The Safety-Progress Classification*

Edward Chang,[1] Zohar Manna,[1,2] and Amir Pnueli[2]

[1] Department of Computer Science, Stanford University, Stanford, CA 94305
[2] Department of Applied Mathematics, Weizmann Institute, Rehovot, Israel

Abstract: We propose a classification of temporal properties into a hierarchy, called the *safety-progress* classification. The classes of the hierarchy are characterized through four views: a language-theoretic view, a topological view, a temporal logic view, and an automata view. In the topological view, the considered hierarchy coincides with the two lower levels of the Borel hierarchy, starting with the closed and open sets. For properties that are expressible by temporal logic and automata, we provide syntactic characterizations of the formulas and automata that correspond to properties in the different classes. We relate this classification to the well-known *safety-liveness* classification, and show that in some sense the two are orthogonal to one another.

Keywords: safety-progress classification, safety-liveness classification, temporal logic, Streett automata, safety, liveness, progress, Borel hierarchy, reactive systems, guarantee, obligation, response, persistence, reactivity

Contents

1 Introduction **2**
- 1.1 Specification . . . 3
- 1.2 The Safety-Liveness Classification . . . 3
- 1.3 The Safety-Progress Classification . . . 5
- 1.4 Terminology . . . 6

2 The Language-theoretic View **6**
- 2.1 The Basic Classes . . . 8
- 2.2 Closure of the Basic Classes . . . 9
- 2.3 Characterization of the Lower Classes . . . 11
- 2.4 Inclusion among the Classes . . . 11
- 2.5 The Compound Classes . . . 12
- 2.6 The Safety-Liveness Classification . . . 15

*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 N00039-84-C-0211, by the United States Air Force Office of Scientific Research under contracts AFOSR-90-0057, and by the European Community ESPRIT Basic Research Action project 3096 (SPEC). The first author was supported by a National Science Foundation Fellowship.

3 The Topological View **18**
3.1 Safety Properties as *Closed* sets . . . 18
3.2 Guarantee Properties as *Open* sets . . . 19
3.3 Response Properties as G_δ sets . . . 19
3.4 Persistence Properties as F_σ sets . . . 20
3.5 The Topological Characterization of Liveness . . . 21

4 The Temporal Logic View **21**
4.1 The Language of Temporal Logic . . . 21
4.2 The Temporal Hierarchy of Properties . . . 23
4.3 Safety Formulas . . . 24
4.4 Guarantee Formulas . . . 26
4.5 Obligation Formulas . . . 28
4.6 Response Formulas . . . 30
4.7 Persistence Formulas . . . 32
4.8 Reactivity Formulas . . . 33
4.9 Relating the Syntactic and Semantic Classifications . . . 36
4.10 Standard Formulas . . . 37
4.11 The Temporal Characterization of Liveness . . . 40

5 The Automata View **42**
5.1 The Hierarchy of Automata . . . 43
5.2 Deciding the Type of a Property . . . 56
5.3 Connections Between Temporal Logic and Automata . . . 57

1 Introduction

This paper deals with some methodological aspects of the development of correct *reactive systems*. Reactive systems are systems (and programs) whose main role is to maintain an ongoing interaction with their environment, rather than to produce some final result on termination. Such systems should be specified and analyzed in terms of their behaviors, i.e., the sequences of states or events they generate during their operation. The class of reactive systems includes operating systems, programs controlling industrial plants, embedded systems, and many others. It is clear that it also includes the classes of concurrent and distributed programs, since, independent of the goal and purpose of the complete system, each component of a concurrent or distributed system must be studied in terms of its ongoing interaction with other system components.

A reactive system may be viewed as a generator of *computations*, which, for now, we may assume to be infinite sequences of states or events. In case the program or system terminates, we may always extend the finite state sequence it has generated by adding an infinite sequence of duplicate states or dummy events to obtain an (infinite) computation.

In general, we define a *property* to be a set of computations. A program P is said to *satisfy* the property Π if all the computations of P belong to Π. Several languages and formalisms have been proposed for expressing properties of programs, including the language of temporal logic [Pnu77], [MP81], [Lam83] and the formalism of automata [AS89], [MP87].

1.1 Specification

An important approach to the specification and verification of reactive systems is based on specifying a program by listing several properties, representing *requirements*, that the program ought to satisfy. This approach enjoys the advantages of abstraction and modularity. By *abstraction* we mean that, since the specifier lists several properties and is not required to show how they can be integrated or worry about how they interact with one another, he is not tempted to overspecify or actually design the system. Consequently, this approach leads to specifications which are considerably free of implementation bias.

By *modularity* we mean that a specification consisting of a list of properties is easy to modify by dropping, adding, or changing a single property. Also, the task of verifying that a proposed implementation satisfies its specification can be done in a modular fashion by verifying each property separately.

One of the major drawbacks of this property-based approach to specification is that, while it discourages overspecification, it may lead to *underspecification*. Thus, a constant concern in working with such specifications is that of *completeness*: Have we specified enough properties to guarantee that any implementation will be "close enough" to our original intent?

A classical example of underspecification is a specification for a mutual exclusion algorithm that includes the obvious requirement that no two processes reside in their critical sections at the same time, but neglects to require that each interested process will eventually obtain access to its critical section. A trivial but obviously unsatisfactory implementation of this flawed specification is one in which no process ever reaches its critical section.

A partial remedy to the completeness problem can be provided by a deeper study and detailed classification of the different types of properties. This can at least provide the specifier with a check list of property classes to consider. For each of the property classes he or she can ask the questions: Is there a property of this type that is relevant to the system I am specifying? Have I already specified it?

1.2 The Safety-Liveness Classification

A useful partition of properties into the classes of *safety* and *liveness* has been suggested by Lamport in [Lam77]. The two classes have been informally characterized as:

- A *safety* property states that some bad thing *never* happens.
- A *liveness* property states that some good thing *eventually* happens.

One important advantage of this classification is that each class encompasses properties of similar character. Safety properties typically represent requirements that should be continuously maintained by the system. They often express *invariance* properties of the system. For example, if the bad thing represents a violation of mutual exclusion, then the statement that it never happens ensures that mutual exclusion is continuously maintained. Liveness properties, on the other hand, typically represent requirements that need not hold continuously, but whose eventual (or repeated) realization must be ensured. For example, if the good thing represents the situation in which a process enters its critical section, then the statement that it eventually happens ensures the absence of livelock (or individual starvation).

Thus, we can immediately identify the flaw in the specification considered above as the absence of an appropriate liveness property. With the addition of this property, the specification is correct and admits only satisfactory solutions.

To draw an analogy with terminating programs, a safety property corresponds to partial correctness, which does not ensure termination but only that all terminating computations produce correct results. Liveness properties correspond to total correctness, which guarantees termination. For reactive systems, which may never terminate, the spectrum of relevant and useful liveness properties is much richer than the single property of termination. For example, it also includes the property requiring that a certain event occurs *infinitely many times*.

While it is generally recognized that a complete specification of a system should include both a safety part and a liveness part, there is cost incurred when using a language that can express both classes of properties. For example, if we are willing to restrict ourselves to expressing only safety properties, then a language of predicates over *finite* behaviors (or finite prefixes of infinite behaviors) suffices. The only justification for using temporal logic or equivalent formalisms, which are considerably more complex since they define predicates over *infinite* behaviors, is for expressing liveness properties. Thus an important reason for studying the classification of properties is to identify the tradeoff between the expressiveness of a specification language and its complexity.

Another reason for wishing to distinguish between safety and liveness properties is that their verification calls for different proof techniques. To prove safety properties, one essentially uses *computational induction* (using the terminology of [Man74]) based on an invariance principle. According to this principle, to establish a safety or an invariance property, we show that it holds initially and that it is preserved by each individual action of the program. Consequently, based on an induction argument on the position in the computation, it follows that the property always holds. Note that the inductive argument appears only in the justification of the proof principle, but not in its application. That is, the user of the principle only establishes the two facts mentioned above, i.e., initial validity and preservation over the program steps. The induction is *implicit*.

Liveness properties, on the other hand, are proven using *structural induction*, i.e., *explicit* induction on some function of the state that measures the distance away from the realization of the "good thing." This induction is often represented as an application of a well-founded argument. We refer the reader to [OL82], [MP84] for a discussion of the proof principles associated with each class.

A more formal semantic definition of safety properties has been given in [Lam83], and a semantic definition of liveness properties has been given in [AS85]. These two definitions lead to the following pleasing consequences:

- The classes of safety and liveness properties are disjoint (except for the trivial properties of the set of no computations and the set of all computations).
- Every property can be represented as the intersection of a safety property and a liveness property.
- The classes contain the obvious properties that are intuitively associated with them, i.e., invariance and partial correctness in the safety class, and termination and absence of individual starvation in the liveness class.

- All the properties classified as safety properties can be proven using an invariance principle, while the properties classified as liveness properties can be proven using a well-founded principle.

In [Sis85] Sistla gave a syntactic characterization of safety properties in terms of the temporal logic formulas expressing them. For liveness properties, however, only partial characterizations were given. In section 4.11 we improve on these results by presenting a complete temporal characterization of the class of liveness properties, including a simpler proof of the completeness of the temporal characterization of the safety class. These results are based on a separation theorem due to Gabbay [Gab87].

Alpern and Schneider (see [AS89], [AS87]) studied the safety-liveness classification in terms of predicate automata, a topic we will cover in section 5. They provided a syntactic characterization of the two classes by imposing structural constraints on the automata describing such properties.

1.3 The Safety-Progress Classification

In this paper we present an alternative classification of properties, which we refer to as the *safety-progress classification*. There is a strong correspondence between this classification and the lower two levels of the classical Borel topological hierarchy.

The safety-progress classification agrees with the safety-liveness classification on the class of safety properties. It differs from the safety-liveness classification on the other classes. Unlike the safety-liveness classification, the safety-progress classification is a hierarchy and not a partition. In particular, some non-safety classes properly contain the class of safety properties. The situation is similar in that respect to the hierarchy of formal languages, where, for example, the class of context-free languages properly contains the class of regular languages. We have to designate a language as *strictly* context-free in order to identify it as context-free but not regular.

It is possible to argue that the significance of the safety-liveness classification, namely, the recognition that the liveness class requires its own proof principle and a more complex specification language, arises from the distinction between the safety and non-safety classes. The safety-progress classification also makes such a distinction, but in addition provides a finer classification of the non-safety classes. This classification enables us to distinguish between properties stating that a certain good thing occurs at least once, and properties stating that a certain good thing occurs infinitely often in a computation.

We examine the safety-progress classification from four points of view. The first view is *language-theoretic*, where we characterize the different classes according to the way they can be constructed from sets of finite sequences. The second view is *topological*, where we characterize the classes as sets with certain topological properties. This is where we establish the correspondence with the Borel topological hierarchy. Next, we consider properties which are expressible in *temporal logic*, and give for each class a syntactic characterization of the formulas that express properties belonging to that class. We also state a theorem allowing us to easily obtain a reasonable upper bound on the class to which a temporal formula belongs. Finally, we consider the expression of properties by *automata*, and give a syntactic characterization of the automata describing properties belonging to each class. We show that these four views of the hierarchy coincide.

A hierarchy very similar to the one considered here has been studied extensively in the context of automata over infinite words, which is the fourth view we consider. The properties of the lower ranks of the hierarchy, which are our main subject of interest, have been established by Landweber in [Lan69]. The complete hierarchy has been analyzed in [Wag79] (and several years later in [Kam85]). They have also established the connection to the topological characterization. Consequently, many of the results described in the section on automata have been established in these two works. The corresponding results about temporal logic can usually be derived from the automata-theoretic results by restriction to *counter-free* automata [Zuc86].

Parts of this paper have appeared in other places [MP89], [MP90b].

1.4 Terminology

Throughout this paper, we take *computations* to be infinite sequences of states. We will use *finite computation prefix*, *proper computation prefix*, and *finite state sequence* interchangeably when referring to finite sequences of states. We assume a (possibly infinite) set of states Σ, and, for the most part, we assume that states have no internal structure.

We denote by Σ^* the set of all finite state sequences over Σ, and by Σ^+ the set of all *non-empty* finite state sequences. We denote the length (number of states) of a finite state sequence σ by $|\sigma|$. Let Σ^ω denote the set of all *infinite* state sequences. For a finite state sequence $\sigma \in \Sigma^+$ and any state sequence σ', we denote by $\sigma \prec \sigma'$ the fact that σ is a proper (finite) prefix of σ'. We denote the more general relation $(\sigma \prec \sigma') \vee (\sigma = \sigma')$ by $\sigma \preceq \sigma'$.

The word $\sigma \cdot \sigma'$ is obtained by concatenating σ' to the end of σ. The concatenation of sets of state sequences $\Phi \cdot \Pi$ contains all infinite state sequences in Φ, as well as $\sigma \cdot \sigma'$ where $\sigma \in \Phi$ is a finite state sequence and $\sigma' \in \Pi$.

2 The Language-theoretic View

As defined above, a property is any set of infinite state sequences. Another name for such a set is a *language* over the alphabet Σ. We will therefore consider the terms property and language to be synonymous. Consistent with this terminology, we will refer to sequences as *words*. In formal language theory, it is customarily required that the alphabet be finite. However, the extension to infinite alphabets is straightforward.

A set $\Phi \subseteq \Sigma^+$ of non-empty finite words is called a *finitary property*. We may refer to a property as an *infinitary property*, when we wish to emphasize the distinction from the finitary properties. Although our ultimate interest is in infinitary properties, since computations are infinite sequences of states, the theory of infinitary properties makes extensive use of finitary properties as the building blocks from which infinitary properties are constructed.

For a finitary or infinitary property Π, we denote by $\mathit{Pref}(\Pi)$ the finitary property containing all finite prefixes of Π.

We define the complements of a finitary property Φ and of an infinitary property Π, denoted respectively by $\overline{\Phi}$ and $\overline{\Pi}$, as

$$\overline{\Phi} = \Sigma^+ - \Phi, \quad \overline{\Pi} = \Sigma^\omega - \Pi.$$

The classification of properties from the language-theoretic view is based on the construction of infinitary properties from finitary ones. We propose four operators for constructing infinitary properties from finitary properties. We present below the definition of the properties that are obtained by applying the operators A, E, R, and P to a given finitary property Φ. We will illustrate these definitions on simple finitary properties, which we will describe using the notation of regular expressions, extended by the notation α^ω denoting the infinite product of the expression α. Thus, the language denoted by α^ω consists of all the infinite words that can be presented as the infinite concatenation

$$\sigma_0 \cdot \sigma_1 \cdot \sigma_2 \cdots$$

where each σ_i is a finite non-empty word belonging to the language denoted by α.

- The property $A(\Phi)$ consists of all infinite words σ such that

  All prefixes of σ belong to Φ.

  For example, if $\Phi = a^+b^*$, then $A(\Phi) = a^\omega + a^+b^\omega$.

- The property $E(\Phi)$ consists of all infinite words σ such that

  Some prefix of σ belongs to Φ.

  For example, $E(a^+b^*) = a^+b^* \cdot \Sigma^\omega$. In fact, it is true for every finitary property Φ that $E(\Phi) = \Phi \cdot \Sigma^\omega$.

- The property $R(\Phi)$ consists of all infinite words σ such that

  Infinitely many prefixes of σ belong to Φ.

  For example, $R(\Sigma^* b) = (\Sigma^* b)^\omega$. This language contains all the words that have infinitely many occurrences of b.

- The property $P(\Phi)$ consists of all infinite words σ such that

  All but finitely many prefixes of σ belong to Φ.

  For example, $P(\Sigma^* b) = \Sigma^* b^\omega$. This language contains all the words that, from a certain point on, contain only occurrences of b.

The motivation for denoting the two last operators by R and P is that prefixes belonging to Φ occur *recurrently* in $R(\Phi)$, and *persistently* (from a certain point on) in $P(\Phi)$.

For some of the developments below, it is useful to define finitary versions of the operators A and E. Let Φ be a finitary property. We define

- The finitary property $A_f(\Phi)$ consists of all *finite* words σ such that

  All prefixes of σ are in Φ.

- The finitary property $E_f(\Phi)$ consists of all *finite* words σ such that

  Some prefix of σ belongs to Φ.

To illustrate the difference between the operators A, E and their finitary versions, consider the following examples:

$$\begin{array}{ll} A_f(a^+b^*) = a^+b^* & A(a^+b^*) = a^\omega + a^+b^\omega \\ E_f(a^+b^*) = a^+b^* \cdot \Sigma^* & E(a^+b^*) = a^+b^* \cdot \Sigma^\omega \end{array}$$

The four operators A, E, R, and P are not completely independent. In fact A and E are dual operators, and so are R and P. By duality we mean that

$$\overline{A(\Phi)} = E(\overline{\Phi}) \quad \text{and} \quad \overline{R(\Phi)} = P(\overline{\Phi})$$

where complementation is taken with respect to Σ^+ for Φ, and with respect to Σ^ω for $A(\Phi)$ and $R(\Phi)$. The finitary operators A_f and E_f are likewise dual:

$$\overline{A_f(\Phi)} = E_f(\overline{\Phi}).$$

Let us show that $\overline{A(\Phi)} = E(\overline{\Phi})$ holds. Clearly $\sigma \in A(\Phi)$ iff all prefixes of σ belong to Φ. Consequently $\sigma \notin A(\Phi)$ iff there exists at least one prefix σ' of σ that does not belong to Φ. This means that σ has a prefix, namely σ', that belongs to $\overline{\Phi}$, which is true iff $\sigma \in E(\overline{\Phi})$.

2.1 The Basic Classes

Based on the four operators A, E, R, and P, we define the four basic classes of infinitary properties.
An infinitary property Π is defined to be

- A *safety* property if $\Pi = A(\Phi)$ for some finitary property Φ. That is, all prefixes of a word $\sigma \in \Pi$ belong to Φ.

- A *guarantee* property if $\Pi = E(\Phi)$ for some finitary property Φ. That is, each word $\sigma \in \Pi$ is *guaranteed* to have some prefix belonging to Φ.

- A *response* property if $\Pi = R(\Phi)$ for some finitary property Φ. That is, each word $\sigma \in \Pi$ *recurrently* has (infinitely many times) prefixes belonging to Φ.

- A *persistence* property if $\Pi = P(\Phi)$ for some finitary property Φ. That is, each word $\sigma \in \Pi$ *persistently* has (continuously from a certain point on) prefixes belonging to Φ.

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

If we interpret the "good" and "bad" things mentioned in Lamport's informal definition of safety and liveness properties as situations or occurrences that can be detected in finite time, then they must correspond to finitary properties. Consequently, we can view the four basic classes as making different claims about the frequency of occurrences of "good" things. According to this interpretation, safety, guarantee, response, and persistence claim, respectively, that a "good" thing occurs always, at least once, infinitely many times, or continuously from a certain point on.

