Characterization of Temporal Property Classes*

Edward Chang †& Zohar Manna† & Amir Pnueli ‡

September 8, 1995

Abstract. This paper presents two novel characterizations of the classes of properties of reactive systems in terms of their expression by temporal logic. The first family of characterizations concerns the safety-progress classification, which describes a hierarchy within the set of temporal properties. Previous characterizations of this hierarchy depended critically on the use of past temporal operators. The characterization presented here identifies the future formulas that belong to each class. This characterization is shown to be complete.

The second characterization concerns the safety-liveness classification, which partitions temporal properties into the classes of safety and liveness. While automata-theoretic and temporal logic characterizations of the safety class have been known for some time, a complete characterization of the liveness class by temporal logic remained open. This paper provides such a characterization.

1 Introduction

Reactive systems are systems that are expected to maintain an ongoing interaction with their environment, rather than to produce some final result on termination. A reactive system may be viewed as a generator of *computations*, i.e., infinite sequences of states or events. A (temporal) *property* is a set of such computations. A program P *satisfies* the property Π if every computation of P belongs to Π. In [MP90] we present a hierarchy of temporal properties, the *safety-progress* hierarchy, which corresponds to the first several layers of the Borel topological hierarchy. The main interest in this hierarchy is that properties belonging to the same class have many features in common, and in particular each class can be associated with a characteristic proof rule used to verify that a given program satisfies a property of this class. Since proof rules for the lower classes are simpler than those for the higher classes, it is a worthwhile endeavor to identify, for a given property, the lowest class in the hierarchy to which it belongs.

*This research was supported in part by the National Science Foundation under grants CCR-89-11512 and CCR-89-13641, by the Defense Advanced Research Projects Agency under contract NAG2-703, by the United States Air Force Office of Scientific Research under contract AFOSR-90-0057, and by the European Community ESPRIT Basic Research Action project 3096 (SPEC).

†Department of Computer Science, Stanford University, Stanford, CA 94305

‡Department of Computer Science, Weizmann Institute, Rehovot, Israel

Several characterizations of the hierarchy are presented in [MP90]. The temporal characterization is based on *canonical formulas*, which are boolean combinations of formulas of the form $\Box p$ and $\Box\Diamond p$, where p is an arbitrary *past* formula. This restricted form avoids the use of other future operators such as the *until* operator $\mathcal{U}$. It follows that, in order to determine the minimal class to which a formula belongs, it should be transformed first to canonical form. While this is always possible, the general transformation may lead to an exponential blowup. Furthermore, certain properties are more naturally expressed by future formulas, and the need to translate to past-oriented specifications may be awkward and unnatural.

In this paper, we remedy these drawbacks of the past-based characterization by presenting a considerably wider syntactic characterization of the safety-progress hierarchy using *standard formulas*. This characterization is applicable to arbitrary temporal formulas, including those that use the *until* and *unless* (*waiting-for*) operators. Consequently, without any preliminary transformation, the presented characterization provides an upper bound on where a formula lies in the hierarchy.

The second classification for which we present a temporal logic characterization is the partition of properties into *safety* and *liveness* classes. The fact that temporal properties naturally partition into two disjoint classes was first observed by Lamport in [Lam77]. Semantic characterizations of the safety and liveness classes were given in [Lam85] and [AS85], respectively.

Since the safety class belongs to both the safety-liveness partition and the safety-progress hierarchy, it possesses a (canonical) temporal logic characterization. This characterization was presented in [LPZ85]. An important step in the temporal characterization of the safety-liveness partition was taken by [Sis85], which provided a complete future-based characterization of safety properties. For liveness, only partial characterizations were given.

In this paper we improve on these results by presenting a complete temporal logic characterization of the liveness class, as well as a simpler proof of the completeness of the temporal logic characterization of the safety class.

In Section 2 we sketch the previously known results about the safety-progress classification and its characterization by canonical formulas. In Section 3 we present the new characterization by standard formulas. Section 4 deals with the safety-liveness classification and its temporal characterization.

2 Safety-Progress: Previous Results

In this section we present the previously known facts about the safety-progress hierarchy as given in [MP90].

2.1 The Language-Theoretic View

First, we present the language-theoretic characterization of the safety-progress hierarchy.

We consider a set Σ of *states*, which we also refer to as *letters*. We refer to an infinite sequence of letters as an (infinite) *word*, and to a finite sequence of letters as a *finite word*.

Let Σ^* denote the set of all finite words, Σ^+ denote the set of all nonempty finite words, and Σ^ω denote the set of all (infinite) words. A *language* is a subset of Σ^ω, i.e., a set of words. A *finitary language* is a subset of Σ^+, i.e., a set of finite nonempty words. In the abstract setting considered here, a *property* is a set of infinite words, i.e., a language.

