ and $\mathcal{B}_L$.

The distinction between safety and liveness plays an important role in the design and analysis of reactive systems, as the proof methods employed to check safety properties differ

PODC 2003, July 13-16, 2003, Boston, Massachusetts, USA.

from those used to check liveness properties. For example, proofs of liveness properties frequently require the construction of well-founded relations —to show that the desired behavior cannot be avoided forever— while safety properties are usually proved by induction on the transition relation — to show that the prohibited behavior is never exhibited [16]. In addition, automatic proof techniques available for safety properties often do not work with liveness properties. For example, in some infinite state systems one can automatically determine if a safety property can be violated, whereas the existence of a fair computation cannot be so determined [2]. In the context of model checking [5, 18], Kupferman and Vardi show that proving safety properties is simpler than establishing general temporal logic properties [12].

The distinction between safety and liveness also has consequences for security properties, as Schneider argues that enforceable security properties correspond to safety properties and that security automata, which can be used to enforce security policies, correspond to Büchi automata that accept safe languages [20].

A lattice theoretic characterization of safety and liveness was given by Gumm [8]. Gumm shows that given a $\vee$-preserving map between complete Boolean algebras, one can derive the Alpern and Schneider theorem characterizing safety and liveness. This generalization was not motivated by any practical problem; in a survey of safety and liveness Kindler [10] says of Gumm's work "it remains an open question whether ... his characterization has practical relevance".

However, there is a practical consequence, which we discovered when we defined safety and liveness for branching time logics. In the branching time framework, our semantic characterization does not give rise to a topology, but it does fit in Gumm's framework.

But, there is a problem with Gumm's framework. If we consider the lattice of Büchi automata definable languages, where meet corresponds to intersection, join to union, and complementation to negation, then we have a Boolean algebra that is not $\vee$-complete, so it does not fit in Gumm's framework. Similarly, the lattice of Rabin tree automata definable languages forms a Boolean algebra that also does not satisfy Gumm's conditions. Therefore, we cannot appeal to the Alpern and Schneider theorem or any of its variants to establish that an ω-regular language (over strings or trees) can be expressed as the intersection of a safe and live language.

In this paper we present a simpler and more general characterization of safety and liveness. As a consequence, we further clarify the distinction between safety and liveness and provide more powerful tools for analyzing these concepts.

1. Our results allow us to simplify previous proofs. For example, the characterization of safety and liveness for Büchi automata and the decomposition theorem, the main results in [4], now follow directly from the results in this paper. Similarly, many of our proofs on safety and liveness in the branching time logic and for Rabin tree automata now follow directly from the results in this paper [14].

2. We carefully analyze the conditions required to prove the decomposition theorem for safety and liveness. We discuss why each of the conditions in our proofs is required, often by providing counterexamples. Such considerations led us to a lattice-theoretic approach, where the basic results are stated in terms of modular, complemented lattices.

3. Our lattice-theoretic framework is more general, therefore simpler than topological framework, *e.g.*, we do not require closure operators to distribute over joins (unions), thus, not only are our results applicable in more contexts, but they are easier to apply, as one needs to establish fewer properties than were previously required.

In the next section, we review the Alpern and Schneider characterization of safety and liveness for the linear time framework and we consider examples due to Rem [19]. In Section 3 we give our lattice-theoretic characterization and show how it applies to the linear time case. In Section 4 we give our branching time characterization and relate it to the lattice-theoretic view. Conclusions are given in Section 5.

2. LINEAR TIME FRAMEWORK

2.1 Preliminaries

$\mathbb{N}$ and ω both denote the natural numbers. Function application is sometimes denoted by an infix dot "." and is left associative. $\mathcal{P}(S)$ denotes the powerset of S. $Dom.f$ denotes the domain of function f. S^* denotes the set of finite sequences over S; S^ω denotes the set of infinite sequences (functions from ω) over S; $S^\infty = S^* \cup S^\omega$. Suppose $s, t \in S^\infty$, $|s|$ denotes the length of s; s is a prefix of t ($s \preceq t$) iff $Dom.s \subseteq Dom.t$ and for all $i \in Dom.s$, $s.i = t.i$; s is a proper prefix of t ($s \prec t$) iff $s \preceq t$ and $s \neq t$.

$\langle Qx : r : b\rangle$ denotes a quantified expression, where Q is the quantifier, x the bound variable, r the range of x (**true** if omitted), and b the body. We sometimes write $\langle Qx \in X : r : b\rangle$ as an abbreviation for $\langle Qx : x \in X \ \wedge\ r : b\rangle$, where r is **true** if omitted, as before. From highest to lowest binding power, we have: parentheses, function application, binary relations (*e.g.*, sBw), equality ($=$) and membership ($\in$), negation ($\neg$), conjunction ($\wedge$) and disjunction ($\vee$), implication ($\Rightarrow$), and finally, binary equivalence ($\equiv$). Spacing is used to reinforce binding: more space indicates lower binding.

We use a proof format that we exhibit with an example. Suppose that $\sqsubseteq$ is a preorder, then a proof $A \sqsubseteq Z$ is given in the following form.

Proof

A

$\sqsubseteq$ { Hint why $A \sqsubseteq B$ }

B

...

Z $\square$

The first three lines establish that $A \sqsubseteq B$, as per the hint. Lines three through five establish that B is related to the expression on line five of the proof by $\sqsubseteq$ (or any restriction of $\sqsubseteq$, usually $=$), and so on. Chaining these steps together, we conclude that $A \sqsubseteq Z$.