Duality of the Basic Classes

A direct consequence of the duality between the operators A and E and the operators R and P is a corresponding duality between the respective classes. This can be expressed by

- Π is a safety property iff $\overline{\Pi}$ is a guarantee property.
- Π is a response property iff $\overline{\Pi}$ is a persistence property.

2.2 Closure of the Basic Classes

Next, we show that each of the four basic classes is closed under the positive boolean operations, namely, union and intersection. We will consider each class in turn.

Closure of the Guarantee Class

Let $E(\Phi_1)$ and $E(\Phi_2)$ be two guarantee properties. We wish to show that their union is also a guarantee property. This is based on the equality

$$E(\Phi_1) \cup E(\Phi_2) = E(\Phi_1 \cup \Phi_2),$$

which appears even more convincing when we write $\Phi \cdot \Sigma^\omega$ for $E(\Phi)$. In this representation, the equality above assumes the form

$$\Phi_1 \cdot \Sigma^\omega \cup \Phi_2 \cdot \Sigma^\omega = (\Phi_1 \cup \Phi_2) \cdot \Sigma^\omega.$$

Next, consider the case of intersection. Here, we base our argument on the equality

$$\Phi_1 \cdot \Sigma^\omega \cap \Phi_2 \cdot \Sigma^\omega = (\Phi_1 \cdot \Sigma^* \cap \Phi_2 \cdot \Sigma^*) \cdot \Sigma^\omega,$$

which can easily be verified. Expressing this equality in terms of the operators E and E_f, we obtain

$$E(\Phi_1) \cap E(\Phi_2) = E(E_f(\Phi_1) \cap E_f(\Phi_2)).$$

This shows that the intersection of two guarantee properties can be expressed as the application of the operator E to the finitary property $E_f(\Phi_1) \cap E_f(\Phi_2)$. It follows that the intersection of guarantee properties is also a guarantee property.

Closure of the Safety Class

The closure of the safety class under intersection and union is established by the following two equalities, both of which can be derived by duality from the corresponding equalities for the guarantee class.

$$\begin{aligned} A(\Phi_1) \cap A(\Phi_2) &= A(\Phi_1 \cap \Phi_2) \\ A(\Phi_1) \cup A(\Phi_2) &= A(A_f(\Phi_1) \cup A_f(\Phi_2)). \end{aligned}$$

Closure of the Response Class

We consider first the simpler case of union. It is not difficult to see that

$$R(\Phi_1) \cup R(\Phi_2) = R(\Phi_1 \cup \Phi_2).$$

This equality states the obvious fact that a word contains either infinitely many Φ_1-prefixes or infinitely many Φ_2-prefixes iff it contains infinitely many $(\Phi_1 \cup \Phi_2)$-prefixes.

For the more difficult case of intersection, we introduce the following definition.

Let Φ_1 and Φ_2 be finitary properties. We define the *minimal extension* of Φ_2 over Φ_1, denoted by $minex(\Phi_1, \Phi_2)$, to be the set of words $\sigma_2 \in \Phi_2$ such that

- There exists a word $\sigma_1 \in \Phi_1$, such that $\sigma_1 \prec \sigma_2$, i.e., σ_2 is a proper Φ_2-extension of σ_1, and
- There is no $\sigma_2' \in \Phi_2$ such that $\sigma_1 \prec \sigma_2' \prec \sigma_2$, i.e., σ_2 is a *minimal* proper Φ_2-extension of σ_1.

Clearly, $minex(\Phi_1, \Phi_2) \subseteq \Phi_2$, and is therefore a finitary property.

As an example, let $\Phi_1 = (a^3)^+$ and $\Phi_2 = (a^2)^+$. Then $minex(\Phi_1, \Phi_2)$ is equal to $(a^6)^+ a^2 + (a^6)^* a^4$. On the other hand, $minex(\Phi_2, \Phi_1) = (a^6)^+ + (a^6)^* a^3 = \Phi_1$.

Now we can express the effect of intersecting two response properties as

$$R(\Phi_1) \ \cap \ R(\Phi_2) \ = \ R(minex(\Phi_1, \Phi_2)).$$

We show inclusion in the two directions. Consider an infinite word $\sigma \in R(\Phi_1) \cap R(\Phi_2)$. Let $\sigma_0^1 \prec \sigma_1^1 \prec \sigma_2^1 \prec \cdots$ be the sequence of Φ_1-prefixes in σ. For each $i = 0, 1, \ldots$, let σ_i^2 be the *shortest* Φ_2-prefix of σ that properly extends σ_i^1. Clearly, $\sigma_i^2 \in minex(\Phi_1, \Phi_2)$ and there are infinitely many of them. It follows that $\sigma \in R(minex(\Phi_1, \Phi_2))$.

In the other direction, assume that $\sigma \in R(minex(\Phi_1, \Phi_2))$. Let $\sigma_0^2 \prec \sigma_1^2 \prec \sigma_2^2 \prec \cdots$ be the sequence of $minex(\Phi_1, \Phi_2)$-prefixes in σ. Clearly, by the definition of $minex$, each of them belongs also to Φ_2, which shows that $\sigma \in R(\Phi_2)$. For each $i = 0, 1, \ldots$, let σ_i^1 be the *longest* proper Φ_1-prefix of σ_i^2. From the definition of minimal extension it follows that, for each $i = 0, 1, \ldots$, σ_i^2 is the minimal Φ_2-extension of σ_i^1. Obviously, $\sigma_0^1 \preceq \sigma_1^1 \preceq \sigma_2^1 \preceq \cdots$ but it remains to show that the containment is strict. Assume it is not, and let $\sigma_j^1 = \sigma_{j+1}^1$ for some j. Then we have the relations

$$\sigma_j^1 \ = \ \sigma_{j+1}^1 \ \prec \ \sigma_j^2 \ \prec \ \sigma_{j+1}^2.$$

These show that σ_{j+1}^2 is not the minimal proper Φ_2-extension of σ_{j+1}^1, contrary to the definition of σ_{j+1}^2. It follows that $\sigma_0^1 \prec \sigma_1^1 \prec \sigma_2^1 \prec \cdots$, and therefore $\sigma \in R(\Phi_1)$.

Closure of the Persistence Class

The closure of the persistence class under intersection and union is established by the following two equalities, both of which can be derived by duality from the corresponding equalities for the response class.

$$\begin{aligned} P(\Phi_1) \ \cap \ P(\Phi_2) \ &= \ P(\Phi_1 \cap \Phi_2) \\ P(\Phi_1) \ \cup \ P(\Phi_2) \ &= \ P(\overline{minex(\overline{\Phi}_1, \overline{\Phi}_2)}). \end{aligned}$$

2.3 Characterization of the Lower Classes

Our definition of the basic classes is *constructive*. This means that we have shown how each of the classes can be constructed by applying the operators A, E, R, and P to finitary properties. In some cases this definition is not easy to apply directly. Consider, for example, the question: How do you show that the property $(a^*b)^\omega$ is *not* a safety property? Going back to the constructive definition, we have to show that there cannot exist a finitary property Φ such that $(a^*b)^\omega = A(\Phi)$. On the face of it, this does not seem to be an easy task.

It is therefore very helpful to derive additional characterizations of the properties, which are independent of the constructive definitions. We present such characterizations for the lower classes of safety and guarantee.

Proposition 2.1 *An infinitary property* Π *is a safety property iff*

$$\Pi \;=\; A(\mathit{Pref}(\Pi)).$$

Proof: Clearly, $\mathit{Pref}(\Pi)$ is a finitary property. If $\Pi = A(\mathit{Pref}(\Pi))$, then by definition, Π is a safety property.

Now assume that Π is a safety property. It is easy to see that $\Pi \subseteq A(\mathit{Pref}(\Pi))$, since for any word $\sigma \in \Pi$ all the prefixes of σ belong to $\mathit{Pref}(\Pi)$.

By the assumption that Π is a safety property it can be presented as $\Pi = A(\Phi)$ for some finitary Φ. By the definition of the A operator it follows that $\mathit{Pref}(\Pi) \subseteq \Phi$. Applying the operator A, which is clearly monotonic, to both sides of this inclusion yields $A(\mathit{Pref}(\Pi)) \subseteq A(\Phi) = \Pi$. This establishes the other direction of the equality. ∎

We may now use the above characterization to show that $(a^*b)^\omega$ is *not* a safety property. A simple calculation yields $\mathit{Pref}((a^*b)^\omega) = (a+b)^+$, from which we get

$$\begin{aligned} A\Big(\mathit{Pref}((a^*b)^\omega)\Big) \;&=\; \\ A((a+b)^+) \;&=\; (a+b)^\omega \;\neq\; (a^*b)^\omega. \end{aligned}$$

We refer to the operator $A(\mathit{Pref}(\Pi))$, applied to an infinitary property Π, as the *safety closure* of Π. Thus the claim above can be reformulated by saying that an infinitary property Π is a safety property iff Π equals its safety closure.

By duality we can immediately obtain a characterization of the guarantee class.

Proposition 2.2 *An infinitary property* Π *is a guarantee property iff*

$$\Pi \;=\; E(\overline{\mathit{Pref}(\overline{\Pi})}).$$

Note that the complement of Π should be taken with respect to Σ^ω, while the complement of $\mathit{Pref}(\overline{\Pi})$ should be taken with respect to Σ^+.

2.4 Inclusion among the Classes

Another interesting relation among the classes is that of inclusion, which arranges the classes of properties in a hierarchy. The two classes of response and persistence properties are higher in the hierarchy in the sense that they properly contain the classes of safety and guarantee.

Response contains Safety and Guarantee

To show containment of the safety class in the response class, we have to show that any safety property $\Pi = A(\Phi)$ can be presented as a response property, i.e., as the application of the operator R to some finitary property. This is easily accomplished by the equality

$$A(\Phi) \;=\; R(A_f(\Phi)).$$

This equality states that all prefixes of σ are in Φ iff σ has infinitely many prefixes σ', such that all the prefixes of σ' are in Φ.

To show that the containment is strict it suffices to consider the property $(a^*b)^\omega$, which consists of all words whose states are either a or b, but have an infinite number of b's. It is easily seen that $(a^*b)^\omega$ is a response property, as it can be presented as $R((a^*b)^+)$. On the other hand, as shown above, this property is not a safety property.

To show that any guarantee property, presentable as $\Pi = E(\Phi)$, is also a response property, we use the equality

$$E(\Phi) \;=\; R(E_f(\Phi)).$$

It is easy to recognize the equality once we present the equality in the equivalent form

$$\Phi \cdot \Sigma^\omega \;=\; R(\Phi \cdot \Sigma^*).$$

To show that the inclusion is strict, we again use the response property $(a^*b)^\omega$. It remains to show it is not a guarantee property, using the appropriate characterization claim. For simplicity, we assume that $\Sigma = \{a, b\}$.

$$\begin{aligned} E\Big(\overline{Pref(\overline{(a^*b)^\omega})}\Big) \;&=\; E\Big(\overline{Pref((a+b)^*a^\omega)}\Big) \\ =\; E(\overline{(a+b)^+}) \;&=\; E(\emptyset) = \emptyset \;\neq\; (a^*b)^\omega \end{aligned}$$

Persistence contains Safety and Guarantee

By duality, we can use the previous results to show that any safety and guarantee properties are presentable as persistence properties. The equalities on which these presentations are based are

$$\begin{aligned} A(\Phi) \;&=\; P(A_f(\Phi)) \\ E(\Phi) \;&=\; P(E_f(\Phi)). \end{aligned}$$

To show the strictness of the containment we may use the property complementing the response property $(a^*b)^\omega$ from above. This is the persistence property $(a+b)^*a^\omega$. By duality, this property belongs to neither the safety nor the guarantee class.

2.5 The Compound Classes

The four classes we have introduced are considered the *basic classes*. As we have seen, each of the basic classes is closed with respect to the *positive* boolean operations of union and intersection, but taking the complement moves us from each class to its dual (safety $\leftrightarrow$ guarantee and response $\leftrightarrow$ persistence).

There are two additional classes, which we refer to as the *compound* classes, that can be obtained by taking unrestricted boolean combinations of the basic classes.

- The Obligation Class.

  This class can be defined (by three equivalent statements) as the class obtainable by

  - Unrestricted boolean combinations of safety properties, or
  - Unrestricted boolean combinations of guarantee properties, or
  - Positive boolean combinations of safety and guarantee properties.

- The Reactivity Class.

  This class can be defined (by three equivalent statements) as the class obtainable by

  - Unrestricted boolean combinations of response properties, or
  - Unrestricted boolean combinations of persistence properties, or
  - Positive boolean combinations of response and persistence properties.

The definitions above display an obvious tradeoff between using unrestricted boolean combinations of a single class or using positive boolean combinations of a class and its dual.

A typical obligation property is given by the expression $a^\omega + \Sigma^* \cdot c \cdot \Sigma^\omega$ which represents a union of the safety property a^ω and the guarantee property $\Sigma^* \cdot c \cdot \Sigma^\omega$.

The Obligation Class

The obligation class obviously contains both the safety and guarantee class. By examining the property $a^\omega + \Sigma^* \cdot c \cdot \Sigma^\omega$ we see that the containment is strict, since this property is neither a safety property nor a guarantee property.

The obligation class is obviously closed under all three boolean operations. Using the third version of the definition, we can use the distributive rule to bring any boolean combination into a conjunctive normal form

$$\bigcap_{i=1}^{n}(\Pi_0^i \cup \cdots \cup \Pi_{k_i-1}^i \cup \Pi_{k_i}^i \cup \cdots \cup \Pi_{m_i-1}^i)$$

where $\Pi_0^i, \ldots, \Pi_{k_i-1}^i$ are safety properties, and $\Pi_{k_i}^i, \ldots, \Pi_{m_i-1}^i$ are guarantee properties. Using the closure of the safety and guarantee classes under union, we can replace the union $\Pi_0^i \cup \cdots \cup \Pi_{k_i-1}^i$ by a single safety property Π_S^i, and the union $\Pi_{k_i}^i \cup \cdots \cup \Pi_{m_i-1}^i$ by a single guarantee property Π_G^i. It follows that any obligation property can be presented as the intersection

$$\bigcap_{i=1}^{n}(A(\Phi_i) \cup E(\Psi_i)),$$

for some $n > 0$, and finitary properties $\Phi_1, \Psi_1, \ldots, \Phi_n, \Psi_n$. We refer to this representation as the *conjunctive normal form* of obligation properties.

In a completely symmetric way we can present each obligation property in a *disjunctive normal form*

$$\bigcup_{i=1}^{n}(A(\Phi_i) \cap E(\Psi_i)).$$

We will mainly use the conjunctive normal form.

Either of these forms introduces a strict hierarchy within the class of obligation properties. We define the subclass of m-obligation properties, for $m = 1, \ldots,$ to consist of all properties that have a conjunctive normal from representation with $n = m$. It is not difficult to see that the subclass of m-obligation properties is contained in the subclass of $(m+1)$-obligation properties.

Less obvious is the fact that this is a strict hierarchy. Define Σ_m to be $[0..2]^m$, i.e., elements of the form $\langle i_1, \ldots, i_m \rangle$ where each $i_j \in \{0, 1, 2\}$. For a letter $a = \langle i_1, \ldots, i_m \rangle \in \Sigma_m$, and a number $j = 1, \ldots, m$, we define the j-th projection of a, denoted by $\pi_j(a)$, to be i_j. For an infinite word $\sigma = a_1, a_2, \ldots$ we define the j-th projection of σ, denoted by $\pi_j(\sigma)$, to be the infinite word $\pi_j(a_1), \pi_j(a_2), \ldots \in [0..2]^\omega$.

For a given $m > 0$, define the language O_m to be the subset of Σ_m^ω such that $\sigma \in O_m$ iff for every $j = 1, \ldots, m$, $\pi_j(\sigma) \in 1^\omega \cup [0..2]^* \cdot 2 \cdot [0..2]^\omega = A(1^+) \cup E([0..2]^* \cdot 2)$. Then O_m is an m-obligation property, but not an m'-obligation property for any $m' < m$. We will defer any proof of this until section 5, where we will give an argument using automata. Our use of the variable size alphabet Σ_m is merely a convenience; we can clearly encode Σ_m by a binary alphabet.

Note that the safety and guarantee properties belong to the subclass of 1-obligation properties, which we refer to as the subclass of *simple obligation* properties.

The last property of the obligation class we wish to discuss is its strict containment in both the response and persistence classes. Observe that the definition of obligation properties as a positive boolean combination of safety and guarantee properties can be recast into an inductive definition as follows.

- Every safety property is an obligation property.
- Every guarantee property is an obligation property.
- If Π_1 and Π_2 are obligation properties, then so are $\Pi_1 \cup \Pi_2$ and $\Pi_1 \cap \Pi_2$.

Based on this definition, it is easy to prove by induction that every obligation property is a response property. This is because every safety property and every guarantee property are response properties, and the union and intersection of response properties are, again, response properties. To show that the containment is strict, we may again use the property $(a^*b)^\omega$, which is a response property but can be shown not to be an obligation property.

An identical argument shows that the obligation class is contained in the persistence class. The property $(a+b)^*a^\omega$, which is a persistence property that can be shown not to be an obligation property, shows that the containment is strict.

The Reactivity Class

There is a very close analogy between the way the obligation class is constructed by boolean combinations of safety and guarantee properties, and the way the reactivity class is constructed by boolean combinations of response and persistence properties. We can therefore transliterate all the properties established for the obligation class into corresponding properties of the reactivity class.

Every reactivity property is presentable in a *conjunctive normal form*

$$\bigcap_{i=1}^{n} (R(\Phi_i) \cup P(\Psi_i)),$$

for some $n > 0$, and finitary properties $\Phi_1, \Psi_1, \ldots, \Phi_n, \Psi_n$.

Similarly, every reactivity property is presentable in a *disjunctive normal form*

$$\bigcup_{i=1}^{n} (R(\Phi_i) \cap P(\Psi_i)),$$

for some $n > 0$.

Either of these forms introduces a strict hierarchy within the class of reactivity properties. We define the subclass of m-reactivity properties, for $m = 1, \ldots,$ to consist of all properties that have a conjunctive normal from representation with $n = m$. It is not difficult to see that the subclass of m-reactivity properties is contained in the subclass of $(m+1)$-reactivity properties.

Less obvious is the fact that this is a strict hierarchy. Let Σ_m be $[0..2]^m$, as defined earlier for obligation properties. Then for a given $m > 0$, define the language $\mathcal{E}_m$ to be the subset of Σ_m^ω such that $\sigma \in \mathcal{E}_m$ iff for every $j = 1, \ldots, m$, $\pi_j(\sigma) \in ([0..2]^* \cdot 2)^\omega \cup [0..2]^* \cdot 1^\omega = R([0..2]^* \cdot 2) \cup P([0..2]^* \cdot 1)$. Then $\mathcal{E}_m$ is an m-reactivity property, but is not an m'-reactivity property for any $m' < m$. Again, we defer proving this until section 5.

Both response and persistence properties belong to the subclass of 1-reactivity properties, which we refer to as the subclass of *simple reactivity properties.*

The class of reactivity properties is closed under all three boolean operations.