The *length* $|\sigma|$ of a finite word $\sigma : a_1, \ldots, a_j$ is j. The length of an infinite word is defined to be ∞. We write $\widehat{\sigma} \prec \sigma$ (equivalently $\sigma \succ \widehat{\sigma}$) to denote that $\widehat{\sigma}$ is a *prefix* of σ.

We introduce four operators A, E, R, and P that construct languages from finitary languages. Let Φ be a finitary language.

- The language $A(\Phi)$ consists of all words σ such that *every* prefix of σ is in Φ. For example, $A(a^+b^*) = a^\omega + a^+b^\omega$.
- The language $E(\Phi)$ consists of all words σ such that *some* prefix of σ is in Φ. For example, $E(a^+b^*) = a^+b^* \cdot \Sigma^\omega$. In fact, it is true for every finitary language Φ that $E(\Phi) = \Phi \cdot \Sigma^\omega$.
- The language $R(\Phi)$ consists of all words σ such that *infinitely many* prefixes of σ are in Φ. For example, $R(\Sigma^* b) = (\Sigma^* b)^\omega$. The words of this language are precisely those that have infinitely many occurrences of b.
- The language $P(\Phi)$ consists of all words σ such that *all but finitely many* prefixes of σ are in Φ. For example, $P(\Sigma^* b) = \Sigma^* b^\omega$. The words of this language are precisely those that, from a certain point on, have only the letter b.

Based on these four operators we define six classes of languages. A language $\Pi \subseteq \Sigma^\omega$ is defined to be

- A *safety* language if $\Pi = A(\Phi)$ for some finitary Φ.
- A *guarantee* language if $\Pi = E(\Phi)$ for some finitary Φ.
- An *m-obligation* language if $\Pi = \bigcap_{i=1}^{m} \Big(A(\Phi_i) \cup E(\Psi_i) \Big)$ for some finitary languages $\Phi_i, \Psi_i, i = 1, \ldots, m$.
- A *response* language if $\Pi = R(\Phi)$ for some finitary Φ.
- A *persistence* language if $\Pi = P(\Phi)$ for some finitary Φ.
- An *m-reactivity* language if $\Pi = \bigcap_{i=1}^{m} \Big(R(\Phi_i) \cup P(\Psi_i) \Big)$ for some finitary languages $\Phi_i, \Psi_i, i = 1, \ldots, m$.

It follows, for example, that the languages $a^\omega + a^+b^\omega$, $a^+b^* \cdot \Sigma^\omega$, $(\Sigma^* b)^\omega$, and $\Sigma^* b^\omega$ are safety, guarantee, response, and persistence languages, respectively.

In the following, let κ stand for one of the classes: *safety, guarantee, obligation, response, persistence*, or *reactivity*.

We also refer to a κ-language as a *κ-property*. We refer to the classification induced by the language-theoretic characterization as the *semantic* characterization, in contrast to the syntactic characterizations of the temporal formulas or automata that specify properties of a given class.

2.2 Temporal Logic

In this section we introduce the language of temporal logic. The language of temporal logic is constructed inductively from a set of propositions $\mathcal{P}$. We assume an interpretation I such that, for each state $s \in \Sigma$, $I(s) \subseteq \mathcal{P}$ specifies the propositions that are true in s.

Every proposition is a formula; if p and q are formulas, then the following are formulas:

$$\neg p \qquad p \vee q \qquad \bigcirc p \qquad p \,\mathcal{U}\, q \qquad \ominus p \qquad p \,\mathcal{S}\, q$$

A formula whose only logical symbols are $\neg$ or $\vee$ is called a *state formula*. A formula which does not use $\bigcirc$ or $\mathcal{U}$ is called a *past formula*. A formula which does not use $\ominus$ or $\mathcal{S}$ is called a *future* formula.

For a state $s \in \Sigma$ and a proposition $p \in \mathcal{P}$, we define $s \models p$ *iff* $p \in I(s)$. A computation $\sigma = s_1, s_2, \ldots$ *satisfies formula* p *at* $j > 0$, denoted $(\sigma, j) \models p$, as follows:

$(\sigma, j) \models p$ *iff* $s_j \models p$, where p is a proposition

$(\sigma, j) \models \neg p$ *iff* $(\sigma, j) \not\models p$

$(\sigma, j) \models p \vee q$ *iff* $(\sigma, j) \models p$ or $(\sigma, j) \models q$

$(\sigma, j) \models \bigcirc p$ *iff* $(\sigma, j+1) \models p$

$(\sigma, j) \models p \,\mathcal{U}\, q$ *iff* $(\sigma, k) \models q$ for some $k \geq j$, and $(\sigma, i) \models p$ for all i, $j \leq i < k$

$(\sigma, j) \models \ominus p$ *iff* $j > 1$ and $(\sigma, j-1) \models p$

$(\sigma, j) \models p \,\mathcal{S}\, q$ *iff* $(\sigma, k) \models q$ for some $k \leq j$, and $(\sigma, i) \models p$ for all i, $k < i \leq j$

A computation σ is a *model* for p, denoted $\sigma \models p$, exactly when $(\sigma, 1) \models p$. Formulas p and q are said to be *equivalent* if they have the same models. Formulas p and q are *congruent* if, for every computation σ and every position $j > 0$, $(\sigma, j) \models p$ *iff* $(\sigma, j) \models q$. A formula p is *satisfiable* if p has at least one model. A formula p is *valid* if every computation is a model for p.