Throughout this paper Σ denotes a fixed *alphabet*, a non-empty set of symbols.

We assume familiarity with the Linear Temporal Logic (LTL) operators X (next-time, sometimes written $\bigcirc$), F (eventually, sometimes written $\Diamond$), G (always, sometimes written $\Box$) and the CTL and CTL* operators A (along all paths), and E (there exists a path). A good reference is [6].

2.2 Safety and Liveness

We review the main results of the Alpern and Schneider characterization for the linear time framework [3]. Our presentation differs from the original, *e.g.*, we use a closure operator, but the results are the same. The linear time closure operator over strings, $lcl : \mathcal{P}(\Sigma^\omega) \rightarrow \mathcal{P}(\Sigma^\omega)$, can be defined as follows.

$$lcl.T = \{t \in \Sigma^\omega \mid \langle \forall x : x \prec t : \langle \exists t' \in T :: x \preceq t' \rangle \rangle \}$$

We say that $cl : \mathcal{P}(X) \rightarrow \mathcal{P}(X)$ is a *topological-closure* operator on X if it satisfies the following four properties.

1. $cl.\emptyset \;=\; \emptyset$
2. $A \subseteq cl.A$
3. $cl(cl.A) \;=\; cl.A$
4. $cl(A \cup B) \;=\; cl.A \cup cl.B$

From a basic result of topology, a topological-closure operator on X defines a topology, where a set $A \subseteq X$ is closed iff $cl.A = A$.

It turns out that lcl is a topological-closure operator on Σ^ω. Safety properties are defined to be the closed sets and liveness properties are defined to be the dense sets. It then follows from topological considerations that any property P is the intersection of $lcl.P$, a safety property, and $P \cup \neg lcl.P$, a liveness property.

Lemma 1 *$P \cup \neg lcl.P$ is a liveness property.*

Proof

$\quad lcl(P \cup \neg lcl.P)$

$= \{$ lcl distributes over $\cup$ $\}$

$\quad lcl.P \;\cup\; lcl(\neg lcl.P)$

$\supseteq \{$ $lcl.A \supseteq A$, Set theory $\}$

$\quad lcl.P \;\cup\; \neg lcl.P$

$= \{$ Set theory $\}$

$\quad \Sigma^\omega \;\Box$

Theorem 1 *Any property can be decomposed into the intersection of a safety and liveness property.*

Proof

$\quad lcl.P \;\cap\; (P \cup \neg lcl.P)$

$= \{$ Set theory $\}$

$\quad (lcl.P \cap P) \;\cup\; (lcl.P \cap \neg lcl.P)$

$= \{$ $A \subseteq lcl.A$, Set theory $\}$

$\quad P \;\cup\; \emptyset$

$= \{$ Set theory $\}$

$\quad P \;\Box$

2.3 Examples

We now take a moment to consider some examples due to Martin Rem [19]. In the examples t is an infinite sequence over Σ.

$p0$:	**false** (corresponds to $\emptyset$)
$p1$:	the first symbol of t is a
$p2$:	the first symbol of t differs from a
$p3$:	the first symbol of t is a and t contains a symbol that differs from a
$p4$:	the number of a's in t is finite
$p5$:	the number of a's in t is infinite
$p6$:	**true** (corresponds to Σ^ω)

Below, we give our translation of the above properties to LTL. Recall that F means eventually (and is sometimes written $\Diamond$) and G means always (and is sometimes written $\Box$).

$p0$:	**false**
$p1$:	a
$p2$:	$\neg a$
$p3$:	$a \;\wedge\; \mathsf{F}\,\neg a$
$p4$:	$\mathsf{FG}\,\neg a$
$p5$:	$\mathsf{GF}\,a$
$p6$:	**true**

Note that $p0$, $p1$, $p2$, and $p6$ are safety properties. The closure of $p3$ is $p1$, so $p3$ is neither a safety property nor a liveness property. The closures of $p4$ and $p5$ are both Σ^ω; thus, they are liveness properties, but they are not safety properties.

2.4 Büchi Automata

In this section, we review the results in [4]. A Büchi automaton $\mathcal{B}$ is a 5-tuple $(\Sigma, Q, q_0, \delta, F)$ where Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is the start state, $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the transition relation, and F is a set of accepting states.

Let $t \in \Sigma^\omega$. A *run* of $\mathcal{B}$ on t is a Q-labeled sequence σ such that $\sigma.0 = q_0$ and for all $i \in \omega$, $\sigma(i+1) \in \delta(\sigma.i, t.i)$. Run σ is *accepting* iff some state of F occurs infinitely often in σ. $\mathcal{L}(\mathcal{B}) = \{t \in \Sigma^\omega \mid$ there is an accepting run of $\mathcal{B}$ on $t\}$ is the language of $\mathcal{B}$.

It is possible to decompose $\mathcal{B}$ into automata $\mathcal{B}_S$ and $\mathcal{B}_L$ such that the set of strings accepted by $\mathcal{B}_S$ is a safety property and the set of strings accepted by $\mathcal{B}_L$ is a liveness property [4]. Furthermore, the set of strings accepted by $\mathcal{B}$ is equal to the intersection of the set of strings accepted by $\mathcal{B}_S$ and $\mathcal{B}_L$, as before. The idea is to define a closure operator on Büchi automata. The operator first removes states that cannot reach an accepting state and then makes every remaining state an accepting state. In this way, the fairness condition is made trivial. It can then be shown that applying this operator to $\mathcal{B}$ results in an automaton whose language is the *lcl* of the language of $\mathcal{B}$. Since Büchi automata are closed under complementation and union, the liveness automaton is the union of $\mathcal{B}$ with the negation of the closure of $\mathcal{B}$, as in Theorem 1.