In Figure 1, we present a diagram that displays the six classes we have discussed and the containment relations that hold between them. The diagram also includes notation from the topological and temporal logic classifications which will be introduced later.

2.6 The Safety-Liveness Classification

As we have already mentioned, there exists another classification that partitions the set of all properties into two *disjoint* classes, the class of *safety* and the class of *liveness* (see [Lam83], [AS85]).

The definition of the safety class is the one we have used before, or equivalently, the characterization of safety properties by

$$\Pi \;=\; A(\mathit{Pref}(\Pi)).$$

An infinitary property Π is defined to be a *liveness* property if every finite word $\sigma \in \Sigma^+$ is a prefix of a word in Π, i.e., $\mathit{Pref}(\Pi) = \Sigma^+$. We define the *liveness extension* of Π by

$$\mathcal{L}(\Pi) \;=\; \Pi \;\cup\; E(\overline{\mathit{Pref}(\Pi)}).$$

Thus, $\mathcal{L}(\Pi)$ consists of all the words of Π plus all the words that have at least one prefix that cannot be extended to a word of Π. Since for every finitary Φ, $\Phi \subseteq \mathit{Pref}(E(\Phi))$,

$$\begin{aligned} \mathit{Pref}(\Pi) \;\cup\; \overline{\mathit{Pref}(\Pi)} \;&\subseteq\; \mathit{Pref}(\Pi) \;\cup\; \mathit{Pref}\Big(E(\overline{\mathit{Pref}(\Pi)})\Big) \\ \Sigma^+ \;&\subseteq\; \mathit{Pref}(\mathcal{L}(\Pi)) \end{aligned}$$

This shows that $\mathcal{L}(\Pi)$ is a liveness property. It can be shown that $\mathcal{L}(\Pi)$ is the smallest liveness property that contains Π. In particular, if Π is already a liveness property, then $\mathcal{L}(\Pi) = \Pi$.

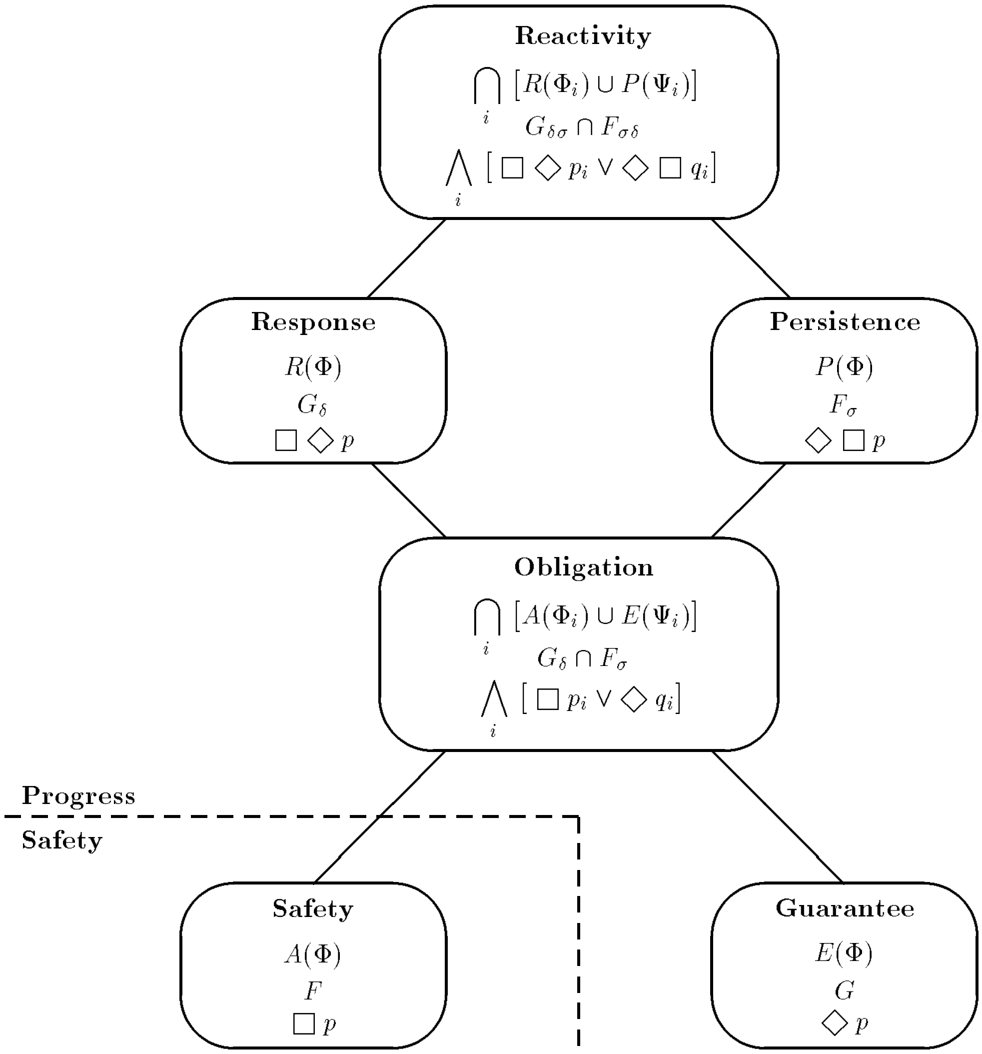

Figure 1: Inclusion Relations between the Classes

Incidentally, liveness properties are *upwards-closed*. This means that if Π is a liveness property, then so is any $\Pi' \supseteq \Pi$.

One of the important aspects of the safety-liveness classification is that it provides an exhaustive partition of the class of all properties. Let $\Sigma = \{a, b, c\}$. Consider the property $\Pi = a^*b \cdot \Sigma^\omega$. It is certainly not a safety property, as it does not equal its safety closure, which is $a^\omega \cup a^*b \cdot \Sigma^\omega$. Neither is Π a liveness property since it is different from its liveness extension, which is

$$\begin{aligned}
\Pi \cup E(\overline{Pref(\Pi)}) &= \Pi \cup E\left(\overline{a^+ + a^*b \cdot \Sigma^*}\right) \\
&= \Pi \cup E(a^*c \cdot \Sigma^*) \\
&= a^*b \cdot \Sigma^\omega + a^*c \cdot \Sigma^\omega \\
&= a^*(b+c) \cdot \Sigma^\omega.
\end{aligned}$$

However, Π can be represented as the intersection of its safety closure and its liveness extension.

Clearly the classes of safety and liveness are disjoint, except for the trivial properties $\emptyset$ and Σ^ω. Thus, the important fact about the safety-liveness classification is stated in the following proposition.

Proposition 2.3 *Every property* Π *can be represented as the intersection*

$$\Pi \ = \ \Pi_S \ \cap \ \Pi_L,$$

where Π_S *is a safety property and* Π_L *is a liveness property.*

Proof: To prove this claim we take Π_S to be the safety closure of Π, $\Pi_S = A(Pref(\Pi))$. For Π_L we take the liveness extension of Π, $\mathcal{L}(\Pi)$. Now, we show that $\Pi = \Pi_S \cap \Pi_L$. We use the definitions of Π_S and Π_L and the distribution of intersection over union to get

$$\Pi_S \ \cap \ \Pi_L = \Big[A(Pref(\Pi)) \cap \Pi\Big] \ \cup \ \Big[A(Pref(\Pi)) \cap E(\overline{Pref(\Pi)})\Big].$$

Since every infinitary property is contained in its safety closure, it follows that $\Pi \subseteq A(Pref(\Pi))$, and hence $A(Pref(\Pi)) \cap \Pi = \Pi$. The equality $A(\Phi) \cap E(\overline{\Phi}) = \emptyset$ is true for every finitary property Φ, in particular for $\Phi = Pref(\Pi)$, which leads to the fact that the second intersection is empty. This shows that

$$\Pi_S \ \cap \ \Pi_L \ = \ \Pi. \ \blacksquare$$

We can identify within the liveness class the same hierarchy we have previously introduced. Let κ stand for the name of any of the five non-safety classes, i.e., guarantee, obligation, response, persistence, or reactivity. We define a property to be a *live* κ-property if it is a liveness property that also belongs to the κ class.

An interesting observation is that if Π is a property of class κ, then its liveness extension $\mathcal{L}(\Pi)$ is a live κ-property. This is because $\mathcal{L}(\Pi)$ is formed by the union of Π with the guarantee property $E(\overline{Pref(\Pi)})$, and all the non-safety classes are closed under union with guarantee properties. Therefore we have the following proposition:

Proposition 2.4 *Any* κ*-property* Π *is representable as the intersection*

$$\Pi \ = \ \Pi_S \ \cap \ \Pi_L,$$

where Π_S *is a safety property and* Π_L *is a live* κ*-property, and* $\kappa \in \{$ *guarantee, obligation, response, persistence, reactivity* $\}$.

This proposition shows that, in some sense, the safety-progress and the safety-liveness classifications are orthogonal to one another.

A special class of liveness properties is the class of *uniform liveness* properties. A property Π is defined to be a *uniform liveness* property if there exists a *single* infinite word $\sigma' \in \Sigma^\omega$, such that $\Sigma^+ \cdot \sigma' \subseteq \Pi$. That is, for every arbitrary $\sigma \in \Sigma^+$, $\sigma \cdot \sigma' \in \Pi$. Note, in comparison, that liveness only requires that, for every arbitrary $\sigma \in \Sigma^+$, there exists *some* $\sigma' \in \Sigma^\omega$ such that $\sigma \cdot \sigma' \in \Pi$.

As an example of a property that is a liveness property but not a uniform liveness property, consider the property $a \cdot \Sigma^* \cdot aa \cdot \Sigma^\omega + b \cdot \Sigma^* \cdot bb \cdot \Sigma^\omega$ over an alphabet $\Sigma = \{a, b\}$. This property requires that the state that appears first in the word appears sometime later, twice in succession. Let σ be any finite word. If σ begins in an a-state, then the proper extension σ' is $aa \cdot \Sigma^\omega$. If σ begins in a b-state, then the proper extension is $bb \cdot \Sigma^\omega$. Clearly, there does not exist a uniform extension σ' that applies to all words $\sigma \in \Sigma^+$.

3 The Topological View

In this section we characterize the different classes in the hierarchy by their topological properties. We will show that the classes of properties in our hierarchy correspond precisely to the lower two (and a half) levels of the Borel topological hierarchy.

First, let us define the appropriate topological notions. The *distance* $\mu(\sigma, \sigma')$ between infinite words σ and σ' is defined to be 0 if they are identical, and

$$\mu(\sigma, \sigma') = 2^{-j}$$

otherwise, where j is the length of the longest prefix on which they agree. For example, for every $n > 0$, $\mu(a^n b^\omega, a^{2n} b^\omega) = 2^{-n}$, since the longest prefix on which these two words agree is a^n of length n.

It is not difficult to see that the distance function μ has all the properties required of a *metric*, and that with this metric, the set Σ^ω becomes a complete metric space.

Following the standard definitions, we say that an infinite sequence of words (each of which is an infinite sequence of states) $\sigma_0, \sigma_1, \ldots$ *converges* to the *limit* σ if the distance $\mu(\sigma, \sigma_k)$ tends to 0 as k goes to infinity. In other words, the length of the maximal prefix common to σ and to σ_k goes to infinity with k.

Consider, for example, the sequence of words $b^\omega, ab^\omega, aab^\omega, aaab^\omega, \ldots$. It is not difficult to see that this sequence converges to the limit a^ω. This is because the sequence of longest prefixes common to a^ω and to $\sigma_k = a^k b^\omega$, which is a^k, gets increasingly longer with k.

Given a set $U \subset \Sigma^\omega$, we define the word $\sigma \in \Sigma^\omega$ to be a *limit point* of the set U if there exists an infinite sequence of words $\sigma_0, \sigma_1, \sigma_2, \ldots$, all of which belong to U, that converges to σ. Clearly, any $\sigma \in U$ is a limit point of U, since the sequence $\sigma, \sigma, \sigma, \ldots$ converges to σ.

We define the (topological) *closure* of U, denoted by $cl(U)$, to be the set of all limit points of U. Obviously, $U \subseteq cl(U)$. For example, the closure of the set $a^+ b^\omega$ is given by $cl(a^+ b^\omega) = a^+ b^\omega + a^\omega$, which consists of the original set plus the limit word a^ω.

We define a set U to be *closed* if it contains all its limit points, i.e., $cl(U) \subseteq U$, which immediately yields $cl(U) = U$. Thus, $a^+ b^\omega + a^\omega$ is a closed set, while $a^+ b^\omega$ is not.

We proceed to consider each of the classes of properties we have introduced above, giving each of them a topological characterization. We will start with the four basic classes.

3.1 Safety Properties as *Closed* sets

To show the correspondence between safety properties and closed sets, we prove the following general identity.

Proposition 3.1 *For any set* $\Pi \subseteq \Sigma^\omega$, $cl(\Pi) = A(\mathit{Pref}(\Pi))$.

Proof: Let $\sigma \in cl(\Pi)$. This means that there exists an infinite sequence $\sigma_0, \sigma_1, \ldots$ of words in Π whose limit is σ. Let $\hat{\sigma} \prec \sigma$ be an arbitrary prefix of σ. By the definition of a sequence converging to a limit, there must exist some $k \geq 0$ such that σ and σ_k share a common prefix σ' which is *longer* than $\hat{\sigma}$. It follows that $\hat{\sigma} \prec \sigma' \prec \sigma_k$, and therefore $\hat{\sigma} \in \mathit{Pref}(\Pi)$. Thus, any prefix of σ is in $\mathit{Pref}(\Pi)$, and consequently $\sigma \in A(\mathit{Pref}(\Pi))$.

In the other direction, let $\sigma \in A(\mathit{Pref}(\Pi))$. For each $k > 0$, let $\hat{\sigma}_k$ denote the prefix of σ of length k. By the assumption, each $\hat{\sigma}_k$ belongs to $\mathit{Pref}(\Pi)$, and therefore there exists a $\sigma_k \in \Pi$, such that $\hat{\sigma}_k \prec \sigma_k$. We claim that the sequence $\sigma_1, \sigma_2, \ldots$ converges to σ. This is because, for any length L, it is sufficient to take $k > L$ to obtain a member of the sequence, namely σ_k, which shares with σ at least the prefix $\hat{\sigma}_k$, whose length exceeds L. Thus, σ is a limit point of Π. ▪

Having established the equivalence of the operators $cl(\Pi)$ and $A(\mathit{Pref}(\Pi))$, the following proposition is obvious:

Proposition 3.2 Π *is a safety property iff it is a closed set.*

The closure of the class of closed sets with respect to finite unions and arbitrary intersections is a known topological fact, and corresponds to the similar closure properties we established in the language-theoretic view. In the inclusion diagram we represent the topological characterization of this class by the letter F, which usually designates the class of closed sets.

3.2 Guarantee Properties as *Open* sets

Following the standard topological definitions, a property Π is defined to be *open* in our topology if for every $\sigma \in \Pi$, there exists an $\epsilon > 0$ such that $\mu(\sigma, \sigma') < \epsilon$ implies $\sigma' \in \Pi$, for every $\sigma' \in \Sigma^\omega$. Intuitively, all the words that are close enough to σ are contained in Π. It is not difficult to see that Π is an open set iff $\overline{\Pi}$ is closed. Thus the families of closed and open sets are *dual*.

Proposition 3.3 Π *is a guarantee property iff it is an open set.*

Proof: We can prove this by duality. Π is a guarantee property iff $\overline{\Pi}$ is a safety property iff $\overline{\Pi}$ is a closed set iff Π is an open set.

However, it may be instructive to present a direct proof of this proposition. Let $\Pi = E(\Phi)$ be a guarantee property. Let $\sigma \in \Pi$ be any word in Π. By definition, σ has a finite prefix $\hat{\sigma} \prec \sigma$ which belongs to Φ. Assume its length to be $L > 0$. Then we claim that all infinite words which share with σ a prefix longer than L are also in Π. Let σ' be such an infinite word. Since σ and σ' share a prefix longer than L, σ' must also have $\hat{\sigma}$ as a prefix. Consequently, $\sigma' \in \Pi$.

In the other direction, Let Π be an open set. Take any infinite word $\sigma \in \Pi$. Since Π is open, there must be some integer $L > 0$, such that all infinite words that share with σ a prefix longer than L are in Π. Let $\hat{\sigma}$ be the prefix of length $L+1$ of σ. We define Φ to be the set of all such prefixes. It is not difficult to see that $E(\Phi) = \Pi$. ▪

In the inclusion diagram we represent the topological characterization of this class by the letter G, which usually designates the class of open sets.

3.3 Response Properties as G_δ sets

A set is defined to belong to the G_δ family if it can be obtained as a countable intersection of open sets. Note that, if we take only a finite intersection of open sets, we still obtain an open set.

Consider, for example, the sequence of open sets

$$G_1 : (a^*b) \cdot \Sigma^\omega \ , \ G_2 : (a^*b)^2 \cdot \Sigma^\omega \ , \ G_3 : (a^*b)^3 \cdot \Sigma^\omega \ , \ \ldots$$

where we assume $\Sigma = \{a, b\}$. Clearly G_k is the set of all words that have at least k occurrences of the letter b. It is not difficult to see that the intersection of the sequence of these sets yields the set $H : (a^*b)^\omega$, that consists of all the words having an infinite number of b's. It can easily be shown that the set H is neither open nor closed, but is by definition a G_δ set.

Proposition 3.4 *Π is a response property iff it is a G_δ set.*

Proof: Assume first that Π is a response property, i.e., $\Pi = R(\Phi)$ for some finitary Φ. For every $k > 0$ define the set Φ_k to be the set of words $\sigma \in \Phi$ such that σ contains at least $k-1$ distinct proper prefixes that also belong to Φ. For each $k > 0$ define $G_k = \Phi_k \cdot \Sigma^\omega = E(\Phi_k)$. It is clear that the G_k's are open, and that

$$\Pi = \bigcap_k G_k.$$

In the other direction, let Π be a G_δ set, i.e., $\Pi = \bigcap_k G_k$ where the sets $G_0, G_1, \ldots$ are open. Each G_k can be represented as $G_k = \Phi_k \cdot \Sigma^\omega$ for some finitary Φ_k. Consider an arbitrary $\sigma \in \Pi$. For every $k \geq 0$, σ must have a prefix $\hat{\sigma}_k$ belonging to Φ_k. Without loss of generality we may assume that $\hat{\sigma}_k$ is the shortest prefix of σ that belongs to Φ_k. There are two cases to be considered. In the first case, the lengths of the prefixes $\hat{\sigma}_k$ are bounded. In that case there exists a single prefix $\hat{\sigma}$ such that $\hat{\sigma}_k \preceq \hat{\sigma}$ for all k. Defining a finitary set $\Phi = \bigcap_k(\Phi_k \cdot \Sigma^*)$, we can easily see that $\hat{\sigma}$ belongs to Φ. Thus, in the bounded case, σ belongs to $\Phi \cdot \Sigma^\omega = E(\Phi)$. Next consider the case that the lengths of the prefixes $\hat{\sigma}_k$ are not bounded. Let us define a new set of prefixes $\sigma'_0, \sigma'_1, \ldots$, where σ'_k is the shortest prefix of σ such that $\hat{\sigma}_0 \preceq \sigma'_k, \ldots, \hat{\sigma}_k \preceq \sigma'_k$. It is not difficult to see that the lengths of the σ'_k are also unbounded. Define the sets $\Psi_k = \bigcap_{i=0}^k(\Phi_i \cdot \Sigma^*)$. Let Γ_k be the *prefix free* subset of Ψ_k, i.e., the set of words $\hat{\sigma}$ belonging to Ψ_k, such that no proper prefix of $\hat{\sigma}$ belongs to Ψ_k. Clearly, each σ'_k belongs to Γ_k. If we denote the union of the Γ_k's by $\Gamma = \bigcup_k \Gamma_k$ then, in the unbounded case, $\sigma \in R(\Gamma)$ since σ has unboundedly many prefixes belonging to Γ. We may summarize the two cases by

$$\Pi \subseteq E(\Phi) \cup R(\Gamma),$$

for the finitary sets Φ and Γ defined above.