For a formula p, we define the *property specified by* p, denoted $Spec(p)$, to be the set of all models of p. A property Π is said to be *specifiable by temporal logic* if there exists a formula p such that $\Pi = Spec(p)$.

The following derived temporal operators are also used below:

$p \Rightarrow q = \Box(p \rightarrow q)$	– "entails"		
$\Diamond p = true \,\mathcal{U}\, p$	– "eventually"	$\diamondminus p = true \,\mathcal{S}\, p$	– "earlier"
$\Box p = \neg\Diamond\neg p$	– "always"	$\boxminus p = \neg \diamondminus \neg p$	– "always in the past"
$p \,\mathcal{W}\, q = p \,\mathcal{U}\, q \vee \Box p$	– "waiting-for"	$p \,\mathcal{B}\, q = p \,\mathcal{S}\, q \vee \boxminus p$	– "back-to"

The operators $\mathcal{W}$ and $\mathcal{B}$ are also called *unless* and *weak since*, respectively.

2.3 Characterization by Canonical Formulae

We define a formula to be

- *A canonical safety formula* if it has the form $\Box p$,

- *A canonical guarantee formula* if it has the form $\Diamond p$,
- *A canonical m-obligation formula* if it has the form $\bigwedge_{i=1}^{m}(\Box p_i \ \vee \ \Diamond q_i)$,
- *A canonical response formula* if it has the form $\Box\Diamond p$,
- *A canonical persistence formula* if it has the form $\Diamond\Box p$,
- *A canonical m-reactivity formula* if it has the form $\bigwedge_{i=1}^{m}(\Box\Diamond p_i \ \vee \ \Diamond\Box q_i)$,

where $p, p_1, \ldots, p_m, q_1, \ldots, q_m$, are arbitrary past formulas.

The following theorem establishes the connection between the semantic characterization and the characterization by canonical formulas.

Theorem 1 *A property specifiable by temporal logic is a κ-property iff it is specifiable by a canonical κ-formula.*

It is not too difficult to show that a property Π specifiable by a canonical κ-formula is a κ-property. For example, if Π is specifiable by the canonical safety formula $\Box p$, then it is equal to $A(\Phi)$, where Φ is the set of all finite words $\sigma : s_1, \ldots, s_k$ such that p holds at position k of σ.

The other direction is more involved and requires the construction of an ω-automaton that accepts exactly the models of the formula specifying the property.

In Figure 1, we present a diagram that displays the six classes of the hierarchy with their respective characterizations in the language-theoretic and temporal logic frameworks. A line connecting two classes in the diagram represents strict containment of the lower class in the higher one.

3 Safety-Progress: Characterization by Standard Formulas

In this section we introduce a new characterization of properties based on the definition of *standard κ-formulas.*

Definition 2 *Standard κ-formulas* are defined as follows.

a. *standard safety formulas*

- Every past formula is a standard safety formula.
- The negation of a standard guarantee formula is a standard safety formula.
- If p and q are standard safety formulas, then so are $p \vee q$, $p \wedge q$, $\bigcirc p$, $\Box p$, and $p \,\mathcal{W}\, q$.

b. *standard guarantee formulas*

- Every past formula is a standard guarantee formula.
- The negation of a standard safety formula is a standard guarantee formula.
- If p and q are standard guarantee formulas, then so are $p \vee q$, $p \wedge q$, $\bigcirc p$, $\Diamond p$, and $p \,\mathcal{U}\, q$.