3. LATTICE THEORETIC CHARACTERIZATION

Recall that a lattice $\langle L, \leq \rangle$ consists of a set L and a partial order $\leq$ on L such that any pair of elements has a meet ($\wedge$) and join ($\vee$). From a more algebraic, but equivalent, viewpoint, a lattice is a triple $\langle L, \wedge, \vee \rangle$ where $\wedge$ and $\vee$ satisfy the associative, commutative, idempotency, and absorption laws.

$$\begin{array}{rrcl} \text{(associative law)} & (a \vee b) \vee c & = & a \vee (b \vee c) \\ \text{(commutative law)} & a \vee b & = & b \vee a \\ \text{(idempotency law)} & a \vee a & = & a \\ \text{(absorption law)} & a \vee (a \wedge b) & = & a \end{array}$$

Each law also has a dual, which is obtained by interchanging $\vee$ and $\wedge$. We can then define $a \leq b \;\equiv\; (a \wedge b) = a$ and it follows that $a \leq b \;\equiv\; (a \vee b) = b$. We will stick to the algebraic view. Note that in light of associativity and commutativity, we do not need parentheses for sequences of joins or meets.

Lemma 2

1. $a \leq b \quad \Rightarrow \quad a \vee c \;\leq\; b \vee c$

2. $a \leq b \quad \Rightarrow \quad a \wedge c \;\leq\; b \wedge c$

Proof

$a \vee c$

$\leq$ { Absorption ($x \leq x \vee y$), Associativity, Commutativity }

$a \vee c \vee b$

$=$ { $a \leq b$, hence $a \vee b \;=\; b$ }

$b \vee c$

The proof of (2) is similar. $\square$

A *lattice-closure* on L is a function $cl : L \to L$ such that the following hold.

1. $a \leq cl.a$
2. $cl.a = cl(cl.a)$
3. $a \leq b \quad \Rightarrow \quad cl.a \leq cl.b$

Lemma 3 $cl(a \vee b) \;\geq\; cl.a \vee cl.b$.

Proof

$cl(a \vee b)$

$=$ { $x \vee x \;=\; x$ }

$cl(a \vee b) \;\vee\; cl(a \vee b)$

$\geq$ { $a \vee b \geq a$, monotonicity of cl, Lemma 2 }

$cl.a \vee cl.b$ $\square$

A lattice has a *unit* element, 1, if $a \wedge 1 = a$ for all $a \in L$. A lattice has a *zero* element, 0, if $a \vee 0 = a$ for all $a \in L$. Suppose we have a lattice with a 0 and a 1, then b is a *complement* of a iff $b \wedge a = 0$ and $b \vee a = 1$. Note that b need not be unique. The set of complements of a is denoted $cmp.a$. A *complemented* lattice is one in which every element has a complement.

A lattice is *modular* if $a \geq c \quad \Rightarrow \quad a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$.

For the remainder of this section, unless we say otherwise, we assume that we are working with $\langle L, \wedge, \vee, 0, 1 \rangle$, a modular complemented lattice. Notice that a Boolean algebra is a special case of a modular complemented lattice.

A *cl-safety* element is one where $a = cl.a$. Note that $cl.a$ is a safety element (as $cl.a = cl(cl.a)$).

A *cl-liveness* element is one where $cl.a = 1$.

Lemma 4 *If $b \in cmp(cl.a)$ then $a \vee b$ is a cl-liveness element.*

Proof

$cl(a \vee b)$

$\geq$ { Lemma 3 }

$cl.a \;\vee\; cl.b$

$\geq$ { $cl.x \geq x$ }

$cl.a \;\vee\; b$

$=$ { $b \in cmp(cl.a)$ }

1 $\square$

Theorem 2 *If cl is a lattice closures, then every element in L is the meet of a cl-safety and cl-liveness element.*

Proof By Theorem 3 where $cl2, cl1 = cl$. $\square$

Theorem 2 is a simple corollary of Theorem 3, which will be useful when we consider branching time, and is presented next. The significance of Theorem 2 is that it can be used to obtain the Alpern Schneider results. To see why, note that $\langle \mathcal{P}(\Sigma^\omega), \cap, \cup, \emptyset, \Sigma^\omega, \neg \rangle$ is a Boolean algebra and *lcl* is a lattice-closure operator. Similarly the set of languages definable by Büchi automata is closed under union, intersection,

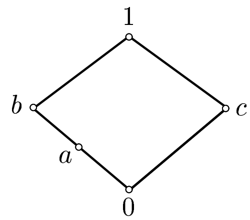

Figure 1: Why we need modularity in Theorem 3. The above Hasse diagram depicts a lattice. Circles denote elements and $x \leq y$ if there is an upward path from x to y. The lattice depicted is not modular: $b \geq a$ but $b \wedge (c \vee a)$ is b whereas $(b \wedge c) \vee (b \wedge a)$ is a. Consider the closure cl, where $cl.a = b$ and cl is the identity otherwise. Element a cannot be expressed as the conjunction of a cl-safety element and a cl-liveness element, as per Lemma 6.

and complementation so it too defines a Boolean algebra and we have shown that the closure operator on Büchi automata corresponds to lcl, so it too is a lattice-closure operator. Recall, that neither the original Alpern and Schneider topological characterization nor Gumm's characterization applies to the Büchi automata case. In fact, the main results of the paper by Alpern and Schneider on safety and liveness for Büchi automata follow directly from our results and basic closure properties of Büchi automata.