It is not difficult to see that $E(\Phi) \subseteq \Pi$. For the second component, consider some word $\sigma \in R(\Gamma)$. This word has infinitely many prefixes, each belonging to some Γ_k. Since the sets Γ_k are prefix free, the infinitely many prefixes must belong to infinitely many different Γ_k's. For any $i \geq 0$, there must be a prefix $\hat{\sigma} \prec \sigma$ belonging to Γ_k for some $k \geq i$. By the definitions of Γ_k and Ψ_k, it follows that $\hat{\sigma} \in \Phi_i \cdot \Sigma^*$. Thus $\sigma \in \Phi_i \cdot \Sigma^\omega$ for every i. Consequently, $R(\Gamma) \subseteq \Pi$. We may conclude

$$\Pi = E(\Phi) \cup R(\Gamma).$$

Therefore, every G_δ set is a response property. ▪

3.4 Persistence Properties as F_σ sets

A set is defined to belong to the F_σ family if it can be obtained as the countable union of closed sets. Obviously, a set U is F_σ if and only if its complement $\overline{U}$ is G_δ.

Proposition 3.5 *Π is a persistence property iff it is a G_σ set.*

This proposition follows from the previous proposition by duality.

3.5 The Topological Characterization of Liveness

Following [AS85], we provide a topological characterization of the safety-liveness classification. We have already shown that the safety class corresponds to the family of closed sets.

As claimed in [AS85], the class of liveness properties corresponds to the family of *dense* sets. Following the standard topological definitions, a property Π is defined to be *dense* in our topology if, for every $\sigma \in \Sigma^\omega$ and any $\epsilon > 0$, there exists a word $\sigma' \in \Pi$ such that $\mu(\sigma, \sigma') < \epsilon$. Thus, members of Π exist arbitrarily close to any word $\sigma \in \Sigma^\omega$.

To show that every liveness property Π is dense, let σ be an arbitrary word and $\epsilon > 0$. By the definition of the distance function μ, there is some L such that, for every word σ' that shares a common prefix with σ of length L, $\mu(\sigma, \sigma') < \epsilon$. By the definition of a liveness property, there is some word $\sigma'' \in \Pi$ which shares a common prefix with σ of length L, so $\mu(\sigma, \sigma'') < \epsilon$. Conversely, if there is some $\sigma' \in \Sigma^+$ that is not a prefix of any word in Π, then for any extension σ of σ' there is no $\sigma'' \in \Pi$ for which $\mu(\sigma, \sigma'') < 2^{-|\sigma'|}$. Thus every dense property is a liveness property.

4 The Temporal Logic View

After studying properties and their classification in abstract language-theoretical and topological settings, we consider a subset of all these properties, the properties expressible by temporal logic. We will show that the hierarchy introduced in the abstract setting still exists within the set of expressible properties. Furthermore, each class has an additional characterization by formulas of a special form that can express exactly the properties belonging to that class. We will consider each of the basic concepts introduced under the abstract setting and show how to restrict them to the framework of properties expressible by temporal logic.

First, we present a brief introduction to the language of temporal logic. We refer the reader to [MP91] for a comprehensive introduction to temporal logic and its use for specification of program properties.

4.1 The Language of Temporal Logic

We assume an underlying assertional language which is used to describe properties of states. We refer to a formula in the assertional language as a *state formula*, or simply as an *assertion*.

A *temporal formula* is constructed out of state formulas to which we apply the boolean operators $\neg$ and $\vee$ (the other boolean operators can be defined from these), and the following basic temporal operators:

$$\bigcirc \text{ -- Next} \qquad \mathcal{U} \text{ -- Until} \qquad \ominus \text{ -- Previous} \qquad \mathcal{S} \text{ -- Since.}$$

A *model* for a temporal formula p is an infinite sequence of states (i.e., a word)

$$\sigma : \; s_1, s_2, s_3, \ldots,$$

where each state s_j provides an interpretation for the assertions mentioned in p.

Given a model σ, as above, we present an inductive definition of a temporal formula p holding at a position $j > 0$ in σ, i.e., that j is a *p-position*, denoted by $(\sigma, j) \models p$.

For a state formula p,

$$(\sigma, j) \models p \iff s_j \models p.$$

That is, we evaluate p locally, using the interpretation given by s_j. If p holds on s_j, we say that s_j is a *p-state*.

$$\begin{array}{lcl}
(\sigma, j) \models \neg p & \iff & (\sigma, j) \not\models p \\
(\sigma, j) \models p \vee q & \iff & (\sigma, j) \models p \text{ or } (\sigma, j) \models q \\
(\sigma, j) \models \bigcirc p & \iff & (\sigma, j+1) \models p \\
(\sigma, j) \models p \,\mathcal{U}\, q & \iff & \text{for some } k,\ j \leq k, (\sigma, k) \models q, \\
 & & \text{and for every } i \text{ such that } j \leq i < k, (\sigma, i) \models p \\
(\sigma, j) \models \ominus p & \iff & j > 1 \text{ and } (\sigma, j-1) \models p \\
(\sigma, j) \models p \,\mathcal{S}\, q & \iff & \text{for some } k,\ k \leq j, (\sigma, k) \models q, \\
 & & \text{and for every } i \text{ such that } k < i \leq j, (\sigma, i) \models p
\end{array}$$

Additional temporal operators can be defined as follows:

$$\begin{array}{ll}
\Diamond p = \mathbf{T}\,\mathcal{U}\, p & \text{– Eventually} \\
\Box p = \neg \Diamond \neg p & \text{– Henceforth} \\
p \,\mathcal{W}\, q = \Box p \vee (p \,\mathcal{U}\, q) & \text{– Waiting-for, Unless (Weak Until)} \\
\diamondminus p = \mathbf{T}\,\mathcal{S}\, p & \text{– Sometimes in the past} \\
\boxminus p = \neg \diamondminus \neg p & \text{– Always in the past} \\
p \,\mathcal{B}\, q = \boxminus p \vee (p \,\mathcal{S}\, q) & \text{– Back-to (Weak Since)} \\
\widetilde{\ominus} p = \neg \ominus \neg p & \text{– Weak Previous}
\end{array}$$

Another useful derived operator is the *entailment* operator, defined by:

$$p \Rightarrow q \iff \Box(p \rightarrow q).$$

We refer to $\bigcirc$ and $\mathcal{U}$, and the operators derived from them as the *future* operators; $\ominus$, $\mathcal{S}$, and the operators derived from them are the *past* operators. A formula that contains no future operators is called a *past* formula. A formula that contains no past operators is called a *future* formula. Note that a state formula is both a past formula and a future formula. We refer to a past formula (future formula) that is not also a state formula as a *strict-past* (*strict-future*, respectively) formula.

If $(\sigma, 1) \models p$, we say that p *holds* on σ, and denote it by $\sigma \models p$. A formula p is called *satisfiable* if it holds on *some* model. A formula is called *valid* if it holds on *all* models.

Two formulas p and q are defined to be *equivalent*, denoted by $p \sim q$, if the formula $p \leftrightarrow q$ is valid, i.e., $\sigma \models p$ iff $\sigma \models q$, for all σ. Formulas p and q are defined to be *congruent* if $\Box(p \leftrightarrow q)$ is valid.

The following are simple examples of temporal formulas and their intuitive meanings as requirements on the sequences (words) that satisfy them.

- $p \rightarrow \Diamond q$
  If initially p then eventually q.

- $\Box(p \rightarrow \Diamond q)$
  Every p-position coincides with or is followed by a q-position.

- $\Box \Diamond q$
  The sequence σ contains infinitely many q-positions.

- $\Diamond \Box q$
  The sequence σ contains only finitely many $\neg q$-positions.

- $(\neg q) \mathcal{W} p$
  If there exist any q-positions, then the first q-position must coincide with or be preceded by a p-position.

- $\Box(q \rightarrow \diamondminus p)$
  Every q-position coincides with or is preceded by a p-position.

- $\neg \ominus \mathrm{T}$
  There is no previous position that satisfies T. Since all positions in the model must satisfy T, this is equivalent to, "there is no previous position." This formula uniquely characterizes the initial position of every model; hence, we refer to this formula as *first*.

Note that quantification is allowed only within state formulas.

4.2 The Temporal Hierarchy of Properties

In the temporal logic framework we will only consider properties $\Pi \subseteq \Sigma^\omega$ that can be associated with a temporal formula φ, such that Π is exactly the set of all (infinite) sequences satisfying φ. In this case, we say that Π is *expressible* (equivalently, *specifiable*) in temporal logic, and define $Sat(\varphi) = \Pi$.

Next, consider a finitary property $\Phi \subseteq \Sigma^+$. To express such properties we use a past formula p. Let $\sigma \in \Sigma^+$ be a finite sequence of length $|\sigma| = n$ and p a past formula. We say that σ *end-satisfies* p, denoted $\sigma \dashv p$, if for some infinite extension $\sigma' \in \Sigma^\omega$ such that $\sigma \prec \sigma'$, $(\sigma', n) \models p$. That is, p holds at the last position of σ within σ'. It is not difficult to see that this definition is dependent only on the first n positions of σ', i.e., only on σ. The finitary property defined by the past formula p is the set of finite sequences that end-satisfy p. For a past formula p, denote by $esat(p)$ the finitary property defined by p, i.e., the set of finite sequences that end-satisfy p. We say that a finitary property Φ is expressible (in temporal logic) if it equals $esat(p)$ for some past formula p.

For example, the finitary property a^*b can be defined by the past formula $b \wedge \widetilde{\ominus} \boxminus a$, which states that b holds now and a holds in all preceding positions.

We can now show that the four language operators A, E, R, and P, when applied to expressible finitary properties, can be represented by the four future modalities $\Box$, $\Diamond$, $\Box \Diamond$, and $\Diamond \Box$. This is stated by the following claims:

- $Sat(\Box p) \;=\; A(esat(p))$
- $Sat(\Diamond p) \;=\; E(esat(p))$
- $Sat(\Box \Diamond p) \;=\; R(esat(p))$
- $Sat(\Diamond \Box p) \;=\; P(esat(p))$

Consider, for example, the first clause. It is clear that $\sigma \in Sat(\Box p)$ iff $\sigma \models \Box p$ iff all prefixes of σ end-satisfy p iff all prefixes of σ belong to $esat(p)$ iff $\sigma \in A(esat(p))$.

Similar arguments justify the remaining three clauses.

In the following we will be interested in the closure of properties expressible by temporal formulas under the operations of union, intersection, and complementation. The following useful identities show that these operations on the properties can be translated into disjunction, conjunction, and negation of the formulas expressing the properties.

$$\begin{array}{rcl} Sat(\varphi) \ \cup \ Sat(\psi) & = & Sat(\varphi \vee \psi) \\ Sat(\varphi) \ \cap \ Sat(\psi) & = & Sat(\varphi \wedge \psi) \\ \overline{Sat(\varphi)} & = & Sat(\neg\varphi), \end{array}$$

where complementation is relative to Σ^ω.

4.3 Safety Formulas

We define a *canonical safety formula* to be a formula of the form

$$\Box\, p,$$

for a past formula p. Such a formula states that all positions in a computation satisfy p. Equivalently, all proper prefixes of a computation end-satisfy p. A formula equivalent to a canonical safety formula is called a *safety formula.*

A property that can be specified by a (canonical) safety formula is called a *safety-specifiable* property. Clearly, a property specified by the formula $\Box\, p$ is a safety property, because it can be presented as

$$A(esat(p)).$$

Usually, safety formulas express *invariance* of some state property over all computations, or *precedence* constraints of the form: if events e_1 and e_2 happen, then e_1 must precede e_2.

In the simpler cases, p is a state formula, and then the formula $\Box\, p$ specifies that all states in the computation satisfy p. An example of such a simple safety property is the formula

$$\Box(x \geq 0),$$

specifying that, in all states of the computation, the value of x is nonnegative.

We will illustrate the utility of state invariances by presenting several typical examples.

- **Partial Correctness**

Let P be a program whose terminal location is ℓ_t. Let ψ be an assertion specifying the post condition of the program, i.e., constraints on the final state of the program. For example, if P computes the factorial of the input x, and places it in the output variable z, the post condition can be $\psi : (z = x!)$. The program is defined to be partially correct with respect to the post condition ψ if every *terminating* computation terminates in a state satisfying ψ. Nothing is implied about non-terminating computations. Partial correctness with respect to ψ can be specified by the safety formula

$$\Box(at\_\ell_t \rightarrow \psi).$$

This formula states that it is invariantly true that if control is at the terminating location ℓ_t, i.e., the program has terminated, then the post condition ψ holds.

- **Mutual Exclusion**

Consider a program consisting of two processes P_1 and P_2 that need to coordinate their entry to critical sections in their code. The program for each process P_i is usually partitioned into three sections: N_i, T_i, and C_i. The section N_i represents the non-critical activity of the process, where no coordination is required. The section T_i represents the *trying* section, where a process decides it needs to access its critical section and engages in a protocol that will ensure eventual access. The section C_i represents the critical section itself. The basic requirement of mutual exclusion algorithms is that it never be the case that both P_1 and P_2 execute their critical sections at the same time. This requirement can be expressed by the safety formula

$$\square \neg(in\_C_1 \wedge in\_C_2),$$

where in_C_i is a control predicate expressing the fact that P_i is currently executing within the section C_i.

The more general case of safety formulas of the form $\square\, p$, where p is not a state formula, is illustrated by the following examples:

- **Precedence**

The basic precedence formula has the form

$$\square(q \rightarrow \diamondminus\, p).$$

It states that if q ever occurs, then it must be preceded by p. There are many applications (and corresponding interpretations) for this formula. For example, it can be interpreted as the property of *causal dependence* of q upon p. That is, q cannot happen unless it is caused by p. If q is a response to the request p, then this property claims that the system does not respond spuriously, without receiving some prior request.

The same property can also be specified by the future formula $(\neg q)\,\mathcal{W} p$. While this is not a safety formula, it is equivalent to the safety formula given above, and therefore specifies precisely the same property.

- **FIFO ordering**

Assume that q represents a response to a request p, and q' represents a similar response to the request p'. For example, the two may represent responses to different customers. The following safety formula states that the order of the responses matches the order of the requests.

$$\square((q \wedge \diamondminus\, q') \rightarrow \diamondminus(p \wedge \diamondminus\, p'))$$

Note that this formula does not ensure any response at all, but it claims that if responses are generated, they appear in the order of their corresponding requests.

Closure of Safety-Specifiable Properties

The class of properties expressible by safety formulas is closed under the positive boolean operations, i.e., intersection and union. As stated before, it suffices to show that if φ and ψ are safety formulas, then so are $\varphi \wedge \psi$ and $\varphi \vee \psi$. That is, it suffices to show that the class of safety formulas is closed under conjunction and disjunction.

To see this we present the following equivalences for the conjunction and disjunction of safety formulas:

$$\begin{array}{rcl}(\Box\, p \;\wedge\; \Box\, q) & \sim & \Box(p \;\wedge\; q)\\ (\Box\, p \;\vee\; \Box\, q) & \sim & \Box(\boxminus\, p \;\vee \boxminus\, q).\end{array}$$

The left-hand side of the second equivalence states, for a computation σ, that either all positions in σ satisfy p or all positions in σ satisfy q. The right-hand side states that, for *each* position i, either all positions $j \leq i$ satisfy p, or all positions $j \leq i$ satisfy q. To see that the right-hand side implies the left-hand side, we consider two cases. If all positions in σ satisfy both p and q then, clearly, the left-hand side follows. If for some j, $(\sigma, j) \not\models p$, then the only way the right-hand side can hold is by having for all $i \geq j$, $(\sigma, i) \models \boxminus\, q$, from which $\Box\, q$ follows.

Since the right-hand side of both equivalences is a safety formula (under the assumption that p and q are past formulas), this establishes the closure of properties expressible by safety formulas under intersection and union.

An important formula is that of *conditional safety*, in which a property expressed by $\Box\, q$ is conditional on a state formula p holding at the first state of the computation. This formula has the form

$$p \rightarrow \Box\, q.$$

While not being a canonical safety formula itself, this formula is a safety formula due to the equivalence

$$(p \rightarrow \Box\, q) \;\sim\; \Box(\diamondminus(p \wedge \mathit{first}) \rightarrow q).$$

The formula on the right states that, at each position j, if j has been preceded by some position $i \leq j$ that satisfies p and is also *first* (forcing $i = 0$), then q should hold at j.

- **Full Partial Correctness**

In specifying terminating programs, one usually specifies a *precondition* φ, in addition to the *postcondition* ψ. The role of the precondition is to constrain the inputs for which the program is expected to produce the right result. For example, a natural precondition for the factorial computing program is: $\varphi : (x \geq 0)$, claiming that the program is expected to produce a correct result only if we start it with a non-negative input. The partial correctness of a program P with respect to both the precondition φ and the postcondition ψ can be specified by the conditional safety formula

$$\varphi \rightarrow \Box(at\_\ell_t \rightarrow \psi).$$

4.4 Guarantee Formulas

A *canonical guarantee formula* is a formula of the form

$$\Diamond\, p,$$

for some past formula p. Such a formula states that at least one position in a computation satisfies p. A formula equivalent to a canonical guarantee formula is called a *guarantee formula*.

A property that can be specified by a (canonical) guarantee formula is called a *guarantee-specifiable property*. Clearly, any property that can be specified by the guarantee formula $\Diamond\, p$ is a guarantee property, since it can be presented as

$$E(esat(p)).$$

Usually, guarantee formulas ensure that some event *eventually* happens. They guarantee that the event happens at least once, but cannot promise more than finitely many repetitions of the event. Therefore, they are mainly used to ensure events that happen *once* in the lifetime of a program execution, such as termination.