c. *standard obligation formulas*

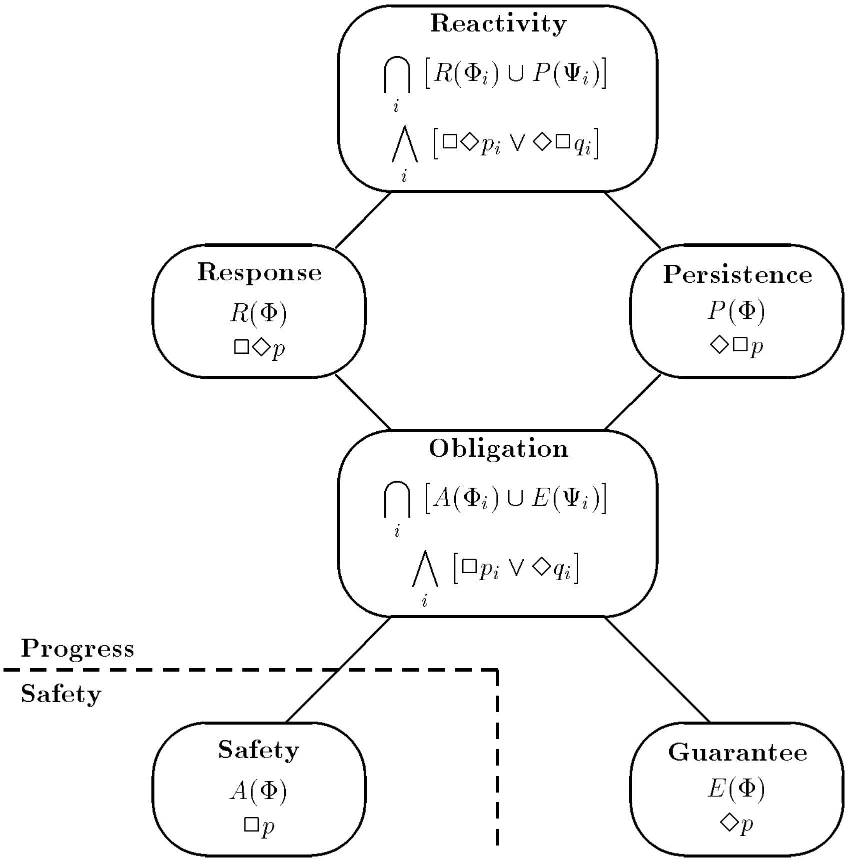

Figure 1: Inclusion Relations between the Classes

- Every standard safety and standard guarantee formula is a standard obligation formula.
- If p and q are standard obligation formulas, then so are $\neg p$, $p \vee q$, $p \wedge q$, and $\bigcirc p$.
- If p is a standard obligation formula and q is a standard guarantee formula, then $p\,\mathcal{U}\,q$ is a standard obligation formula.
- If p is a standard safety formula and q is a standard obligation formula, then $p\,\mathcal{W}\,q$ is a standard obligation formula.

d. *standard response formulas*

- Every standard safety and standard guarantee formula is a standard response formula.
- The negation of a standard persistence formula is a standard response formula.

- If p and q are standard response formulas, then so are $p \vee q$, $p \wedge q$, $\bigcirc p$, $\Box p$, and $p \mathcal{W} q$.
- If p is a standard response formula and q is a standard guarantee formula, then $p \mathcal{U} q$ is a standard response formula.

e. *standard persistence formulas*

- Every standard safety and standard guarantee formula is a standard persistence formula.
- The negation of a standard response formula is a standard persistence formula.
- If p and q are standard persistence formulas, then so are $p \vee q$, $p \wedge q$, $\bigcirc p$, $\Diamond p$, and $p \mathcal{U} q$.
- If p is a standard safety formula and q is a standard persistence formula, then $p \mathcal{W} q$ is a standard persistence formula.

f. If p and q are standard κ-formulas, then so are $\ominus p$, $\diamondminus p$, $\boxminus p$, $p \mathcal{S} q$, and $p \mathcal{B} q$. ▟

The definition does not mention the *reactivity* class since every temporal formula is a standard reactivity formula.

3.1 Separated properties

To establish the connection between the characterization by standard formulas and the other characterizations, we introduce the notion of *separated properties.*

Definition 3 A *separated property* Ω is a subset of $\Sigma^+ \times \mathcal{N}$ such that

$$\forall \langle \sigma, j \rangle \in \Omega.\ 0 < j \leq |\sigma|$$

A separated property Ω is *upwards-closed* if

$$\forall \langle \sigma, j \rangle \in \Omega.\ \forall \sigma' \succ \sigma.\ \langle \sigma', j \rangle \in \Omega$$

Similarly, a separated property Ω is *downwards-closed* if

$$\forall \langle \sigma, j \rangle \in \Omega.\ \forall \sigma' \prec \sigma.\ j \leq |\sigma'| \rightarrow \langle \sigma', j \rangle \in \Omega \ ▟$$

Theorem 4 We can characterize standard *safety, guarantee, response*, and *persistence* formulas as follows.

a. If p is a standard safety formula, there exists a downwards-closed separated property Ω such that

$$(\sigma, j) \vDash p \quad \textit{iff} \quad \text{for every } \sigma' \prec \sigma.\ j \leq |\sigma'| \rightarrow \langle \sigma', j \rangle \in \Omega$$

b. If p is a standard guarantee formula, there exists an upwards-closed separated property Ω such that

$$(\sigma, j) \vDash p \quad \textit{iff} \quad \text{for some } \sigma' \prec \sigma.\ \langle \sigma', j \rangle \in \Omega$$

d. If p is a standard response formula, there exists a separated property Ω such that

$$(\sigma, j) \vDash p \quad \textit{iff} \quad \text{for infinitely many } \sigma' \prec \sigma.\ \langle \sigma', j \rangle \in \Omega$$

e. If p is a standard persistence formula, there exists a separated property Ω such that

$$(\sigma, j) \vDash p \quad \textit{iff} \quad \text{for all but finitely many } \sigma' \prec \sigma.\ \langle \sigma', j \rangle \in \Omega$$