Theorem 3 *Suppose cl1 and cl2 are lattice closures such that for all $x \in L$, $cl1.x \leq cl2.x$. Then any element in L is the meet of a cl1-safety and a cl2-liveness element.*

Proof Suppose $a \in L$ and $b \in cmp(cl2.a)$.

$$\begin{aligned}
& cl1.a \ \wedge \ (a \vee b) \\
= \{ & \ cl1.a \geq a, \text{ Modularity } \} \\
& (cl1.a \wedge a) \ \vee \ (cl1.a \wedge b) \\
= \{ & \ a \leq cl1.a \ \} \\
& a \ \vee \ (cl1.a \wedge b) \\
= \{ & \ \text{Lemma 5 } (a, b, c \leftarrow cl1.a, cl2.a, b) \text{ and } x \vee 0 = x \ \} \\
& a \ \square
\end{aligned}$$

Lemma 5 $c \in cmp.b \ \wedge \ a \leq b \quad \Rightarrow \quad a \wedge c = 0$

Proof

$$\begin{aligned}
& a \wedge c \\
\leq \{ & \ a \leq b, \text{ Lemma 2 } \} \\
& b \wedge c \\
= \{ & \ c \in cmp.b \ \} \\
& 0 \ \square
\end{aligned}$$

We end this section by showing why modularity is needed. Consider the Hasse diagram in Figure 1. It corresponds to a lattice —where circles denote elements of the lattice and $x \leq y$ if there is an upward path from x to y— that is not modular. Consider the closure cl, where $cl.a = b$ and cl is the identity otherwise.

Lemma 6 *Element a of the (non-modular) lattice depicted in Figure 1 cannot be expressed as the conjunction of a cl-safety element and a cl-liveness element, where $cl.a = b$ and cl is the identity otherwise.*

Proof The only liveness element is 1, but since $x \wedge 1 = x$, $x \wedge 1 = a$ iff $x = a$ and a is not a safety element. $\square$

4. BRANCHING TIME FRAMEWORK

4.1 Preliminaries

For a relation R, we write $R|_S$ for R left-restricted to the set S, *i.e.*, $R|_S = \{\langle a, b\rangle \mid (\langle a, b\rangle \in R) \ \wedge \ (a \in S)\}$. $[i..j]$ denotes the set $\{k \in \mathbb{N} \mid i \leq k \leq j\}$. A set $U \subseteq S^\infty$ is *prefix-closed* iff for all $u \in U$ and for all $t \preceq u$, $t \in U$.

An *unlabeled tree* is a prefix-closed subset of $\mathbb{N}^*$. A *tree* w is a pair $\langle \mathsf{W}, \mathrm{w}\rangle$ where W is an unlabeled tree and $\mathrm{w} : \mathsf{W} \to \Sigma$. A tree $\langle \mathsf{W}, \mathrm{w}\rangle$ is *total* if $\mathsf{W} \neq \emptyset$ and for all $\sigma \in \mathsf{W}$, there exists $\rho \in \mathsf{W}$ such that $\sigma \prec \rho$. A tree $\langle \mathsf{W}, \mathrm{w}\rangle$ is *finite-depth* if there exists $n \in \mathbb{N}$ such that for all $\sigma \in \mathsf{W}$, $|\sigma| \leq n$. By A^{tot}, A^{nt}, and A^f we denote the set of total, non-total, and finite-depth trees, respectively. The set of trees is denoted by A^{all}; note $A^{all} = A^{tot} \cup A^{nt}$ and $A^f \subset A^{nt}$. Let $t = \langle \mathsf{W}, \mathrm{w}\rangle$ be a tree. A $p \subseteq \mathsf{W}$ is a *path* in t iff p is a totally ordered (by $\preceq$), prefix-closed subset of W. Given a tree $\langle \mathsf{W}, \mathrm{w}\rangle$ and a node $\sigma \in \mathsf{W}$ we define the path $\overline{\sigma} = \{\sigma' \in \mathsf{W} \mid \sigma' \preceq \sigma\}$. We extend w to paths: given path $p = p_0 p_1 \cdots$, $\mathrm{w}(p) = (\mathrm{w}.p_0)(\mathrm{w}.p_1)\cdots$.

We now define what it means to concatenate trees and use this notion to define what it means for one tree to be a prefix of another. The closure, safety elements, and liveness elements can then be defined as in the linear time case.

4.2 Safety and Liveness

Given trees w and x, we define a preliminary notion of tree concatenation, denoted $w \cdot x$.

Definition 1 *Let $w = \langle \mathsf{W}, \mathrm{w}\rangle$ and $x = \langle \mathsf{X}, \mathrm{x}\rangle$ be trees.* $w \cdot x = \langle \mathsf{W} \cup \mathsf{X}, \mathrm{w} \cup (\mathrm{x}|_{\mathsf{X} \setminus \mathsf{W}})\rangle$.

Note that $w \cdot x$ is a tree. The above notion of concatenation is not what we need. The problem is that it allows one to extend w at non-leaf nodes. To remedy this, we define an appropriate notion of concatenation after we define the notion of a leaf.