A simple example, in which p is a state formula, is the formula

$$\Diamond(terminal)$$

specifying that some state of the computation is terminal. Clearly, if all computations of a given program satisfy this formula, the program is terminating. Instead of using the abstract predicate *terminal*, we can use the more concrete formula $\Diamond(at\_\ell_t)$, claiming that all computations eventually reach the terminal location ℓ_t.

We observe that the *conditional guarantee* formula

$$p \rightarrow \Diamond\, q,$$

while not being a canonical guarantee formula, is a guarantee formula. This is because it is equivalent to the canonical guarantee formula

$$\Diamond(\diamondminus(first \wedge p) \rightarrow q).$$

This formula states that we eventually reach a position such that, if looking back towards the origin, we detect that p held at the initial position, then q holds now.

- **Total Correctness**

A program P is said to be *totally correct* with respect to a precondition φ and a postcondition ψ if *all* computations starting at a φ-state terminate at a ψ-state. This property can be expressed by the conditional guarantee formula

$$\varphi \rightarrow \Diamond(at\_\ell_t \wedge \psi).$$

Closure of Guarantee-Specifiable Properties

Many features of the guarantee-specifiable class of properties can be obtained by the duality relation between the safety-specifiable and guarantee-specifiable classes. This duality is based on the equivalence

$$\neg \Box\, p \;\sim\; \Diamond(\neg p).$$

From this equivalence we can immediately conclude that Π is a guarantee-specifiable property iff the complementary property $\overline{\Pi}$ (i.e., the set of all computations *not* in Π) is a safety-specifiable property.

In principle we could justify the closure properties of the guarantee-specifiable class using duality and the corresponding closure properties of the safety-specifiable class. However, we prefer to give a direct justification.

Similar to the class of safety formulas, the class of guarantee formulas is closed under the positive boolean operations of disjunction and conjunction.

This can be shown using the following equivalences:

$$
\begin{aligned}
(\Diamond p \;\vee\; \Diamond q) &\;\sim\; \Diamond[p \;\vee\; q] \\
(\Diamond p \;\wedge\; \Diamond q) &\;\sim\; \Diamond[\diamondminus p \;\wedge\; \diamondminus q].
\end{aligned}
$$

The second equivalence claims that a computation σ contains both a p-position and a q-position iff it has a position i such that there exists a q-position $j \leq i$, and a p-position $k \leq i$, both preceding i.

The class of guarantee formulas is not closed under negation. The negation of a guarantee formula is a safety formula. Similarly, the negation of a safety formula is a guarantee formula. We say that the classes of guarantee and safety formulas are *dual*, as each can be obtained by negation from the other.

4.5 Obligation Formulas

Some properties cannot be expressed by either safety or guarantee formulas alone, and must be expressed as a boolean combination of such formulas. We therefore consider the class of such properties.

A *canonical simple obligation formula* is of the form

$$\Box p \;\vee\; \Diamond q,$$

where p and q are past formulas. This formula states that either p holds at all positions of a computation or q holds at some position. A formula equivalent to a canonical simple obligation formula is called a *simple obligation formula.*

A property that can be specified by a simple obligation formula is called a *simple obligation-specifiable property.* Clearly, any property that can be specified by the simple obligation formula $\Box p \vee \Diamond q$ is a simple obligation property, since it can be presented as

$$A(esat(p)) \;\cup\; E(esat(q)).$$

An obviously equivalent form for simple obligation formulas is

$$\Diamond r \;\rightarrow\; \Diamond q,$$

which states that if some position satisfies r, then some position satisfies q. (No temporal ordering between the occurrences of r and q is implied.)

- **Exceptions**

A typical example of properties that are naturally specified by simple obligation formulas is that of exceptional occurrences. Assume that a program in the normal course of its behavior is not supposed to terminate but to maintain some regular activity. However, in the case of the occurrence of some exceptional event p, it is supposed to take some exceptional action q and halt. Specifying this behavior can be done by the simple obligation formula

$$\Diamond p \rightarrow \Diamond(q \wedge \diamondminus p).$$

Note that this formula also requires that q happens only *after* some occurrence of p.

General Obligation Formulas

The class of properties specifiable by simple obligation formulas is closed under union. To see this, we observe the trivial equivalence

$$[(\Box p_1 \vee \Diamond q_1) \;\vee\; (\Box p_2 \vee \Diamond q_2)] \;\sim\; [(\Box p_1 \vee \Box p_2) \;\vee\; (\Diamond q_1 \vee \Diamond q_2)].$$

Using the closure of both the safety and guarantee formula classes under disjunction, this leads to an equivalent simple obligation formula. However, the class of properties specifiable by simple obligation formulas is *not* closed under intersection. This implies that by taking conjunctions of simple obligation formulas we obtain a more powerful class.

We therefore define a *canonical m-obligation formula* to be a formula of the form

$$\bigwedge_{i=1}^{m} [\Box p_i \;\vee\; \Diamond q_i].$$

A formula equivalent to a canonical m-obligation formula is called an *m-obligation formula.*

Correspondingly, a property specifiable by such a formula is called an *m-obligation-specifiable property.*

This class is the largest class that can be obtained by taking finite boolean combinations of safety and guarantee formulas.

Proposition 4.1 *Every boolean combination of safety and guarantee formulas is a simple obligation formula.*

Proof: Consider an arbitrary boolean combination of safety and guarantee formulas. First we push all negations into the past formulas, changing $\wedge$ into $\vee$, $\Box$ into $\Diamond$, and vice-versa. Next we bring the formula into a conjunctive normal form:

$$\bigwedge_{i=1}^{n} [\Box p_1^i \;\vee\; \ldots \;\vee\; \Box p_{k_i}^i \;\vee\; \Diamond q_1^i \;\vee\; \ldots \;\vee\; \Diamond q_{m_i}^i].$$

We then use the closure properties of the safety and guarantee formulas to collapse all of $\Box p_1^i \vee \ldots \vee \Box p_{k_i}^i$ into a single safety formula, and $\Diamond q_1^i \vee \ldots \vee \Diamond q_{m_i}^i$ into a single guarantee formula. ∎

This proposition also implies that the class of obligation formulas is closed under all boolean operations.

Inclusion of the Lower Classes

The class of obligation-specifiable properties strictly contains the class of safety-specifiable properties and the class of guarantee-specifiable properties. To see this, observe that the property described by $\Box p \vee \Diamond q$ for propositions p and q cannot be specified by either safety or guarantee formulas.

The class of obligation-specifiable properties forms an infinite strict hierarchy. The class of properties expressible by a conjunction of $n+1$ simple obligation formulas strictly contains the class of properties expressible by a conjunction of only n simple obligation formulas.

4.6 Response Formulas

A *canonical response formula* is a formula of the form:

$$\Box \Diamond p,$$

for some past formula p. It states that infinitely many positions in the computation satisfy p. A formula equivalent to a canonical response formula is called a *response formula.*

A property that can be specified by a (canonical) response formula is called a *response-specifiable property.*

Clearly, a property specifiable by the response formula $\Box \Diamond p$ is a response property, since it can be presented as

$$R(esat(p)).$$

One of the most important forms of response formulas is the formula

$$\Box(p \rightarrow \Diamond q), \quad \text{equivalently,} \quad p \Rightarrow \Diamond q.$$

The fact that this is a response formula is established by the equivalence

$$\Box(p \rightarrow \Diamond q) \quad \sim \quad \Box \Diamond((\neg p) \mathcal{B} q).$$

The formula on the right states that there exists infinitely many positions in which there is no pending request, i.e., a request that has not been followed by a response.

In general, response formulas ensure that some event happens infinitely many times. Formulas of the form $p \Rightarrow \Diamond q$ express the property of *responsiveness* of a system, stating that every stimulus p has a response q.

- **Accessibility**

Consider again a program for solving the mutual exclusion problem. As already mentioned in the introduction, the safety formula that forbids the two processes from residing together in their critical sections is only part of the specification. It can be easily implemented by a program that does not allow any process to ever access its critical section. To exclude such spurious solutions we must add the requirement that each process interested in entering its critical section will eventually succeed. This requirement can be specified by the response formula

$$\Box(in\_T_i \rightarrow \Diamond in\_C_i).$$

This formula requires that whenever one of the processes is in the trying section T_i, then eventually it will be in the critical section C_i.

Closure of Response-Specifiable Properties

The class of properties specifiable by response formulas is closed under the positive boolean operations.

This is shown by the following equivalences:

$$\begin{array}{lcl} [\Box \Diamond p \ \vee \ \Box \Diamond q] & \sim & \Box \Diamond (p \ \vee \ q) \\ [\Box \Diamond p \ \wedge \ \Box \Diamond q] & \sim & \Box \Diamond \big(q \ \wedge \ \ominus((\neg q)\, \mathcal{S} p)\big). \end{array}$$

The past formula $q \wedge \ominus((\neg q)\,\mathcal{S}\,p)$ expresses precisely the *minex* operator applied to the finitary properties $esat(p)$ and $esat(q)$, that is,

$$esat\big(q \;\wedge\; \ominus((\neg q)\,\mathcal{S}\,p)\big) \;=\; minex(esat(p),\; esat(q)).$$

To see this, let σ be a finite sequence that end-satisfies $q \wedge \ominus((\neg q)\,\mathcal{S}\,p)$. By definition its last position satisfies q, and hence $\sigma \in esat(q)$. The formula $\ominus((\neg q)\,\mathcal{S}\,p)$, holding at the last position of σ, requires that there exists some proper prefix $\sigma' \prec \sigma$, such that σ' end-satisfies p, and hence $\sigma' \in esat(p)$, and for every σ'', $\sigma' \prec \sigma'' \prec \sigma$, σ'' does *not* end-satisfy q, i.e., $\sigma'' \notin esat(q)$. It follows that $\sigma \in minex(esat(p),\; esat(q))$.

Inclusion of the Lower Classes

All safety and guarantee formulas can be shown to be special cases of response formulas. Thus, the class of response-specifiable properties contains the classes of safety-specifiable and guarantee-specifiable properties.

This containment is supported by the following two equivalences:

$$\begin{array}{rcl} \Box\, p & \sim & \Box \Diamond(\boxminus\, p) \\ \Diamond\, p & \sim & \Box \Diamond(\diamondminus\, p). \end{array}$$

The second equivalence, for example, states that a computation σ has a p-position iff there are infinitely many positions in whose past there is a p-position.

The containment of both classes is strict. This is shown by the property $\Box \Diamond p$ for a state formula p, which cannot be expressed by either a safety or a guarantee formula. In fact, it cannot be expressed by any obligation formula.

- **Justice (Weak Fairness)**

One of the important properties belonging to the response-specifiable class is that of *justice* (also called *weak fairness*). The representation of concurrent programs as fair transition systems (see for example [MP83]) associates a justice requirement with some transitions τ in the system. This requirement can be formulated as

> It is not the case that, from a certain point on, the transition τ is continually enabled but never taken.

Thus, any computation satisfying this requirement must have infinitely many positions at which either τ is disabled or τ is taken. This can be expressed by the response formula

$$\Box \Diamond[\neg En(\tau) \;\vee\; taken(\tau)].$$

We assume the existence of formulas $En(\tau)$ and $taken(\tau)$, which hold at a given position in a computation precisely when the transition τ is enabled or taken at this position. The formula $En(\tau)$ is a state formula, while $taken(\tau)$ is a past formula.

4.7 Persistence Formulas

A *canonical persistence formula* is a formula of the form

$$\diamondsuit \square p,$$

for some past formula p. The formula states that all but finitely many positions (all positions from a certain point on) in the computation satisfy p. A formula equivalent to a canonical persistence formula is called a *persistence formula.*

A property that can be specified by a (canonical) persistence formula is called a *persistence-specifiable property.* Clearly, any property that can be specified by the persistence formula $\diamondsuit \square p$ is a persistence property, since it can be presented as

$$P(esat(p)).$$

Usually, persistence formulas are used to describe the eventual stabilization of some state or past property of the system. They allow an arbitrary delay until the stabilization occurs, but require that once it occurs it is continuously maintained.

For example, p may represent a certain stimulus to the system, and it is required that following an occurrence of p, the system will eventually stabilize by continuously maintaining q. This requirement may be specified by the persistence formula

$$\square(p \rightarrow \diamondsuit \square q),$$

which is equivalent to the canonical persistence formula

$$\diamondsuit \square(\diamondminus p \rightarrow q).$$

The latter form requires that all states, from a certain point on, satisfy the requirement that if p has already occurred then q currently holds. Note that this also covers the case that p never occurs, in which case nothing is implied about q.

Closure of Persistence-Specifiable Properties

The classes of properties specifiable by persistence and response formulas are *dual.* This means that the complement of a property in one of the classes belongs to the other. This is supported by the two equivalences:

$$\begin{array}{rcl} \neg(\square \diamondsuit p) & \sim & \diamondsuit \square(\neg p) \\ \neg(\diamondsuit \square p) & \sim & \square \diamondsuit(\neg p). \end{array}$$

This duality can be used for easy transfer of results holding for one class into the other class. For example, all the closure and inclusion properties of the persistence-specifiable class can be derived from the corresponding properties of the response-specifiable class.

For example, we have that the class of properties that can be expressed by persistence formulas is closed under the positive boolean operations. This is shown by the following equivalences, derived by duality from the corresponding equivalences for the response case:

$$\begin{array}{rcl} (\diamondsuit \square p \ \wedge \ \diamondsuit \square q) & \sim & \diamondsuit \square(p \ \wedge \ q) \\ (\diamondsuit \square p \ \vee \ \diamondsuit \square q) & \sim & \diamondsuit \square\big(q \ \vee \ominus(p \, \mathcal{S}(p \ \wedge (\neg q)))\big). \end{array}$$

To see the validity of the second equivalence, we will show first that the left-hand side implies the right-hand side.

Obviously, $\Diamond\Box q$ implies the right-hand side. If $\Diamond\Box p$ is true and $\Diamond\Box q$ is not, let i be the position beyond which p is continuously true, and $j \geq i$ some position at which q is false (by $\Diamond\Box q$ being false there are infinitely many such positions). It is easy to see that for every position $k > j$, $(\sigma, k) \models \ominus(p\,\mathcal{S}(p \wedge (\neg q)))$.

Next, we show that the right-hand side implies the left-hand side. Again, we consider two cases. If $\Diamond\Box q$ holds, then obviously the left-hand side follows. In the other case, there are infinitely many $\neg q$-positions. Let i be the position beyond which the subformula ψ: $q \vee \ominus(p\,\mathcal{S}(p \wedge (\neg q)))$ continuously holds. Consider an arbitrary position $j > i$, and let $k > j$ be the first $\neg q$-position to the strict right of j. Since ψ holds at k and q does not, it follows that $\ominus(p\,\mathcal{S}(p \wedge (\neg q)))$ must hold at k. This means that p must extend from $k-1$ back to, and including, the first $\neg q$-position preceding k. Since there is no $\neg q$-position between j and k, p must also hold at j. Since j is an arbitrarily chosen position greater than i, it follows that $\Box p$ holds at $i+1$, and hence $\Diamond\Box p$ holds at position 0.

Inclusion of the Lower Classes

All safety and guarantee formulas are special cases of persistence formulas. Thus, the class of persistence-specifiable properties contains the classes of properties specifiable by safety and guarantee formulas.

This containment is supported by the following two equivalences:

$$\begin{array}{lcl} \Box p & \sim & \Diamond\Box(\boxminus p) \\ \Diamond p & \sim & \Diamond\Box(\diamondminus p). \end{array}$$

The second equivalence, for example, states that a computation σ has a p-position iff all positions, from a certain point on, have p in their past.

The containment of both classes is strict. This is shown by the property $\Diamond\Box p$ for a state formula p, which cannot be expressed by either a safety or a guarantee formula. In fact, it cannot be expressed by any obligation formula.

4.8 Reactivity Formulas

A *canonical simple reactivity formula* is a formula formed by a disjunction of a response formula and a persistence formula

$$\Box\Diamond p \;\vee\; \Diamond\Box q$$

where p and q are past formulas. This formula states that either the computation contains infinitely many p-positions, or all but finitely many of its positions are q-positions. A formula equivalent to a canonical simple reactivity formula is called a *simple reactivity formula.*

A property that can be specified by a (canonical) simple reactivity formula is called a *simple reactivity-specifiable property.* Clearly, any property that can be specified by the simple reactivity formula $\Box\Diamond p \vee \Diamond\Box q$ is a simple reactivity property, since it can be presented as

$$R(esat(p)) \;\cup\; P(esat(q)).$$

In many cases we specify such properties by a formula of the form

$$\square \diamond r \;\rightarrow\; \square \diamond p,$$

which is obviously a simple reactivity formula.

This formula states that if the computation contains infinitely many r-positions it must also contain infinitely many p-positions. This is a convenient abstraction to a situation in which we want to commit the system to eventually respond, but not specify a bound on how many stimuli r may happen before the eventual response p will be produced.

The Different types of Responsiveness — A Summary

So far, we have encountered several types of responsiveness, which can be specified by formulas belonging to the different classes. Let us review these different versions. Assume that p represents a stimulus, to which the system responds by producing q.

- The guarantee formula

  $$p \rightarrow \diamond q,$$

  ensures that if p is true *initially* then q will eventually happen.

- The (simple) obligation formula

  $$\diamond p \rightarrow \diamond(q \wedge \diamondminus p),$$

  ensures that if p happens at least once, then its earliest occurrence will be followed by at least *one* occurrence of q.

- The response formula

  $$\square(p \rightarrow \diamond q),$$

  ensures that *every* occurrence of p is followed by an occurrence of q.

- The persistence formula

  $$\square(p \rightarrow \diamond \square q),$$

  ensures that an occurrence of p will be eventually followed by a continuous maintenance of q.

- The (simple) reactivity formula

  $$\square \diamond p \rightarrow \square \diamond q,$$

  ensures that infinitely many occurrences of p are responded to by infinitely many occurrences of q.

The type of responsiveness represented by simple reactivity formulas allows the program P to ignore finitely many requests but not infinitely many of them. This description should not be taken too literally, in the sense that no implementation of this requirement can be based on the idea of "let us wait first and see whether there are going to be infinitely many p events or only finitely many of them." Any reasonable implementation of such a requirement must sincerely attempt to respond to all requests, but the liberal specification tolerates failures to respond in the case of finitely many requests.

The class of properties specifiable by simple reactivity formulas is closed under union. This is due to the equivalences

$$[(\Box\Diamond p_1 \;\vee\; \Diamond\Box q_1) \;\vee\; (\Box\Diamond p_2 \;\vee\; \Diamond\Box q_2)] \sim$$
$$[(\Box\Diamond p_1 \;\vee\; \Box\Diamond p_2) \;\vee\; (\Diamond\Box q_1 \;\vee\; \Diamond\Box q_2)]$$

and the closure of the response and persistence-specifiable classes under union.

However, the simple reactivity-specifiable class is, in general, not closed under intersection or complementation.