Definition 5 *The finitary property Φ induced by a separated property Ω is*

$$\{\sigma | \langle \sigma, 1 \rangle \in \Omega\}$$

Corollary 6 *Every standard κ-formula specifies a κ-property.*

Proof: The finitary properties Φ induced by the separated properties Ω described in Theorem 4 clearly establish this corollary. ▟

Corollary 7 *Every standard obligation formula specifies an obligation property.*

Note that Corollaries 6 and 7 establish one direction of the connection between the standard and the semantic characterizations. The other direction follows trivially by observing that every canonical κ-formula is a standard κ-formula.

3.2 Completeness of the Future Fragment

The standard formulas classification restricted to future formulas is complete. This is stated in the following theorem.

Theorem 8 *A property Π that is specifiable by temporal logic is a κ-property if and only if it is specifiable by a future standard κ-formula.*

Proof: We will outline the proof which appears in the full version of this paper. By Theorem 1 it suffices to show that every canonical κ-formula is equivalent to some standard κ-formula.

Recall that canonical formulas are boolean combinations of formulas of the form $\Box p$ and $\Box\Diamond p$, where p is an arbitrary past formula. We say that a finite sequence of states $\sigma : s_1, \ldots, s_j$ *end-satisfies* p, denoted $\sigma \dashv p$, if $(\sigma \cdot \sigma', j) \vDash p$ for any $\sigma' \in \Sigma^\omega$; since p is a past formula, the choice of σ' is irrelevant.

We use a translation derived from [SPH84] to obtain, for every past formula p, a star-free expression α such that $\sigma \dashv p$ exactly when σ is in the language of α. Then [Zuc86] provides a translation from star-free expressions to finitary future temporal logic, in which formulas are interpreted over finite state sequences and the predicate *last*, which holds only at the last state of a sequence, is introduced. Thus we have a translation from every past formula p to a finitary future formula φ such that $\sigma \dashv p$ if and only if $\sigma \vDash \varphi$.

It is a property of the translation that an infinite sequence σ is a model of $\varphi\langle true/last \rangle$, the future formula obtained by replacing *last* by *true* in φ, exactly when there is some prefix of σ that satisfies φ. Therefore $\varphi\langle true/last\rangle$ is equivalent to $\diamond p$, and since $\varphi\langle true/last\rangle$ is a standard guarantee formula, this shows the case for $\kappa = guarantee$. The case for $\kappa = safety$ follows by duality.

For the case of $\kappa = response$ (and $\kappa = persistence$, by duality), we construct the *limit of* φ, a future formula which accepts the set of infinite states sequences with infinitely many prefixes satisfying φ; these are also exactly the models of $\Box\diamond p$. This construction is given in [Zuc86] and, after some manipulation, yields a standard response formula. ∎

3.3 A Verification Strategy Using Standard Formulas

In this section we will briefly describe how the standard formulas characterization may be used in verification. We represent programs as *fair transition systems.* In this model, a program P consists of the following components: V (a finite set of *state variables*), Σ (a set of states, each of which is an interpretation over V), $\mathcal{T}$ (a finite set of *transitions*), and Θ (an assertion characterizing all initial states).

Every transition $\tau \in \mathcal{T}$ is associated with a *transition relation* $\rho_\tau(V, V')$, which may refer to unprimed and primed versions of the state variables. The purpose of ρ_τ is to express a relation between a state s and its successor s'; we define s' to be a *τ-successor* of s if $\langle s, s'\rangle \vDash \rho_\tau(V, V')$, where $\langle s, s'\rangle$ is the joint interpretation which interprets $x \in V$ as $s(x)$ and x' as $s'(x)$. A transition τ is *enabled on a state* s if $En(\tau)$, given by $(\exists V')\rho_\tau(V, V')$, holds. Consider an infinite sequence of states $\sigma: \; s_1, s_2, \ldots$. A transition τ is *enabled at position* k if τ is enabled on s_k. A transition τ is *taken at position* k if s_{k+1} is a τ-successor of s_k. The sequence σ is a *computation of* P if it satisfies the following requirements:

- *Initiality* — State s_1 is *initial*, i.e., $s_1 \vDash \Theta$
- *Consecution* — For every $j > 0$, there is some $\tau \in \mathcal{T}$ such that s_{j+1} is a τ-successor of s_j
- *Justice* — For every $\tau \in \mathcal{T}$, if τ is continually enabled beyond some position j, then τ is taken at some position $k > j$.