Definition 2 *Let $w = \langle \mathsf{W}, \mathrm{w}\rangle$ be a tree.* $leaf(z, w) \equiv z \in \mathsf{W} \ \wedge \ \neg\langle \exists y \in \mathsf{W} :: z \prec y\rangle$.

Given trees w and x, we define the notion of tree concatenation, denoted wx, as follows.

Definition 3 *Let $w = \langle \mathsf{W}, \mathrm{w}\rangle$ and $x = \langle \mathsf{X}, \mathrm{x}\rangle$ be trees. Let* $\mathsf{X}' = \{y \in \mathsf{X} \mid y \in \mathsf{W} \ \vee \ \langle \exists z : leaf(z, w) : z \prec y\rangle\}$. *Let* $x' = \langle \mathsf{X}', \mathrm{x}|_{\mathsf{X}'}\rangle$. $wx = w \cdot x'$.

Clearly, wx is a tree; the proof amounts to showing that x' is a tree. We now define when one tree is a prefix of another.

Definition 4 $x \sqsubseteq y \quad \equiv \quad \langle \exists z :: xz = y \rangle$

In [14], we show that $\sqsubseteq$ is a partial order and that if $x \sqsubseteq y$ then for all w, $wx \sqsubseteq wy$. $\mathcal{P}(A^{tot})$ is the set of branching time properties. Note that $\langle \mathcal{P}(A^{tot}), A^{tot}, \emptyset, \cup, \cap, \neg \rangle$ and $\langle \mathcal{P}(A^{all}), A^{all}, \emptyset, \cup, \cap, \neg \rangle$ are Boolean algebras. We define two closure operators on $\mathcal{P}(A^{tot})$, the non-total and finite-depth closures, as follows.

Definition 5 $ncl.p = \{ \ y \in A^{tot} \mid \langle \forall x \in A^{nt} : x \sqsubseteq y : \langle \exists z \in p :: x \sqsubseteq z \rangle \rangle \ \}$

Definition 6 $fcl.p = \{ \ y \in A^{tot} \mid \langle \forall x \in A^{f} : x \sqsubseteq y : \langle \exists z \in p :: x \sqsubseteq z \rangle \rangle \ \}$

The proofs that ncl, fcl are lattice-closures and that $ncl.p \subseteq fcl.p$ follow directly from results in [14].

Since we have two types of closures, we have two types of safety properties: existentially safe (ES) and universally safe (US). The intuition is that the existentially safe properties guarantee at least one computation along which nothing bad happens. The universally safe properties guarantee that nothing bad happens during any computation.

Definition 7 *(Existentially Safe)* $p \in ES \ \equiv \ p = ncl.p$

Definition 8 *(Universally Safe)* $p \in US \ \equiv \ p = fcl.p$

It turns out that fcl defines a topology, as $fcl.p \cup fcl.q = fcl.(p \cup q)$. However, $ncl.(p \cup q) \subseteq ncl.p \cup ncl.q$ is not a theorem, thus ncl does not define a topology.

Definition 9 *(Existentially Live)* $p \in EL \ \equiv \ ncl.p = A^{tot}$

Definition 10 *(Universally Live)* $p \in UL \ \equiv \ fcl.p = A^{tot}$

Theorem 4 *Every branching time property is the intersection of:*

1. *An existentially safe and an existentially live property;*
2. *A universally safe and a universally live property; and*
3. *An existentially safe and a universally live property.*

Proof By Theorem 3. $\square$

The above theorem shows that three of the four obvious ways of decomposing a property into the meet of a safety and liveness property hold. What about the fourth? The following theorem allows us to show that it does not hold.

Theorem 5 *Suppose cl1 and cl2 are closure operators such that $cl2.a = 1$ and $cl1.a < 1$. Then there do not exist s, l such that $cl2.s = s$, $cl1.l = 1$ and $a = s \wedge l$.*

Proof Suppose $cl2.s = s$, $cl1.l = 1$, and $a = s \wedge l$. Then:

$$
\begin{array}{ll}
& a = s \wedge l \\
\Rightarrow & \{ \text{ Def. of } \leq \ \} \\
& a \leq s \\
\Rightarrow & \{ \ cl2 \text{ is a lattice-closure } \} \\
& cl2.a \leq cl2.s \\
\Rightarrow & \{ \ cl2.a = 1 \ \} \\
& cl2.s = 1 \\
\Rightarrow & \{ \ s = cl2.s \ \} \\
& s = 1 \\
\Rightarrow & \{ \ a = s \wedge l \ \} \\
& a = l \\
\Rightarrow & \{ \ cl1.l = 1 \ \} \\
& cl1.a = 1 \\
\Rightarrow & \{ \ cl1.a < 1 \ \} \\
& \textbf{false} \ \square
\end{array}
$$

Since the set of trees satisfying the CTL formula $\mathsf{AF}p$ (along all paths, eventually p) satisfies the preconditions on the previous theorem, we have shown that not every property can be expressed as the intersection of a universally safe and an existentially live property.

Our decomposition of a property into a safety property and a liveness property is extreme in the following sense. (See [19] for a linear-time version of the extremal theorems.)