Obviously, the class of properties specifiable by simple reactivity formulas contains the classes of properties specifiable by response and persistence formulas, and hence also the classes specifiable by safety, guarantee and obligation formulas. This containment is strict since the property specifiable by $\Box\Diamond p \vee \Diamond\Box q$ cannot be expressed by any formula belonging to a lower class.

- **Compassion (Strong Fairness)**

Simple reactivity formulas can express the requirement of *compassion* (also called *strong fairness*) associated with special transitions of a fair transition system. Typically, we associate compassion requirements with transitions that correspond to communication or synchronization statements in the program, such as statements dealing with semaphores.

The compassion requirement associated with a transition τ demands that

> It is not the case that τ is enabled infinitely many times but taken only finitely many times.

Equivalently, this requirement demands that if the transition τ is enabled infinitely many times in a computation σ, then it must be taken infinitely many times. This can be expressed by the formula

$$\Box\Diamond En(\tau) \;\rightarrow\; \Box\Diamond\, taken(\tau).$$

General Reactivity Properties

Richer classes of properties can be expressed by conjunctions of simple reactivity formulas of the form

$$\bigwedge_{i=1}^{m}\Big[\Box\Diamond p_i \;\vee\; \Diamond\Box q_i\Big].$$

Since, in general, the conjunction of two simple reactivity formulas is not a simple reactivity formula, taking such conjunctions leads to stronger expressive power.

We refer to such a formula as a *canonical m-reactivity formula*, and to the property it specifies as an *m-reactivity-specifiable property*. A formula equivalent to a canonical m-reactivity formula is called an *m-reactivity formula.*

The class of reactivity-specifiable properties is the maximal class we ever need to consider. This is due to the following normal form theorem:

Theorem 4.2 *Every temporal formula is a reactivity formula.*

The proof of this proposition is based on a translation between future and past temporal formulas. A detailed proof of this theorem is beyond the scope of this paper. The interested reader may consult [MP90a] for some details of such a proof.

A natural example of a property specifiable by a reactivity formula is the total statement of fairness for a fair transition system. Since each individual fairness requirement is expressible by a simple reactivity formula (response formula if it is a justice requirement), the statement that *all* fairness requirements hold is expressible as the conjunction of several simple reactivity formulas.

Our approach to specification of programs is inherently conjunctive. This means that a specification is presented as a conjunction of requirements, expressed by temporal formulas, all of which should be valid over the program. In verifying that a specification is valid over a given program, we can verify the validity of each requirement separately. Therefore, the fact that one of the requirements is a conjunction by itself, rather than a simple reactivity formula, does not greatly complicate or simplify the situation. Hence, in the context of a full specification, which is always a conjunction, we may assume each requirement to be at most a simple reactivity formula.

The family of properties specifiable by reactivity formulas forms an infinite hierarchy by itself. Level k of the hierarchy, for $k > 0$, consists of all the properties that can be specified by a conjunction

$$\bigwedge_{i=1}^{n}[\Box \Diamond p_i \ \vee \ \Diamond \Box q_i]$$

for some $n \leq k$. This hierarchy is strict, since the conjunction

$$\bigwedge_{i=1}^{n+1}[\Box \Diamond p_i \ \vee \ \Diamond \Box q_i]$$

with p_i, q_i, $i = 1, \ldots, n+1$, being uninterpreted propositions, is not equivalent to any conjunction of n or fewer simple reactivity formulas.

4.9 Relating the Syntactic and Semantic Classifications

As we will see, the syntactic hierarchy of properties, based on their expression by particular form of formulas, is consistent with the language-theoretic (semantic) hierarchy, based on the construction of infinitary properties by applying the operators A, E, R, and P to finitary properties. When introducing the class corresponding to each type of formula κ, where

$$\kappa \in \left\{ \begin{array}{lll} \textit{safety}, & \textit{guarantee}, & m\textit{-obligation}, \\ \textit{response}, & \textit{persistence}, & m\textit{-reactivity} \end{array} \right\}$$

we immediately showed that any κ-specifiable property is a κ property according to the semantic classification. For example, as soon as we defined the notion of safety formulas, we showed that any property specifiable by a safety formula is a safety property according to the semantic classification.

We are now ready to consider the other direction. Suppose Π is an infinitary property that is known to be specifiable by a temporal formula and is *also* known to be a κ-property. Can we conclude that it is a κ-specifiable property? We have already answered this question positively for the case $\kappa = \textit{reactivity}$. This is due to Theorem 4.2, which stated that any temporal formula is equivalent to a canonical reactivity formula. The next theorem answers this question positively for the other classes as well.

Theorem 4.3 *A property* Π *that is specifiable in temporal logic is a* κ*-property iff it is specifiable by a canonical* κ*-formula, where* $\kappa \in$ { *safety, guarantee, m-obligation, response, persistence, m-reactivity* }.

We defer the proof of this theorem to subsection 5.3.

4.10 Standard Formulas

The temporal characterization presented above 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 implied in the subsequent section on automata may lead to an exponential blowup.

As a typical example, consider a system in which the event p should trigger the two responses r_1 and r_2, but r_1 should always weakly precede (i.e., precede or coincide with) r_2. The property stating that, from every occurrence of p, the next occurrence of r_2 must be weakly preceded by an occurrence of r_1, can be expressed by the temporal formula

$$p \Rightarrow (\neg r_2)\, \mathcal{W} r_1$$

which uses the *unless* (*waiting-for*) operator $\mathcal{W}$. Note that this formula does not state that r_2 will actually occur, following p. It only states that if r_2 occurs, it must be weakly preceded by r_1. Consequently, it is a *safety* formula. To use the temporal characterization by canonical formulas, we must first transform it to canonical form. Indeed, the formula above is equivalent to

$$\Box(r_2 \;\rightarrow\; (\neg p)\, \mathcal{B} r_1)$$

which uses the *back-to* (also known as the weak *since*) operator $\mathcal{B}$. This formula states that, going back from every occurrence of r_2, we must encounter an r_1 before we encounter a p. This form identifies the considered property as a *safety* property, since it it is of the form $\Box\, q$ for some past formula q. While the two formulas are equivalent, specifiers often prefer the use of *future* formulas and may find the need to transform to past-oriented specifications awkward and unnatural.

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

This new characterization of properties is based on the definition of *standard formulas* of type κ.

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

$$p \vee q \qquad p \wedge q \qquad \bigcirc p \qquad \Box p \qquad p \mathcal{W} q$$

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

$$p \vee q \qquad p \wedge q \qquad \bigcirc p \qquad \Diamond p \qquad p \mathcal{U} q$$

c. standard *obligation* formulas

– Every standard safety and standard guarantee formula is a standard obligation formula.

– If p and q are standard obligation formulas, then so are

$$\neg p \qquad p \vee q \qquad p \wedge q \qquad \bigcirc p$$

– If p is a standard obligation formula and q is a standard guarantee formula, then the following is a standard obligation formula.

$$p \mathcal{U} q$$

– If p is a standard safety formula and q is a standard obligation formula, then the following is a standard obligation formula.

$$p \mathcal{W} q$$

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

$$p \vee q \qquad p \wedge q \qquad \bigcirc p \qquad \Box p \qquad p \mathcal{W} q$$

– If p is a standard response formula and q is a standard guarantee formula, then the following is a standard response formula.

$$p \mathcal{U} q$$

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

$$p \vee q \qquad p \wedge q \qquad \bigcirc p \qquad \Diamond p \qquad p\,\mathcal{U}\,q$$

– If p is a standard safety formula and q is a standard persistence formula, then the following is a standard persistence formula.

$$p\,\mathcal{W}\,q$$

f. If p and q are standard κ-formulas, where κ is one of safety, guarantee, obligation, response, or persistence, then the following are standard κ-formulas.

$$\ominus p \qquad \diamondminus p \qquad \boxminus p \qquad p\,\mathcal{S}\,q \qquad p\,\mathcal{B}\,q$$

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

In the temporal hierarchy, every canonical obligation formula is both a canonical response formula and a canonical persistence formula; although this is not stated explicitly in the above definition, it is clear that every standard obligation formula is both a standard response formula and a standard persistence formula.

Separated Properties

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

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

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

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

Theorem 4.4 *Standard safety, guarantee, response, and persistence formulas may be characterized as follows.*

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

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

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

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

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

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

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

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

Note that the converse to Theorem 4.4 does not hold; in particular, $p\mathcal{W}(\neg p)$ is not a standard guarantee formula, even though it can be defined by an upwards-closed separated property.

Corollary 4.5 *Every standard κ-formula, where κ is one of safety, guarantee, response, or persistence, specifies a κ-property.*

The finitary properties Φ induced by the sets Ω described in Theorem 4.4 clearly establish this corollary.

Corollary 4.6 *Every standard obligation formula specifies an obligation property.*

Note that Corollaries 4.5 and 4.6 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.

If we consider future standard κ-formulas, i.e., standard κ-formulas with no past operators, we encounter the following question:

> **Open Problem:** *Is every κ-specifiable property specifiable by a future standard κ-formula?*

4.11 The Temporal Characterization of Liveness

Similar to our previous efforts to give a syntactic characterization to the *safety-progress* hierarchy, we provide a syntactic characterization to the *safety-liveness* partition hierarchy. Since the *safety* class belongs to both the *safety-liveness* partition and to the *safety-progress* hierarchy, it possesses a canonical temporal characterization as all properties that can be specified by a formula of the form $\Box\, p$ for some past formula p. This characterization was presented in [LPZ85]. The work in [Sis85] proposes other temporal characterizations of the safety-liveness classification. The characterization proposed for safety properties agrees with our future standard safety formulas.

In this paper we improve on these results by presenting a complete temporal characterization of the *liveness* class, and a simpler proof of the completeness of the temporal characterization of the safety class. These results are based on a separation theorem due to Gabbay [Gab87].

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

- A canonical safety formula is any formula of the form $\square\, p$, 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 terminology.

Theorem 4.7 (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.

Using the separation theorem, we can now prove the main characterization result.

Theorem 4.8 *A property specifiable by temporal logic is a λ-property iff it is specifiable by a canonical λ-formula, where $\lambda \in \{$ safety, 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 $esat(p)$, 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 $\neg \ominus true$ 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, $\bigvee_{i=1}^{k} p_i$ hold at all positions of σ. It follows that $\chi: \ \Box(\bigvee_{i=1}^{k} p_i)$ holds at position 1 of σ.

Next, consider a word σ that satisfies $\chi: \ \Box(\bigvee_{i=1}^{k} p_i)$. 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 mixed 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 holds at all positions, including position j of σ. This means that $(\sigma, j) \models p_i$ 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 extended 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 $\Diamond \bigvee_{i=1}^{k}(p_i \wedge \bigcirc f_i)$ (at position 1). 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. As in the previous case, we construct the formula $\psi: \ \diamondminus(\mathit{first} \wedge \varphi)$ 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. It follows that $\Box(\bigvee_{i=1}^{k} p_i)$ is valid. ▪

5 The Automata View

An alternative formalism for specifying temporal properties is that of finite-state automata (see [AS89], [MP87]). A *Streett automaton* $\mathcal{M}$ consists of the following components:

- Q — A finite set of automaton-states.

- $q_0 \in Q$ — An initial automaton-state.

- $\delta : Q \times \Sigma \mapsto Q$ — A *transition function* specifying, for each $q \in Q$ and $s \in \Sigma$, the next state $\delta(q, s)$ to which the automaton moves when reading s while in automaton-state q. If there is no $s \in \Sigma$ such that $\delta(q, s) = q'$, we say that there is no transition from q to q'.

- $\mathcal{L} : \big((R_1, P_1), \ldots, (R_m, P_m)\big)$ — A list of *accepting pairs*. Each pair consists of $R_i \subseteq Q$, the i-th set of *recurrent* states, and $P_i \subseteq Q$, the i-th set of *persistent* states.

We refer to an automaton with m accepting pairs as an m-automaton. For a 1-automaton, which is also called a *plain automaton*, we refer to R_1 and P_1 as R and P, respectively. Streett automata are shown to have expressive power exceeding temporal logic in [Wol83], and have been studied in [Str82].

Let

$$\sigma\colon\ s_1, s_2, s_3, \ldots \in \Sigma^\omega$$

be a computation. Computations are presented as input to an automaton, which either accepts them or rejects them. An infinite sequence of automaton-states

$$r\colon\ q_1, q_2, q_3, \ldots \in Q^\omega$$

is called a *run* of $\mathcal{M}$ over σ if:

1. The first state of the run, q_1, is the initial state of $\mathcal{M}$.

2. For every $i > 0$, $q_{i+1} = \delta(q_i, s_i)$.

It follows from the definition of the transition function that an automaton has exactly one run over any computation. Alternative definitions are considered in [Str82] and [MP87].

We define the *infinite visitation set* of σ, $\mathit{vinf}(\sigma)$, to be the set of automaton-states that occur infinitely often in the run of $\mathcal{M}$ over σ. An m-automaton *accepts* a computation σ if, for every $i = 1, \ldots, m$, either $\mathit{vinf}(\sigma) \cap R_i \neq \emptyset$ or $\mathit{vinf}(\sigma) \subseteq P_i$.

5.1 The Hierarchy of Automata

We define the following classes of automata by introducing restrictions on their transition conditions and accepting pairs.

- A *safety automaton* is a plain automaton such that $R = \emptyset$ and there is no transition from $q \notin P$ to $q' \in P$. It follows that a run of a safety automaton is accepting if and only if all the states appearing in the run are in P.

- A *guarantee automaton* is a plain automaton such that $P = \emptyset$ and there is no transition from $q \in R$ to $q' \notin R$. It follows that a run of a guarantee automaton is accepting if and only if only finitely many of the states appearing in the run are not in R.

- An *m-obligation automaton* is an m-automaton such that, for each $i = 1, \ldots, m$:

  - there is no transition from $q \notin P_i$ to $q' \in P_i$.

– there is no transition from $q \in R_i$ to $q' \notin R_i$.

This definition implies that once a run of an obligation automaton exits P_i, it can never reenter P_i again, and once it enters R_i, it can never get out. We refer to a 1-obligation automaton as a *simple obligation automaton.*

- A *response automaton* is a plain automaton such that $P = \emptyset$.
- A *persistence automaton* is a plain automaton such that $R = \emptyset$.
- An *m-reactivity automaton* is any unrestricted m-automaton. We refer to a 1-reactivity automaton as a *simple reactivity automaton.*

We define the (infinitary) property specified by an automaton $\mathcal{M}$, $\Pi_{\mathcal{M}}$, as the set of all computations that are accepted by $\mathcal{M}$.

An infinitary property $\Pi \subseteq \Sigma^\omega$ is said to be *specifiable by automata* if there exists an automaton $\mathcal{M}$ such that $\Pi = \Pi_{\mathcal{M}}$. The following theorem relates the syntactic characterization of the different types of automata to the semantic characterization of the properties they specify.

Theorem 5.1 *A property* Π *that is specifiable by automata is a* κ*-property iff it is specifiable by a* κ*-automaton, where* $\kappa \in$ *{ safety, guarantee, obligation, response, persistence, reactivity }.*

For the basic classes, this theorem has been proved in [Lan69], with some minor differences in the definitions of safety and guarantee automata. The case of reactivity, and in fact the complete hierarchy above, has been solved in [Wag79].

For completeness, we include our proof of the theorem, which for most of the cases is straightforward.

Proof: It is easy to show that a κ-automaton specifies a κ-property. Let $\mathcal{M}$ be an m-automaton. Define the *extended transition function* $\widehat{\delta}(\sigma)$ *of* $\mathcal{M}$ to be the automaton-state that is reached after reading σ, where $\sigma \in \Sigma^+$.

Define $\Phi(q) = \{\sigma \in \Sigma^+ \mid \widehat{\delta}(\sigma) = q\}$ for each $q \in Q$.

By definition, a computation σ is accepted by an m-automaton $\mathcal{M}$ if and only if, for each $i = 1, \ldots, m$, its corresponding run r either visits states in R_i infinitely many times, or is constrained from a certain point on to visit only states in P_i. It follows that

$$\Pi_{\mathcal{M}} = \bigcap_{1 \le i \le m} \left(R(\bigcup_{q \in R_i} \Phi(q)) \cup P(\bigcup_{q \in P_i} \Phi(q)) \right)$$

Consequently, every property specifiable by an m-automaton is an m-reactivity property. However, for the special cases of κ-automata, the above expression can be further simplified.

▪ For a safety automaton, we are given that $m = 1$ and $R = \emptyset$, so $\Pi_{\mathcal{M}} = P(\bigcup_{q \in P} \Phi(q))$. Since there are no transitions from a state $q \notin P$ to a state $q' \in P$, it follows that

$$\Pi_{\mathcal{M}} = A(\bigcup_{q \in P} \Phi(q)),$$

which establishes that $\Pi_{\mathcal{M}}$ is a safety property.

- For a guarantee automaton, we are given that $m = 1$ and $P = \emptyset$, so $\Pi_{\mathcal{M}} = R(\bigcup_{q \in R} \Phi(q))$. Since there are no transitions from a state $q \in R$ to a state $q' \notin R$, it follows that

$$\Pi_{\mathcal{M}} = E(\bigcup_{q \in R} \Phi(q)),$$

which establishes that $\Pi_{\mathcal{M}}$ is a guarantee property.

- For an m-obligation automaton, we are given that there are neither transitions from a state $q \notin P_i$ to a state $q' \in P_i$, nor transitions from a state $q \in R_i$ to a state $q' \notin R_i$. It follows that

$$\Pi_{\mathcal{M}} = \bigcap_{1 \le i \le m} \left(A(\bigcup_{q \in P_i} \Phi(q)) \cup E(\bigcup_{q \in R_i} \Phi(q)) \right)$$

which establishes that $\Pi_{\mathcal{M}}$ is an m-obligation property.

- For a response automaton, we are given that $m = 1$ and $P = \emptyset$, and therefore $\Pi_{\mathcal{M}} = R(\bigcup_{q \in R} \Phi(q))$.
- For a persistence automaton, we are given that $m = 1$ and $R = \emptyset$, and therefore $\Pi_{\mathcal{M}} = P(\bigcup_{q \in P} \Phi(q))$.

Consider now the other direction of the theorem. It states that a κ-property specifiable by automata can be specified by a κ-automaton. Now, assume that a κ-property Π is specifiable by automata. Then there exists an automaton

$$\mathcal{M} = \langle Q, q_0, \delta, \mathcal{L} \rangle, \quad \mathcal{L} = \{(R_i, P_i),\ i = 1, \ldots, k\}$$

specifying Π. Without loss of generality, we assume that each state in $\mathcal{M}$ is reachable from the initial state q_0.

Let $\widehat{\delta}(\sigma)$ be the extended transition function of $\mathcal{M}$.

- First consider the case that Π is a guarantee property, in which case $\Pi = E(\Phi)$ for some finitary property Φ. Without loss of generality, we may assume that Φ is *extension closed*, i.e., $\Phi \cdot \Sigma^* \subseteq \Phi$. This is justified by the observation that $E(\Phi \cdot \Sigma^*) = E(\Phi)$.