Given a temporal logic specification φ, P *satisfies* φ if every computation of P is a model of φ. One approach to verification, i.e., proving that P satisfies φ, is to determine where φ belongs in the safety-progress hierarchy, choose a suitable proof rule for properties of that class, translate φ to an equivalent specification φ' for which the proof rule is applicable, and then perform the verification.

Consider, for example, a system in which a server process must respond to requests by a client process, unless the server is able to observe that the client must eventually achieve some goal. This can be specified by:

$$\varphi: \quad (request \;\rightarrow\; \diamond response)\, \mathcal{W}\, (\diamond goal)$$

According to the standard formulas characterization, φ is a response property, and we might hope to apply rule `C-RESPONSE`. This rule states that, in order to show $\Box\diamond p$, it suffices to find intermediate assertions φ_i and helpful transitions τ_i such that the following conditions hold. Every $(\neg p)$-position entails that one of the φ_i

C-RESPONSE

Find intermediate assertions $\varphi_0, \ldots, \varphi_k$
and helpful transitions $\tau_1, \ldots, \tau_k$:

R1. $\neg p \;\Rightarrow\; \bigvee_{i=0}^{k} \varphi_i$

R2. $\varphi_0 \;\Rightarrow\; p$

For every $i > 0$:

R3. $(\varphi_i \;\wedge\; \rho_\tau) \;\Rightarrow\; \bigvee_{j \leq i} \varphi'_j$ for every $\tau \in \mathcal{T}$

R4. $(\varphi_i \;\wedge\; \rho_{\tau_i}) \;\Rightarrow\; \bigvee_{j < i} \varphi'_j$

R5. $\varphi_i \;\Rightarrow\; En(\tau_i)$

$$\overline{\qquad \Box\Diamond p \qquad}$$

holds. The final φ_0 entails p. Every transition from a φ_i-position results in a φ_j-position with $j \leq i$, and the helpful transition τ_i must result in a φ_j-position with $j < i$. Finally, φ_i entails that the helpful transition is enabled, so by the justice requirement, τ_i is eventually taken.

Before applying rule C-RESPONSE, we must first translate φ to its equivalent canonical form $\Box\Diamond p$. This can be achieved by

$$p: \quad \diamondminus\, goal \;\vee\; (\neg request)\, \mathcal{B}\, response$$

and we may the proceed with the verification.

There are two principal drawbacks to this approach. First, in translating to canonical form, we obtain a formula that has less intuitive meaning than the original φ. Second, we are obligated to work with past formulas during the verification. In particular, the premises of rule C-RESPONSE are entailments.

Consider instead the following strategy. Starting with the original specification φ, we attempt to replace subformulas of φ by state formulas. In the example given above, for instance, it would suffice to identify state formulas p and q such that the program P satisfies the following:

(1) $p \;\Rightarrow\; request \rightarrow \Diamond response$

(2) $q \;\Rightarrow\; \Diamond goal$

(3) $p\, \mathcal{W}\, q$

We can further simplify (1) if we identify a state formula r such that P satisfies:

(1a) $r \;\Rightarrow\; \Diamond response$

(1b) $p \;\Rightarrow\; request \rightarrow r$

The formulas that we obtain through this procedure, such as (1a), (1b), (2), and (3), have a simple form, and we can provide straightforward proof rules for verifying them. Rule WAITING, for example, can be used for formulas (1b) and (3), and a simpler version of rule C-RESPONSE can be used for (1a) and (2). In this manner we can verify complex specifications piecemeal without having to translate the original specification into any particular form, and our proof rules yield temporal conclusions from first-order premises.

As an example, consider the following program:

WAITING	
Find strengthening assertion φ:	
W1. $p \rightarrow (r \vee \varphi)$	
W2. $\varphi \rightarrow q$	
W3. $(\varphi \wedge \rho_\tau) \rightarrow (r \vee \varphi)$	for every $\tau \in \mathcal{T}$
$p \Rightarrow q \, \mathcal{W} \, r$	

local x, y, z : **integer where** $x = y = 0, z > 0$

$$P1 :: \begin{bmatrix} \ell_0\text{: } \mathbf{while}\ y = 0\ \mathbf{do} \\ \quad \begin{bmatrix} \ell_1\text{: } \mathbf{if}\ x = 2\ \mathbf{then} \\ \quad \begin{bmatrix} \ell_2\text{: } x := 1 \\ \ell_3\text{: } x := 0 \end{bmatrix} \end{bmatrix} \\ \ell_4\text{: } x := 1 \\ \ell_5\text{: } x := 0 \end{bmatrix} \quad || \quad P2 :: \begin{bmatrix} m_0\text{: } \mathbf{loop\ forever\ do} \\ \quad \begin{bmatrix} m_1\text{: } x := 2 \\ m_2\text{: } \mathbf{if}\ x = 1\ \mathbf{then} \\ \quad m_3\text{: } y := 1 \\ m_4\text{: } \mathbf{if}\ y = 1\ \mathbf{then} \\ \quad m_5\text{: } z := 0 \end{bmatrix} \end{bmatrix}$$