Theorem 6 *Suppose cl1 and cl2 are lattice closures such that for all $x \in L$, $cl1.x \leq cl2.x$. If $s = cl1.s$ or $s = cl2.s$ and $a = s \wedge z$, then $cl1.a \leq s$.*

Proof

$$
\begin{array}{ll}
& a = s \wedge z \\
\Rightarrow & \{ \text{ Def. of } \leq \ \} \\
& a \leq s \\
\Rightarrow & \{ \ cl1 \text{ is a lattice closure } \} \\
& cl1.a \leq cl1.s \\
\Rightarrow & \{ \ cl1.s \leq cl2.s, s = cl1.s \text{ or } s = cl2.s \ \} \\
& cl1.a \leq s \ \square
\end{array}
$$

Note that setting $cl1 = cl2$ gives us a version of the theorem applicable to cases where there is only one lattice closure operator, *e.g.*, in the linear time case, we set $cl1 = cl2 = lcl$.

Theorem 6 says that any decomposition of a into a $cl1$-safety or $cl2$-safety element and some other element (not necessarily a liveness element) requires that the safety element be weaker (*i.e.*, above or $\geq$) than $cl1.a$. Thus, $cl1.a$

is the strongest safety element that can be used in the decomposition of a. If we are thinking about specifications, this means that $cl1.a$ does as much of the specifying as possible and that x does not specify any safety property not already captured by $cl1.a$. Such properties are called "machine closed" [1]. In the context of automated verification it is often the case that safety properties are easier to check than liveness properties; the above theorem identifies the strongest safety property implied by a, which allows us to make the most of a checker for safety properties.

One may wonder if the liveness property in the decomposition above is also extreme. It is, if our lattice is also *distributive*, *i.e.*, it satisfies $a \wedge (b \vee c) \;=\; (a \wedge b) \vee (a \wedge c)$. (The difference is that we do not require $a \geq c$, as we did with modular lattices.) One can show that $\wedge$ distributes over $\vee$ —as in our definition of a distributive lattice— iff $\vee$ distributes over $\wedge$.

Theorem 7 *Suppose that $\langle L, \wedge, \vee, 0, 1\rangle$, is a distributive lattice and that cl1 and cl2 are lattice closures such that for all $x \in L$, $cl1.x \leq cl2.x$. If $s = cl1.s$ or $s = cl2.s$ and $a = s \wedge z$, then $z \leq a \vee b$, where $b \in cmp(cl1.a)$.*

Proof

$$a \vee b$$

$= \{$ Def. of a $\}$

$$(s \wedge z) \vee b$$

$\geq \{$ $cl1.a \leq s$ by Theorem 6 $\}$

$$(cl1.a \wedge z) \vee b$$

$= \{$ Distributivity $\}$

$$(cl1.a \vee b) \wedge (z \vee b)$$

$= \{$ $b \in cmp(cl1.a)$ $\}$

$$z \vee b$$

$\geq \{$ Def. of $\leq$ $\}$

$$z \; \square$$

As was the case with Theorem 6, setting $cl1 = cl2$ gives us a version of the theorem applicable to cases where there is only one lattice closure operator, *e.g.*, in the linear time case, we set $cl1 = cl2 = lcl$.

Theorem 7 says that any decomposition of a into a *cl1*-safety or *cl2*-safety element and some other element z (not necessarily a liveness element) requires that z be stronger (*i.e.*, below or $\leq$) than $a \vee b$, where $b \in cmp(cl1.a)$. Thus, $a \vee b$ is the weakest element that can be used in the decomposition of a. If we are thinking about specifications, this means that $a \vee b$ does as little of the specifying as possible. In the context of automated verification it is often the case that liveness properties are more difficult to check than safety properties; the above theorem identifies the weakest liveness property implied by a, which allows us to limit reasoning about liveness properties to the greatest extent possible.

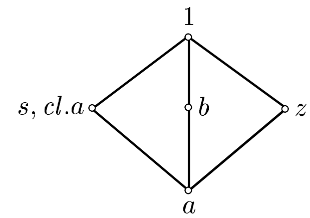

Figure 2: Why we need distributivity in Theorem 7. The above Hasse diagram depicts a lattice. Circles denote elements and $x \leq y$ if there is an upward path from x to y. The lattice depicted is modular, but not distributive: $s \wedge (b \vee z)$ is s whereas $(s \wedge b) \vee (s \wedge z)$ is a. Consider any lattice closure cl that maps a to s. We have that s is a safety element, $a = s \wedge z$, $b \in cmp(cl.a)$, but $z \leq a \vee b$ does not hold.

Theorem 7 does not hold for modular lattices. The lattice depicted in Figure 2 is modular, but not distributive. Consider any lattice closure cl that maps a to s. We have that s is a safety element, $a = s \wedge z$, $b \in cmp(cl.a)$, but $z \leq a \vee b$ does not hold.

In a distributive lattice, complements are unique, thus one can replace b in Theorem 7 with $\neg(cl1.a)$.

An important consequence of the above theorems for the branching time framework is the following theorem.

Theorem 8 *If $(q \in ES \quad \vee \quad q \in US)$ and $p = (q \cap r)$, then $ncl.p \subseteq q$ and $r \subseteq (p \cup \neg ncl.p)$.*

Proof By theorems 6 and 7. $\square$

4.3 Examples

We now return to Martin Rem's example properties given in Section 2.3. Since we are now examining the branching time framework, we will express them using the branching time logic CTL*. When translating the examples to properties over trees, it is sometimes the case that there is more than one reasonable translation. For example, we can translate $p4$ into both $q4a$ and $q4b$; neither of these translations captures the notion that there are only a finite number of a's in a tree but rather that there are a finite number of a's on some (all) paths in the tree.