We construct the guarantee automaton:

$$\mathcal{M}' = \langle Q, q_0, \delta, \{(R', \emptyset)\} \rangle,$$

where Q, q_0, and δ are as before. R' is defined by

$$R' = \{q \in Q \mid q = \widehat{\delta}(\sigma) \text{ for some } \sigma \in \Phi\}.$$

To show that $\mathcal{M}'$ is a guarantee automaton, we will show that if $q \in R'$, then $\delta(q, s) \in R'$ for every $s \in \Sigma$. Let $\sigma \in \Phi$ be such that $q = \widehat{\delta}(\sigma)$. Since Φ is extension closed, it follows that $\sigma \cdot s \in \Phi$, and therefore $\delta(q, s) = \widehat{\delta}(\sigma \cdot s) \in R'$.

Next, we show that $\sigma \in \Pi$ if and only if $\sigma \in \Pi_{\mathcal{M}'}$.

Assume that $\sigma \in \Pi$. Let σ' be a prefix of σ such that $\sigma' \in \Phi$. By the definition of R', $\widehat{\delta}(\sigma') \in R'$. Thus, the run of $\mathcal{M}'$ over σ visits R', and since $\mathcal{M}'$ is a guarantee automaton, it must stay in R' forever. It follows that $\mathcal{M}'$ accepts σ.

In the other direction, assume that $\sigma \in \Pi_{\mathcal{M}'}$. Let σ' be a prefix of σ such that $q = \widehat{\delta}(\sigma') \in R'$. By the definition of R', there is some $\widehat{\sigma} \in \Phi$ such that $\widehat{\delta}(\widehat{\sigma}) = \widehat{\delta}(\sigma') = q$. Let σ'' be such that $\sigma = \sigma' \cdot \sigma''$. The word $\widehat{\sigma} \cdot \sigma''$ contains a prefix belonging to Φ; it is therefore in Π and must be accepted by $\mathcal{M}$. Since $\widehat{\delta}(\sigma') = \widehat{\delta}(\widehat{\sigma})$, $vinf(\sigma' \cdot \sigma'') = vinf(\widehat{\sigma} \cdot \sigma'')$. Therefore $\mathcal{M}$ must also accept $\sigma' \cdot \sigma''$, so $\sigma \in \Pi$.

▪ Now consider the case that Π is a safety property. It follows that $\overline{\Pi} = \Sigma^\omega - \Pi$ is a guarantee property specifiable by automata, so by the previous construction there is a guarantee automaton

$$\mathcal{M}' = \langle Q,\ q_0,\ \delta,\ \{(R', \emptyset)\}\rangle$$

such that $\Pi_{\mathcal{M}'} = \overline{\Pi}$.

Then

$$\mathcal{M}'' = \langle Q,\ q_0,\ \delta,\ \{(\emptyset, Q - R')\}\rangle$$

is a safety automaton such that $\Pi_{\mathcal{M}''} = \Pi$.

▪ Consider the case that Π is an m-obligation property, and can therefore be represented as

$$\bigcap_{i=1}^{m} (A(\Phi_i) \cup E(\Psi_i)).$$

Without loss of generality, we assume that, for each $i = 1, \ldots, m$, Φ_i is prefix closed and Ψ_i is extension closed. That is, $\sigma \in \Phi_i$ and $\sigma' \prec \sigma$ implies $\sigma' \in \Phi_i$, and $\sigma \in \Psi_i$ and $\sigma \prec \sigma'$ implies $\sigma' \in \Psi_i$. The prefix closure of Φ_i implies that $\overline{\Phi}_i$, its complement, is extension closed.

A *cycle* in $\mathcal{M}$ is any closed path. A cycle $\mathcal{C}$ is called a *good cycle* if a word whose infinite visitation set is equal to (the states occurring in) $\mathcal{C}$ is accepted by $\mathcal{M}$. It is called a *bad cycle* if a word whose infinite visitation set is $\mathcal{C}$ is rejected by $\mathcal{M}$.

A state q is called a *good state* if it is a member of some good cycle. It is called a *bad state* if it is a member of some bad cycle.

Lemma 5.2 *No state can be both good and bad.*

Proof: Assume, to the contrary, that state q is a member of both the good cycle $\mathcal{C}_{\mathrm{G}}$ and the bad cycle $\mathcal{C}_{\mathrm{B}}$. Let σ_0 be a finite word causing the automaton to reach q, and σ_{G}, σ_{B} words that cause the automaton to move from q to q along $\mathcal{C}_{\mathrm{G}}$ and $\mathcal{C}_{\mathrm{B}}$, respectively.

Consider the infinite word $\sigma_1 : \sigma_0 \cdot {\sigma_{\mathrm{B}}}^\omega$. Since this word has the infinite visitation set $\mathcal{C}_{\mathrm{B}}$, it is rejected by $\mathcal{M}$. This implies that σ is not in $\bigcap_{i=1}^{m}(A(\Phi_i) \cup E(\Psi_i))$. It follows that there exists some $i_1 \in [1..m]$ such that σ_1 has a prefix in $\overline{\Phi}_{i_1}$. Consequently, there exists an n_1 such that $\sigma_0 \cdot {\sigma_{\mathrm{B}}}^{n_1} \in \overline{\Phi}_{i_1}$.

Next, consider the infinite word $\widehat{\sigma}_1 : \sigma_0 \cdot {\sigma_{\mathrm{B}}}^{n_1} \cdot {\sigma_{\mathrm{G}}}^\omega$. This word has the infinite visitation set $\mathcal{C}_{\mathrm{G}}$, and is therefore accepted. However, since it contains a prefix in $\overline{\Phi}_{i_1}$ it can belong to Π only if it also contains a prefix in Ψ_{i_1}. There exists therefore an $\widehat{n}_1$ such that $\sigma_0 \cdot {\sigma_{\mathrm{B}}}^{n_1} \cdot {\sigma_{\mathrm{G}}}^{\widehat{n}_1} \in \Psi_{i_1}$.

Now consider the infinite word $\sigma_2 : \sigma_0 \cdot {\sigma_{\mathrm{B}}}^{n_1} \cdot {\sigma_{\mathrm{G}}}^{\widehat{n}_1} \cdot {\sigma_{\mathrm{B}}}^\omega$. As before, this word is rejected, and there must exist some i_2 such that $\sigma_2 \notin (A(\Phi_{i_2}) \cup E(\Psi_{i_2}))$. Since σ_2 contains a prefix in Ψ_{i_1}, i_2 must be distinct from i_1. Therefore there exists an n_2 such that $\sigma_0 \cdot {\sigma_{\mathrm{B}}}^{n_1} \cdot {\sigma_{\mathrm{G}}}^{\widehat{n}_1} \cdot {\sigma_{\mathrm{B}}}^{n_2} \in \overline{\Phi}_{i_2}$. By considering $\sigma_0 \cdot {\sigma_{\mathrm{B}}}^{n_1} \cdot {\sigma_{\mathrm{G}}}^{\widehat{n}_1} \cdot {\sigma_{\mathrm{B}}}^{n_2} \cdot {\sigma_{\mathrm{G}}}^\omega$, we identify an $\widehat{n}_2$ such that $\sigma_0 \cdot {\sigma_{\mathrm{B}}}^{n_1} \cdot {\sigma_{\mathrm{G}}}^{\widehat{n}_1} \cdot {\sigma_{\mathrm{B}}}^{n_2} \cdot {\sigma_{\mathrm{G}}}^{\widehat{n}_2} \in \Psi_{i_1} \cap \Psi_{i_2}$.

Continuing in this way, we finally construct a word $\sigma_0 \cdot {\sigma_{\mathrm{B}}}^{n_1} \cdot {\sigma_{\mathrm{G}}}^{\widehat{n}_1} \cdot {\sigma_{\mathrm{B}}}^{n_2} \cdot {\sigma_{\mathrm{G}}}^{\widehat{n}_2} \cdots {\sigma_{\mathrm{B}}}^{n_m} \cdot {\sigma_{\mathrm{G}}}^{\widehat{n}_m}$ that belongs to $\Psi_{i_1} \cap \cdots \cap \Psi_{i_m}$, where $i_1, \ldots, i_m$ are pairwise distinct. Consider now the infinite word

$$\sigma_0 \cdot {\sigma_{\mathrm{B}}}^{n_1} \cdot {\sigma_{\mathrm{G}}}^{\widehat{n}_1} \cdot {\sigma_{\mathrm{B}}}^{n_2} \cdot {\sigma_{\mathrm{G}}}^{\widehat{n}_2} \cdots {\sigma_{\mathrm{B}}}^{n_m} \cdot {\sigma_{\mathrm{G}}}^{\widehat{n}_m} \cdot {\sigma_{\mathrm{B}}}^{\omega}.$$

By arguments similar to the preceding ones, this word should be rejected. On the other hand, it contains prefixes in $\Psi_1, \ldots, \Psi_m$, and is therefore accepted.

We conclude that no state can belong to both a good and a bad cycle. ∎

It follows that no cycle can contain both a good and a bad state.

Now, decompose $\mathcal{M}$ into maximal strongly connected components $K_1, \ldots, K_\ell$. These components can be arranged in a sequence so that, if there is an edge from (a state in) K_i to (a state in) K_j, then $i \leq j$. If there is an edge from K_i to K_j, $i \neq j$, we call K_i a *predecessor* of K_j. Note that $q_0 \in K_1$.

A component K is called *non-trivial* if there exists a cycle traversing all the states in K. Trivial components consist of a single state q with no transition from q to itself.

By the observation following Proposition 5.2, each non-trivial component consists either of all good states or all bad states. States that form trivial components are called *neutral*, and they are neither good nor bad. We call a component good, bad, or neutral according to the classification of its states.

Let q be a state. For $n \geq 1$, we say that q has *alternation* n if there exists a path

$$q_0, \ldots, b_1, \ldots, g_1, \ldots, b_{n-1}, \ldots, g_{n-1}, \ldots, b_n, \ldots, q,$$

connecting q_0 to q, where b_i and g_i are good and bad states, respectively. In this presentation, b_1 can coincide with q_0 and b_n can coincide with q.

By definition, every q has alternation 0. The following lemma shows that the possible alternations of states are bounded by m, the number of sets in the intersection $\bigcap_{i=1}^{m}(A(\Phi_i) \cup E(\Psi_i))$.

Lemma 5.3 *No state can have alternation bigger than m.*

Proof: Assume, to the contrary, that some state q has an alternation

$$q_0, \ldots, b_1, \ldots, g_1, \ldots, b_{n-1}, \ldots, g_{n-1}, \ldots, b_n, \ldots, q,$$

with $n > m$. By a construction similar to that of Proposition 5.2, we construct a word that causes the automaton to progress from q_0 to b_1, loop several times around the bad cycle to which b_1 belongs, progress to g_1, loop several times around the good cycle to which g_1 belongs, and so on, until by the time it exits g_m, the word belongs to each of $\Psi_1, \ldots, \Psi_m$. From this point on, there should be no path to a bad state, and therefore $m \geq n$. ∎

We define the *rank* of a state q, denoted by $\rho(q)$, to be the maximal alternation that q has. By the above, the rank of every state is bounded by m. We observe the following properties of the ranks of states:

- If q is connected to q' then $\rho(q) \leq \rho(q')$.
- If there is a path $q, n_1, \ldots, n_p, q'$, where q is good, q' is bad, and $n_1, \ldots, n_p$, $p \geq 0$, are neutral, then $\rho(q) < \rho(q')$.
- All states in a strongly connected component have the same rank.

We will now show how to construct an obligation automaton with m accepting pairs that accepts Π.

For $i = 0, 1, \ldots, m$, let Q_i denote the set of all states whose rank is i. We denote by $Q_{<i}$ the set of all states whose rank is smaller than i.

Let G_i denote the set of all good states whose rank is i.

For $i = 1, \ldots, m$, we define the sets:

R_i : The set of all states reachable from a G_j-state for some $j \geq i$,

P_i : $Q_{<i}$.

Let $\mathcal{M}'$ denote the automaton whose states, initial state, and transition functions are identical to those of M, and whose accepting list is

$$(R_1, P_1), \ldots, (R_m, P_m).$$

The automaton $\mathcal{M}'$ is an obligation automaton. A state in $\overline{P}_i$ has rank not smaller than i. By the first property of the ranking function ρ, such a state cannot be connected to a state in P_i whose rank is smaller than i.

Clearly, a state reachable from an R_i-state is reachable from a G_j-state, for some $j \geq i$, and is therefore also in R_i.

Lemma 5.4 *The automata $\mathcal{M}$ and $\mathcal{M}'$ are equivalent.*

Proof: Assume that σ is accepted by $\mathcal{M}$. Let I be the infinity set of $\mathcal{M}$ over σ. Since σ is accepted, I is contained in one of the good components K_t. We will show that, for each $i = 1, \ldots, m$, I is either contained in P_i or has a nonempty intersection with R_i. Assume that all states in I have the rank r. Let g be one of the states in I. Since the rank of g is r, there exists a path connecting q_0 to g, whose alternation is r. As we go along this path, we encounter good states $g_1, \ldots, g_r$, whose ranks are $1, \ldots, r$, respectively. It follows that g (and hence all of I) belongs to each of $R_1, \ldots, R_r$. Since their rank is r, they also belong to $P_{r+1}, \ldots, P_m$.

In the other direction, assume that σ is accepted by $\mathcal{M}'$ and let I be the infinity set of $\mathcal{M}'$ over σ. As we observed earlier, the states in I must be either all good or all bad. We will show that they are all good.

Consider first the case $r = 0$. In this case, I cannot contain any bad states, otherwise the rank of its states would have been at least 1.

Next consider an $r > 0$. The set I must either be contained in P_r or have a nonempty intersection with R_r. Since P_r only allows states with ranks below r, I must intersect R_r. Let $q \in I \cap R_r$. This means that there is a path from some good state g_j to q, where g_j has a rank $j \geq r$. Since the rank of q is r, j must equal r. If q was bad, its rank would therefore have been at least $r + 1$. It follows that the states in I must all be good. ▟

- Next, consider the case that Π is a response property. This means that $\Pi = R(\Phi)$ for some finitary Φ.

A cycle A is called an *R_i-free good cycle*, for $i = 1, \ldots, m$, if A is a good cycle that has no intersection with R_i. A state q is called an *R_i-free good state* if it belongs to some R_i-free good cycle. Let F_i denote the set of all R_i-free good states.

Let $\mathcal{M}'$ be the automaton coinciding with $\mathcal{M}$ in all but $\mathcal{L}$, the list of accepting pairs, which is given by $(R'_1, \emptyset), \ldots, (R'_m, \emptyset)$, where for each $i = 1, \ldots, m$:

$$R'_i = R_i \cup F_i$$

We will show that $\mathcal{M}$ and $\mathcal{M}'$ accept precisely the same language.

Consider first a word σ accepted by $\mathcal{M}$. Let J be the infinite visitation set $vinf(\sigma)$. Consider any $i = 1, \ldots, m$. Since $\mathcal{M}$ accepts σ, either $J \cap R_i \neq \emptyset$ or $J \subseteq P_i \cap \overline{R_i}$. In the first case $J \cap R'_i \neq \emptyset$. In the second case, J is an R_i-free good cycle. It follows that $J \subseteq F_i$, hence $J \cap R'_i \neq \emptyset$.

Consider next a word σ accepted by $\mathcal{M}'$. Let J be its infinite visitation set. Suppose σ is rejected by $\mathcal{M}$. This implies that, for some $i = 1, \ldots, m$, $J \cap R_i = \emptyset$ and there exists some $q_{\mathrm{B}} \in J \cap \overline{P_i}$. Since σ is accepted by $\mathcal{M}'$, there also exists some $q_{\mathrm{G}} \in J \cap F_i$. Let G be the R_i-free good set to which q_{G} belongs. Let σ_0 be a finite word such that $\widehat{\delta}(\sigma_0) = q_{\mathrm{G}}$. Let π_1 be a cyclic path from q_{G} to itself, traversing J, and let σ_1 be a finite word leading the automaton from q_{G} to q_{G} along π_1. Let π_2 be a cyclic path from q_{G} to itself, exactly traversing G, and let σ_2 be a finite word leading the automaton from q_{G} to q_{G} along π_2.

Observe that, for every word σ' such that $\widehat{\delta}(\sigma') = q_{\mathrm{G}}$, there exists a positive integer n such that $\sigma' \cdot (\sigma_2)^n$ contains a prefix $\sigma'' \prec \sigma' \cdot (\sigma_2)^n$, $|\sigma''| > |\sigma'|$, which belongs to Φ. To see this, observe that the computation $\sigma' \cdot (\sigma_2)^\omega$ has G as its infinite visitation set, and is therefore in Π. Consequently, $\sigma' \cdot (\sigma_2)^\omega$ must have infinitely many prefixes in Φ, infinitely many of which are longer than σ'. The shortest of these is a prefix of $\sigma' \cdot (\sigma_2)^n$ for an appropriate $n > 0$.

Now consider the following word:

$$\widehat{\sigma} \;=\; \sigma_0 \sigma_1 (\sigma_2)^{n_1} \sigma_1 (\sigma_2)^{n_2} \cdots$$

where the n_j's are chosen so that $\widehat{\sigma}$ has infinitely many prefixes in Φ. That is, for each

$$\widehat{\sigma}_{j-1} \;=\; \sigma_0 \sigma_1 (\sigma_2)^{n_1} \cdots \sigma_1 (\sigma_2)^{n_{j-1}} \sigma_1$$

we choose an $n_j > 0$ such that $\widehat{\sigma}_{j-1} \cdot (\sigma_2)^{n_j}$ has a prefix in Φ longer than $\widehat{\sigma}_{j-1}$.

It follows, on one hand, that $\widehat{\sigma}$ has infinitely many prefixes in Φ, so $\widehat{\sigma} \in \Pi$.

On the other hand, the infinite visitation set of $\widehat{\sigma}$ is $J \cup G$, which has an empty intersection with R_i and at least one state $q_{\mathrm{B}} \notin P_i$. Therefore $\widehat{\sigma} \notin \Pi_{\mathcal{M}}$, contradicting the hypothesis that $\mathcal{M}$ accepts Π. Consequently, there cannot exist a word σ which is accepted by $\mathcal{M}'$ but rejected by $\mathcal{M}$. It follows that $\mathcal{M}'$ is equivalent to $\mathcal{M}$.

It remains to show that such an automaton is equivalent to an automaton with a single R and a single $P = \emptyset$. This is essentially a closure property stating that the intersection of response automata is equivalent to a single response automaton. The construction is similar in spirit to the formula

$$[\Box \Diamond p \;\wedge\; \Box \Diamond q\,] \quad \sim \quad \Box \Diamond (q \;\wedge\; \ominus((\neg q)\, \mathcal{S} p))$$

showing the closure under conjunction of response formulas. For $m = 2$, construct an automaton which "detects" visits to R_2-states only when the most recent previous visit to an $(R_1 \cup R_2)$-state was a visit to an R_1-state.