The "goal" of this program is to set $z = 0$. It is clear that this happens exactly if $y = 1$ eventually holds. Process $P2$ signals "request" by setting $x = 2$, and process $P1$ will "respond" by setting $x = 1$. If $P2$ observes a response before $P1$ changes to $x = 0$, then $P2$ will set $y = 1$. We can prove

$$\big(x = 2 \rightarrow \Diamond(x = 1)\big) \ \mathcal{W} \, \Diamond(z = 0)$$

by taking p, q, and r to be:

$$\begin{array}{ll} p: & at\_\ell_{0\ldots3} \\ q: & y = 1 \\ r: & at\_\ell_{0\ldots3} \ \wedge \ x = 2 \end{array}$$

4 Safety-Liveness: A New Characterization

First, consider the language-theoretic characterization of the safety and liveness properties, as formulated in [AS85].

- A property Π is a *safety* property if it equals $A(\Phi)$ for some finitary language Φ.
- A property Π is a *liveness property* if every finite word $\sigma \in \Sigma^+$ is a prefix of some infinite word σ' belonging to Π. That is, every finite word can be extended to an infinite word in Π.

The temporal characterization of these two classes is based on the following definition.

- A *canonical safety formula* is any formula of the form $\Box p$, where p is some past formula.

- A *canonical liveness formula* is any formula of the form $\diamond \bigvee_{i=1}^{k}(p_i \wedge \bigcirc f_i)$, where $p_1 \ldots, p_k$ are past formulas such that $\square(\bigvee_{i=1}^{k} p_i)$ is valid, and $f_1, \ldots, f_k$ are satisfiable future formulas.

The definition of canonical safety formula is identical to the one used for the safety-progress hierarchy, and is repeated here for completeness.

To establish the connection between the characterization by canonical temporal formulas and the semantic characterization, we use the following separation theorem due to Gabbay [Gab87]. The following version of this theorem has been adapted to fit our notation.

Theorem 9 (Separation) *Every temporal formula is congruent to a formula of the form $\bigvee_{i=1}^{k}(p_i \wedge \bigcirc f_i)$, where $p_1 \ldots, p_k$ are past formulas and $f_1, \ldots, f_k$ are future formulas.*

Theorem 10 *A property specifiable by temporal logic is a λ-property iff it is specifiable by a canonical λ-formula, where λ is either* safety *or* liveness.

Proof: We consider separately the cases of safety and liveness. Let Π be a property specified by the canonical safety formula $\square p$. Taking the finitary language Φ to be the set of all finite words $s_1, \ldots, s_k$ satisfying the past formula p at position k, it is not difficult to see that $\Pi = A(\Phi)$.

In the other direction, assume that the property Π is specified by the temporal formula φ and is known to be a safety property. In this case there exists a finitary language Φ such that $\Pi = A(\Phi)$. Consider the formula $\psi : \diamondminus(\mathit{first} \wedge \varphi)$, where *first* stands for the formula $\neg \circleddash \mathit{true}$ and is true precisely at position 1 of every word.

The formula ψ has the property that, for every word σ and position $j > 0$, $(\sigma, j) \models \psi$ iff $(\sigma, 1) \models \varphi$ iff $\sigma \in \Pi$. Applying the separation theorem to ψ, we obtain a congruent formula $\bigvee_{i=1}^{k}(p_i \wedge \bigcirc f_i)$. Without loss of generality, we may assume that every f_i is satisfiable; otherwise we can drop the conjunction $p_i \wedge \bigcirc f_i$ from the formula. Define the formula $\chi : \square(\bigvee_{i=1}^{k} p_i)$, which is obviously a canonical safety formula. We will show that a word σ satisfies χ iff it belongs to Π. This will show that Π is specifiable by a canonical safety formula.

Assume first that $\sigma \in \Pi$. In that case it satisfies φ which implies that ψ holds at all positions of σ. This means that $\bigvee_{i=1}^{k}(p_i \wedge \bigcirc f_i)$, and therefore $\bigvee_{i=1}^{k} p_i$, holds at all positions of σ. It follows that $\chi : \square(\bigvee_{i=1}^{k} p_i)$ holds at position 1 of σ.

Next, consider a word σ that satisfies $\chi : \square(\bigvee_{i=1}^{k} p_i)$. Consider an arbitrary position $j > 0$. Obviously, for some $i = 1, \ldots, k$, p_i holds at position j. Since f_i is satisfiable, there exists a word $\widehat{\sigma} : \widehat{s}_1, \widehat{s}_2, \ldots$, satisfying f_i. Consider the word $\sigma' : s_1, \ldots, s_j, \widehat{s}_1, \widehat{s}_2, \ldots$, obtained by concatenating $\widehat{\sigma}$ to the end of the j-prefix of σ. Clearly, $p_i \wedge \bigcirc f_i$ holds at position j of σ'. It follows that $\bigvee_{i=1}^{k}(p_i \wedge \bigcirc f_i)$, and therefore ψ, holds at position j. Consequently, σ' belongs to Π and, therefore, all of its prefixes, including $s_1, \ldots, s_j$, belong to Φ. Since j was an arbitrarily chosen position, this shows that all prefixes of σ belong to Φ, and therefore σ belongs to Π.