$q0$:	**false**	**false** (corresponds to $\emptyset$)
$q1$:	a	a
$q2$:	$\neg a$	$\neg a$
$q3a$:	$a \;\wedge\; \mathsf{F}\, \neg a$	$\mathsf{A}(a \;\wedge\; \mathsf{F}\, \neg a) \;\equiv\; a \;\wedge\; \mathsf{AF}\neg a$
$q3b$:		$\mathsf{E}(a \;\wedge\; \mathsf{F}\, \neg a) \;\equiv\; a \;\wedge\; \mathsf{EF}\neg a$
$q4a$:	$\mathsf{FG}\, \neg a$	$\mathsf{A}(\mathsf{FG}\, \neg a)$
$q4b$:		$\mathsf{E}(\mathsf{FG}\, \neg a)$
$q5a$:	$\mathsf{GF}\, a$	$\mathsf{A}(\mathsf{GF}\, a)$
$q5b$:		$\mathsf{E}(\mathsf{GF}\, a)$
$q6$:	**true**	**true** (corresponds to A^{tot}).

For those not familiar with CTL*, here is an informal translation of the above CTL* sentences into English. $q1$ is true of any tree whose root is labeled with a; similarly $q2$ is true of any tree whose root is not labeled with a. $q3a$ is true of the trees whose root is labeled with a and along each path there is a node labeled with $\neg a$. $q3b$ is true of the trees whose root is labeled with a and along some path there is a node labeled with $\neg a$. $q4a$ is true of the trees where along each path, eventually all nodes are labeled with $\neg a$. $q4b$ is true of the trees where along some path, eventually all nodes are labeled with $\neg a$. $q5a$ is true of the trees where along each path, infinitely many nodes are labeled with a. $q5b$ is true of the trees where along some path, infinitely many nodes are labeled with a.

It is not difficult to show that $q0$, $q1$, $q2$, and $q6$ are universally safe (and hence existentially safe).

$fcl.q3a = q1$, as before, but $ncl.q3a \neq q1$ (consider a tree that has at least two paths such that along one of the paths a always holds; this tree is not in $ncl.q3a$). $ncl.q3a \neq q3a$ (trees can be sequences, so $\{ay \mid y \in \Sigma^\omega\} \subseteq ncl.q3a$). $ncl.q3b = q1$ and $fcl.q3b = q1$.

$fcl.q4a = A^{tot}$, as before, but $ncl.q4a \neq A^{tot}$ (consider a tree that has at least two paths such that along one of the paths a always holds; this tree is not in $ncl.q4a$). $ncl.q4a \neq q4a$ (trees can be sequences, so $\{y \mid y \in \Sigma^\omega\} \subseteq ncl.q4a$). $ncl.q4b = A^{tot}$, so $fcl.q4b = A^{tot}$.

$fcl.q5a = A^{tot}$, as before, but $ncl.q5a \neq A^{tot}$ (consider a tree that has at least two paths such that along one of the paths $\neg a$ always holds; this tree is not in $ncl.q5a$). $ncl.q5a \neq q5a$ (trees can be sequences, so $\{y \mid y \in \Sigma^\omega\} \subseteq ncl.q5a$). $ncl.q5b = A^{tot}$, so $fcl.q5b = A^{tot}$.

4.4 Rabin Tree Automata

Let $k \in \mathbb{N}$. A tree $\langle \mathsf{W}, \mathrm{w} \rangle$ is a *k-branching tree* iff for all $\sigma \in \mathsf{W}$ there exists exactly k unique elements of $\mathbb{N}$, $a_0, \ldots, a_{k-1}$, such that $\sigma a_0, \ldots, \sigma a_{k-1} \in \mathsf{W}$. In what follows we consider sets of trees which are k-branching. By $A^{k,tot}$ and $A^{k,f}$ we denote, respectively, the set of k-branching trees and the set of finite trees whose non-leaf nodes have exactly k successors. We carry over the definitions of *ncl* and *fcl* from the previous sections, restricted now to k-branching trees over finite alphabets.

A Rabin tree automaton $\mathcal{B} = (\Sigma, Q, q_0, \delta, \Phi)$ on k-ary infinite trees is defined as follows: Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is the start state, $\delta : Q \times \Sigma \to \mathcal{P}(Q^k)$ is the transition relation, and Φ is the accepting condition.

Let $t = \langle \mathsf{W}, \mathrm{w} \rangle \in A^{k,tot}$. A *run* of $\mathcal{B}$ on t is a Q labeled tree $r = \langle \mathsf{W}, \rho \rangle \in A_Q^{k,tot}$ such that $\rho.\lambda = q_0$ and for all $\sigma \in \mathsf{W}$ and successors $\sigma a_0, \ldots, \sigma a_{k-1} \in \mathsf{W}$, $(\rho.\sigma a_0, \ldots, \rho.\sigma a_{k-1}) \in \delta(\rho.\sigma, \mathrm{w}.\sigma)$. Run r is *accepting* iff for all infinite paths p in W, $\rho(p) \models \Phi$. $\mathcal{L}(\mathcal{B}) = \{t \in A^{k,tot} \mid$ there is an accepting run of $\mathcal{B}$ on $t\}$ is the language of $\mathcal{B}$.

The accepting condition, Φ, is given by specifying pairs of sets $(green_i, red_i) \in (\mathcal{P}(Q))^2$ for $i \in [0..m]$, for some m. Φ holds on a path if for some i, some green state is visited infinitely often and all red states are visited finitely often, *i.e.*, $\Phi = \vee_{i \in [1..m]}[(\vee_{g \in green_i} \mathsf{GF} g) \wedge (\wedge_{r \in red_i} \mathsf{FG} \neg r)]$.

For notational convenience, given a Rabin automaton $\mathcal{B} = (\Sigma, Q, q_0, \delta, \Phi)$ we will refer to $\mathcal{B}(q)$, $q \in Q$, as the automaton given by $(\Sigma, Q, q, \delta, \Phi)$.

Given automaton $\mathcal{B} = (\Sigma, Q, q_0, \delta, \Phi)$ such that $\mathcal{L}.\mathcal{B} \neq \emptyset$, note that $\mathcal{L}.\mathcal{B} = \mathcal{L}.(\Sigma, Q', q_0, \delta', \Phi)$ where δ' is δ restricted to Q' and $Q' = \{q \in Q \mid \mathcal{L}(\mathcal{B}(q)) \neq \emptyset\}$.

We define the finite depth closure, *rfcl*, of an automaton as follows: if $\mathcal{L}.\mathcal{B} = \emptyset$, $rfcl.\mathcal{B} = \mathcal{B}$; otherwise, $rfcl.\mathcal{B} = (\Sigma, Q', q_0, \delta', \Phi')$ where $\Phi' = \vee_{q \in Q'} \mathsf{GF} q$ is a condition that holds along all paths and is generated from $\{(Q', \emptyset)\}$.

In [14] we show that $\mathcal{L}(rfcl.\mathcal{B}) = fcl.\mathcal{L}(\mathcal{B})$. Similarly, one can define the non-total closure of a Rabin automaton. Since languages definable by Rabin automata are (effectively) closed under complementation, intersection, and union [22], they form a Boolean algebra and we can use Theorem 3 to obtain the following result.

Theorem 9 *For any Rabin tree automaton, $\mathcal{B}$, there exist effectively derivable Rabin automata $\mathcal{B}_{safe}$ and $\mathcal{B}_{live}$ such that the language of $\mathcal{B}$ is the intersection of the languages of $\mathcal{B}_{safe}$ and $\mathcal{B}_{live}$. As above, $\mathcal{B}_{safe}$ and $\mathcal{B}_{live}$ can be chosen so that their languages are existentially safe and existentially live, universally safe and universally live, or existentially safe and universally live.*

5. CONCLUSIONS

We gave a lattice-theoretic characterization of safety and liveness that is more general and simpler than previous characterizations and we carefully analyzed the conditions required to prove the main results in this paper. This allowed us to give a uniform treatment of previous results on safety and liveness, including the results for the linear time and branching time frameworks and for ω-regular string and tree languages. Our results allowed us to drastically simplify previous proofs, *e.g.*, we showed that the characterization of safety and liveness for Büchi automata and the decomposition theorem, the main results in [4], follow directly from our results.

6. REFERENCES

[1] M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991.

[2] P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160(1–2):109–127, 2000.

[3] B. Alpern and F. B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, Oct. 1985.

[4] B. Alpern and F. B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.

[5] E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In *Logic of Programs: Workshop*, volume 131 of *LNCS*, pages 52–71. Springer-Verlag, May 1981.

[6] E. A. Emerson. Temporal and modal logic. In van Leeuwen [23], pages 995–1072.

[7] E. A. Emerson and J. Y. Halpern. “Sometimes” and “not never” revisited: on branching versus linear time temporal logic. *JACM*, 33(1):151–178, Jan. 1986.

[8] H. P. Gumm. Another glance at the Alpern-Schneider characterization of safety and liveness in concurrent executions. *Information Processing Letters*, 47(6):291–294, Oct. 1993.

[9] D. Harel and A. Pnueli. On the development of reactive systems. In K. R. Apt, editor, *Logics and Models of Concurrent Systems*, volume F-13 of *NATO ASI Series*, pages 477–498. Springer-Verlag, 1985.

[10] E. Kindler. Safety and liveness properties: A survey. *EATCS-Bulletin*, 53:268–272, June 1994.

[11] D. Kozen. Results on the propositional Mu-Calculus. *Theoretical Computer Science*, pages 334–354, December 1983.

[12] O. Kupferman and M. Y. Vardi. Model checking of safety properties. In N. Halbwachs and D. Peled, editors, *Computer-Aided Verification–CAV '99*, volume 1633 of *LNCS*, pages 172–183. Springer-Verlag, 1999.

[13] L. Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering SE-3*, 2:125–143, Mar. 1977.

[14] P. Manolios and R. Trefler. Safety and liveness in branching time. In *Sixteenth Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 366–374. IEEE Computer Society, 2001.

[15] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1990.

[16] S. S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):455–495, 1982.

[17] A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57, Providence, Rhode Island, 1977. IEEE.

[18] J. P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proc. of the 5th International Symposium on Programming*, volume 137 of *LNCS*, 1982.

[19] M. Rem. A personal perspective of the Alpern-Schneider characterization of safety and liveness. In W. H. J. Feijen, editor, *Beauty is Our Business*, pages 365–372. Springer-Verlag, 1990.

[20] F. B. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(1):30–50, 2000.

[21] A. P. Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6:495–511, 1994.

[22] W. Thomas. Automata on infinite objects. In van Leeuwen [23], pages 135–192.

[23] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science: Volume B: Formal Models and Semantics*. Elsevier, Amsterdam, 1990.