Next, consider the case of liveness. In one direction, let Π be a property specifiable by the canonical liveness formula $\diamond \bigvee_{i=1}^{k}(p_i \wedge \bigcirc f_i)$. Let $\sigma : s_1, \ldots, s_j$ be an arbitrary finite word. Since $\Box(\bigvee_{i=1}^{k} p_i)$ is valid, $\bigvee_{i=1}^{k} p_i$ must hold at all positions, including position j of σ. This means that $(\sigma, j) \vDash p_i$ for some $i = 1, \ldots, k$. Since f_i is satisfiable, there exists a word $\widehat{\sigma} : \widehat{s}_1, \widehat{s}_2, \ldots$, satisfying f_i. Consider the word $\sigma' : \ s_1, \ldots, s_j, \widehat{s}_1, \widehat{s}_2, \ldots$, obtained by concatenating $\widehat{\sigma}$ to the end of σ. Clearly, $p_i \wedge \bigcirc f_i$ holds at position j of σ'. It follows that $\bigvee_{i=1}^{k}(p_i \wedge \bigcirc f_i)$ holds at position j, implying that σ' satisfies $\Diamond \bigvee_{i=1}^{k}(p_i \wedge \bigcirc f_i)$. Consequently, σ' belongs to Π, and we have shown that any finite word can be extended to a word in Π.

In the other direction, assume that the property Π is specified by the temporal formula φ and is known to be a liveness property. Again we construct the formula ψ : $\diamondminus(\mathit{first} \wedge \varphi)$ and, using the separation theorem, the formula $\chi : \bigvee_{i=1}^{k}(p_i \wedge \bigcirc f_i)$ congruent to ψ. Since φ specifies Π, it follows that a word σ is in Π iff it satisfies ψ iff it satisfies $\Diamond\chi = \Diamond \bigvee_{i=1}^{k}(p_i \wedge \bigcirc f_i)$. We will show that $\Box(\bigvee_{i=1}^{k} p_i)$ is valid, thus showing that $\Diamond\chi$ is a canonical liveness formula specifying Π.

Let $\sigma : \ s_1, \ldots, s_j$ be an arbitrary finite word. Since Π is a liveness property, there exists an infinite word extending σ, $\sigma' : \ s_1, \ldots, s_j, s_{j+1}, \ldots$, which belongs to Π. Since φ specifies Π, $\bigvee_{i=1}^{k}(p_i \wedge \bigcirc f_i)$ and, therefore, $\bigvee_{i=1}^{k} p_i$ must hold at all positions of σ', including position j. As $\bigvee_{i=1}^{k} p_i$ is a past formula, this implies that it holds at position j of σ. This shows that $\bigvee_{i=1}^{k} p_i$ holds at an arbitrary position of an arbitrary finite sequence, so $\Box(\bigvee_{i=1}^{k} p_i)$ is valid. ▟

References

[AS85] B. Alpern and F.B. Schneider. Defining liveness. *Info. Proc. Lett.*, 21:181–185, 1985.

[Gab87] D. Gabbay. The declarative past and imperative future. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logic in Specification*, volume 398 of *Lect. Notes in Comp. Sci.*, pages 407–448. Springer-Verlag, 1987.

[Lam77] L. Lamport. Proving the correctness of multiprocess programs. *IEEE Trans. Software Engin.*, 3:125–143, 1977.

[Lam85] L. Lamport. *Basic Concepts*, volume 190 of *Lect. Notes in Comp. Sci.*, pages 19–30. Springer-Verlag, 1985.

[LPZ85] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Proc. Conf. Logics of Programs*, volume 193 of *Lect. Notes in Comp. Sci.*, pages 196–218. Springer-Verlag, 1985.

[MP90] Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *Proc. 9th ACM Symp. Princ. of Dist. Comp.*, pages 377–408, 1990.

[Sis85] A.P. Sistla. On caracterization of safety and liveness properties in temporal logic. In *Proc. 4th ACM Symp. Princ. of Dist. Comp.*, pages 39–48, 1985.

[SPH84] R. Sherman, A. Pnueli, and D. Harel. Is the interesting part of process logic uninteresting?: A translation from *pl* to *pdl*. *SIAM J. Comp.*, 13:825–839, 1984.

[Zuc86] L. Zuck. *Past Temporal Logic*. PhD thesis, Weizmann Institute, 1986.