▪ Now consider the case that Π is a persistence property. It follows that $\overline{\Pi} = \Sigma^\omega - \Pi$ is a response property specifiable by automata, so by the previous construction there is a response automaton

$$\widetilde{\mathcal{M}} = \langle Q,\ q_0,\ \widetilde{\delta},\ \{(\widetilde{R}, \emptyset)\}\rangle$$

such that $\Pi_{\widetilde{\mathcal{M}}} = \overline{\Pi}$.

Then

$$\widehat{\mathcal{M}} = \langle Q,\ q_0,\ \widetilde{\delta},\ \{(\emptyset, Q - \widetilde{R})\}\rangle$$

is a persistence automaton such that $\Pi_{\widehat{\mathcal{M}}} = \Pi$.

▪ Finally, let Π be an m-reactivity property which is accepted by the automaton $\mathcal{M}$. Being an m-reactivity property, Π can be represented as

$$\bigcap_{i=1}^{m}(R(\Phi_i) \cup P(\Psi_i)).$$

An *alternating chain* of length $k > 0$ is a sequence of cycles

$$B_1 \subset G_1 \subset B_2 \subset \cdots \subset G_{k-1} \subset B_k,$$

where $B_1, \ldots, B_k$ are bad cycles and $G_1, \ldots, G_{k-1}$ are good cycles.

A cycle X is defined to be *of height* 0 if there is no bad cycle $B \subseteq X$. It is defined to be *of height* $k > 0$ if the maximal length of an alternating chain contained in X (i.e., $B_k \subseteq X$), is k.

Lemma 5.5 *If* Π *is an* m*-reactivity property recognized by an automaton* $\mathcal{M}$*, then* $\mathcal{M}$ *has no alternating chains longer than* m*.*

Proof: Assume to the contrary, that $\mathcal{M}$ has an alternating chain

$$B_1 \subset G_1 \subset B_2 \subset \cdots \subset G_m \subset B_{m+1}.$$

By induction on $n = 1, \ldots, m+1$, we will prove the following two claims:

- Claim 1 — For every G_n, $n = 1, \ldots, m$, and every $\sigma_0 \in \Sigma^+$, such that $\widehat{\delta}(\sigma_0) = q \in G_n$, there exists a word $\sigma \in \Sigma^+$ such that
  1. The word σ causes $\mathcal{M}$ to move from q back to q along a path that is fully contained in G_n.
  2. There are n pairwise distinct indices $i_1, i_2, \ldots, i_n$, and n prefixes $\sigma_1, \ldots, \sigma_n$ of $\sigma_0 \cdot \sigma$, each longer than σ_0, such that $\sigma_j \in \Phi_{i_j}$ for $j = 1, \ldots, n$.
- Claim 2 — For every B_n, $n = 1, \ldots, m+1$, and every $\sigma_0 \in \Sigma^+$, such that $\widehat{\delta}(\sigma_0) = q \in B_n$, there exists a word $\sigma \in \Sigma^+$ such that
  1. The word σ causes $\mathcal{M}$ to move from q back to q along a path that is fully contained in B_n.
  2. There are n pairwise distinct indices $i_1, i_2, \ldots, i_n$, and n prefixes $\sigma_1, \ldots, \sigma_n$ of $\sigma_0 \cdot \sigma$, each longer than σ_0, such that $\sigma_j \in \Phi_{i_j}$ for $j = 1, \ldots, n-1$ and $\sigma_n \in \overline{\Psi}_{i_n}$.

Base case — B_1

Consider σ_0 such that $\widehat{\delta}(\sigma_0) = q \in B_1$. Let σ_1 be a word that causes $\mathcal{M}$ to move from q back to q while traversing all of B_1. The infinite word $\sigma_0 \cdot \sigma_1^\omega$ has B_1 as its infinite visitation set and is therefore rejected by $\mathcal{M}$. It follows that there exists some $i_1 \in [1..m]$ such that infinitely many prefixes of $\sigma_0 \cdot \sigma_1^\omega$ are in $\overline{\Psi}_{i_1}$. Take k to be the smallest positive integer such that $\sigma_0 \cdot \sigma_1^k$ contains at least one such prefix longer than σ_0. Then $\sigma = \sigma_1^k$ is the word promised by Claim 2 for B_1.

Induction step — B_n to G_n

Consider σ_0 such that $\widehat{\delta}(\sigma_0) = q \in G_n$. Without loss of generality, we can assume that $q \in B_n \subset G_n$. This is because, if q is not already in B_n, we can extend σ_0 a little to lead to some $q' \in B_n$ without going out of G_n.

By the induction hypothesis for B_n and σ_0, there exists a σ_1 leading from q to itself while remaining within B_n (and hence within G_n), such that $\sigma_0 \cdot \sigma_1$ contains prefixes $\tau_1, \ldots, \tau_n$, all longer than σ_0, such that $\tau_1, \ldots, \tau_{n-1}$ belong, respectively, to $\Phi_{i_1}, \ldots, \Phi_{i_{n-1}}$ and $\tau_n \in \overline{\Psi}_{i_n}$, for some pairwise distinct $i_1, \ldots, i_n$. We say that σ_1 has the signature $s_1 : \langle i_1, \ldots, i_{n-1}, i_n \rangle$, where we order $i_1 < i_2 < \cdots < i_{n-1}$.

Let $\widehat{\sigma}$ be a word that moves $\mathcal{M}$ from q back to itself while traversing all of G_n.

Next, apply the induction hypothesis to B_n and $\sigma_0 \cdot \sigma_1 \cdot \widehat{\sigma}$. This yields a word σ_2 with some signature s_2.

We repeat this construction to obtain an infinite word

$$\sigma_{inf} : \sigma_0 \cdot \sigma_1 \cdot \widehat{\sigma} \cdot \sigma_2 \cdot \widehat{\sigma} \cdot \sigma_3 \cdots \ .$$

Since σ_{inf} contains infinitely many instances of $\widehat{\sigma}$, each causing $\mathcal{M}$ to traverse all of G_n, this word is accepted. As there are only finitely many different possible signatures, some signature $s : \langle i_1, \ldots, i_n \rangle$ must appear infinitely many times. This means that σ_{inf} contains infinitely many prefixes belonging to Φ_{i_j} for each $j = 1, \ldots, n-1$, and also infinitely many prefixes belonging to $\overline{\Psi}_{i_n}$. The latter implies that σ_{inf} is not in $P(\Psi_{i_n})$. Since σ_{inf} is accepted it must belong to $R(\Phi_{i_n})$. Consequently σ_{inf} also contains infinitely many prefixes belonging to Φ_{i_n}. Let k be the smallest integer such that

$$\sigma_0 \cdot \sigma_1 \cdot \widehat{\sigma} \cdot \sigma_2 \cdot \widehat{\sigma} \cdot \sigma_3 \cdots \sigma_k$$

contains prefixes belonging to $\Phi_1, \ldots, \Phi_n$, all longer than σ_0. Then $\sigma_1 \cdot \widehat{\sigma} \cdot \sigma_2 \cdot \widehat{\sigma} \cdot \sigma_3 \cdots \sigma_k$ is the word promised by Claim 1 for G_n.

Induction step — G_n to B_{n+1}

Consider σ_0 such that $\widehat{\delta}(\sigma_0) = q \in B_{n+1}$. Without loss of generality, we can assume that $q \in G_n \subset B_{n+1}$.

By the induction hypothesis for G_n and σ_0, there exists a σ_1 leading from q to itself while remaining within G_n (and hence within B_{n+1}), such that $\sigma_0 \cdot \sigma_1$ contains prefixes $\tau_1, \ldots, \tau_n$, all longer than σ_0, such that $\tau_1, \ldots, \tau_n$ belong, respectively, to $\Phi_{i_1}, \ldots, \Phi_{i_n}$, for some pairwise distinct $i_1, \ldots, i_n$. As before, we say that σ_1 has the signature $s_1 : \langle i_1, \ldots, i_n \rangle$, where we order $i_1 < i_2 < \cdots < i_n$.

Let $\widehat{\sigma}$ be a word that moves $\mathcal{M}$ from q back to itself while traversing all of B_{n+1}.

Next, apply the induction hypothesis to G_n and $\sigma_0 \cdot \sigma_1 \cdot \widehat{\sigma}$. This yields a word σ_2 with some signature s_2.

We repeat this construction to obtain an infinite word

$$\sigma_{inf} : \sigma_0 \cdot \sigma_1 \cdot \widehat{\sigma} \cdot \sigma_2 \cdot \widehat{\sigma} \cdot \sigma_3 \cdots \ .$$

Since σ_{inf} contains infinitely many instances of $\widehat{\sigma}$, each causing $\mathcal{M}$ to traverse all of B_{n+1}, this word is rejected. As there are only finitely many different possible signatures, some signature $s : \langle i_1, \ldots, i_n \rangle$ must appear infinitely many times. This means that σ_{inf} contains infinitely many prefixes belonging to Φ_{i_j} for each $j = 1, \ldots, n$. Since σ_{inf} is rejected there must exist some i such that infinitely many prefixes of σ_{inf} belong to $\overline{\Psi}_i$

and all but finitely many prefixes of σ_{inf} belong to $\overline{\Phi}_i$. For each i_j, $j = 1, \ldots, n$, σ_{inf} contains infinitely many prefixes belonging to Φ_{i_j}. Therefore i must be different from all of $i_1, \ldots, i_n$. We have thus identified a sequence of pairwise distinct indices $i_1, \ldots, i_n, i$ such that, for each i_j, $j = 1, \ldots, n$, σ_{inf} contains infinitely many prefixes belonging to Φ_{i_j}, and σ_{inf} also contains infinitely many prefixes belonging to $\overline{\Psi}_i$.

Let k be the smallest integer such that

$$\sigma_0 \cdot \sigma_1 \cdot \widehat{\sigma} \cdot \sigma_2 \cdot \widehat{\sigma} \cdot \sigma_3 \cdots \sigma_k$$

contains prefixes belonging to $\Phi_1, \ldots, \Phi_n$, and a prefix belonging to $\overline{\Psi}_i$, all longer than σ_0. Then $\sigma_1 \cdot \widehat{\sigma} \cdot \sigma_2 \cdot \widehat{\sigma} \cdot \sigma_3 \cdots \sigma_k$ is the word promised by Claim 2 for B_{n+1}.

Returning now to the main proof, an alternating chain of length $m+1$ would imply the existence of $m+1$ pairwise distinct indices $i_1, \ldots, i_m, i_{m+1}$, each ranging over $1, \ldots, m$. This, of course, is impossible, and therefore no such alternating chain can exist. ▪

We observe that if the property Π_m is specifiable by the m-automaton $\mathcal{M}_m$ and the property Π_n is specifiable by the n-automaton $\mathcal{M}_n$, then the property $\Pi_m \cap \Pi_n$ is specifiable by an $(m+n)$-automaton $\mathcal{M}_{m+n}$. The automaton $\mathcal{M}_{m+n}$ can be obtained by forming the cross product of $\mathcal{M}_m$ and $\mathcal{M}_n$.

Thus, instead of constructing an m-automaton that specifies a property Π, it is sufficient to construct a set of m plain (simple reactivity) automata $\mathcal{M}_1, \ldots, \mathcal{M}_n$ specifying, respectively, the properties $\Pi_1, \ldots, \Pi_m$, such that $\Pi = \Pi_1 \cap \cdots \cap \Pi_m$.

In the following lemma, we consider an automaton $\mathcal{M}$ and two families of sets of states

$$\mathcal{L} : \{L_1, \ldots, L_p\} \quad \text{and} \quad \mathcal{V} : \{V_1, \ldots, V_r\}.$$

The containment mentioned in the formulation of the lemma is a weak one, i.e., for every X, X contains itself.

Lemma 5.6 *Given an automaton $\mathcal{M}$ and two families of sets $\mathcal{L}$ and $\mathcal{V}$, there exists a simple reactivity automaton $\mathcal{M}_{\mathcal{L},\mathcal{V}}$ such that $\mathcal{M}_{\mathcal{L},\mathcal{V}}$ accepts a word σ iff the infinite visitation set of $\mathcal{M}$ over σ either contains some set $L \in \mathcal{L}$ or is contained in some set $V \in \mathcal{V}$.*

Proof: Construct the following automaton:

$$\mathcal{M}' = \langle Q',\ q'_0,\ \delta',\ \{R', P'\} \rangle$$

$Q' = Q \times Q^p \times 2 \times r \times 2$.

Each state $q' \in Q'$ has the following structure:

$$q' = \langle q, q_1, \ldots, q_p, f_R, j, f_P \rangle,$$

where $q \in Q$, $q_i \in L_i$, $i = 1, \ldots, p$, $f_R, f_P \in \{0, 1\}$ and $1 \le j \le r$.

We assume that the states of $\mathcal{M}$ are ordered in some linear order. For each L_i, $i = 1, \ldots, p$, we define $min(L_i)$ to be the state of L_i appearing first in the linear order. For $q \in L_i$ we define $next(q, L_i)$ to be the first state $\hat{q} \in L_i$ appearing after q in the linear order. If $q \in L_i$ is the last L_i-state in the linear order, then $next(q, L_i) = min(L_i)$.

The roles of the different components in q' is as follows:

- The state q simulates the behavior of the original automaton. Each $q_i \in L_i$ anticipates the next L_i-state we expect to meet. If the run visits all the L_i's infinitely many times, each anticipated q_i will be matched infinitely many times.
- The response flag f_R is set to 1 each time one of the anticipated L_i-states is matched.
- The index j checks whether the run of $\mathcal{M}$ stays completely within one of the sets $V_1, ..., V_r$ from a certain point on. It moves cyclically over $1, ..., r$, and at any point checks whether the next automaton-state is in V_j. If the next automaton-state is in V_j, then j retains its value and the next value of f_P will be 1. Otherwise, j is incremented (modulo r), and the next value of f_P is 0.

$$q_0' = \langle q_0, min(L_1), \ldots, min(L_p), 0, 1, 0\rangle$$

- δ' is defined as follows:

$$\delta'(\langle q, q_1, \ldots, q_p, f_R, j, f_p\rangle, s) = \langle \tilde{q}, \tilde{q}_1, \ldots, \tilde{q}_p, \tilde{f}_R, \tilde{j}, \tilde{f}_P\rangle$$
$$\textit{iff}$$
$$\begin{array}{ll} & \tilde{q} = \delta(q, s) \\ \wedge & \bigwedge_{i=1}^{p} \left(\begin{array}{c} (\tilde{q} = q_i) \wedge (\tilde{q}_i = next(q_i, L_i)) \\ \vee \\ (\tilde{q} \neq q_i) \wedge (\tilde{q}_i = q_i) \end{array} \right) \\ \wedge & (\tilde{f}_R = 1) \leftrightarrow \bigvee_{i=1}^{p} (\tilde{q} = q_i) \\ \wedge & \left(\begin{array}{c} (\tilde{q} \in V_j) \wedge (\tilde{j} = j) \\ \vee \\ (\tilde{q} \notin V_j) \wedge (\tilde{j} = [j \ mod\ r] + 1) \end{array} \right) \\ \wedge & (\tilde{f}_P = 1) \leftrightarrow (\tilde{q} \in V_j). \end{array}$$

  - The first clause in this definition states that the first component q follows the same path that would be followed by the original automaton.
  - The second clause states, for each $i = 1, \ldots, p$, that either the newly visited automaton-state $\tilde{q}$ matches the anticipated state of L_i, in which case we modify q_i to the next L_i-state in sequence, or that there is no match and q_i remains the same.
  - The third clause states that f_R is set to one iff $\tilde{q}$ matches one of the anticipated states. If different from 1 it must be 0.
  - The fourth clause states that if $\tilde{q}$ belongs to V_j then j is preserved. Otherwise it is incremented by one in a cyclic manner.
  - The last clause states that f_P is set to 1 whenever $\tilde{q}$ is in V_j.

- The acceptance sets are defined by

$$\begin{array}{rcl} R' & = & \{\langle q, q_1, \ldots, q_p, f_R, j, f_P\rangle \in Q' \mid f_R = 1\} \\ P' & = & \{\langle q, q_1, \ldots, q_p, f_R, j, f_P\rangle \in Q' \mid f_P = 1\}. \end{array}$$

Let σ be a computation and r' the corresponding run of $\mathcal{M}'$ over σ. If r' visits R' infinitely many times, this implies that r, the run of $\mathcal{M}$ over σ, visits infinitely many times *all* the states of some L_i. This shows that $vinf(\sigma) \supseteq L_i$.

If r' stays contained in P' from a certain point on, it means that the value of j is never changed beyond that point and hence r is contained in V_j from that point on.

A similar argument establishes that $\sigma \in \Pi_{\mathcal{M}'}$ if $vinf(\sigma)$ contains some set $L \in \mathcal{L}$ or is contained in some set $V \in \mathcal{V}$. also accepted by $\mathcal{M}'$. ∎

For $k = 0, \ldots, m$, let $\mathcal{H}_k$ denote the family of all good sets of height k. We also denote

$$\begin{aligned} \mathcal{H}_{\geq k} &: \bigcup_{i \geq k} \mathcal{H}_i \\ \mathcal{H}_{<k} &: \bigcup_{i<k} \mathcal{H}_i \end{aligned}$$

Thus, $\mathcal{H}_{\geq k}$ is the family of all good sets whose height is not lower than k, while $\mathcal{H}_{<k}$ is the family of all good sets whose height is smaller than k.

For each $k = 1, \ldots, m$, we define a simple reactivity automaton $\mathcal{M}_k$, which is $\mathcal{M}_{\mathcal{L}_k, \mathcal{V}_k}$ as constructed in Proposition 5.6, taking

$$\mathcal{L}_k : \mathcal{H}_{\geq k} \qquad \text{and} \qquad \mathcal{V}_k : \mathcal{H}_{<k}.$$

Interpreting the meaning of these automata we see that a word σ is accepted by $\mathcal{M}_k$ iff the infinite visitation set of $\mathcal{M}$ over σ either contains some good set of height not lower than k or is contained in some good set of height smaller than k.

Lemma 5.7 *An infinite word σ is accepted by $\mathcal{M}$ iff it is accepted by $\mathcal{M}_1, \ldots, \mathcal{M}_m$.*

Proof: Assume that σ is accepted by $\mathcal{M}$ and let X be the infinite visitation set of $\mathcal{M}$ over σ. We will show that σ is accepted by each of $\mathcal{M}_1, \ldots, \mathcal{M}_m$. Obviously, X is a good set and has some height h, $0 \leq h \leq m$.

For each $k \leq h$, σ is accepted by $\mathcal{M}_k$ since X contains X which is a member of $\mathcal{H}_{\geq k}$. For each $k > h$, σ is accepted by $\mathcal{M}_k$ since X is contained in X which is a member of $\mathcal{H}_{<k}$.

In the other direction, assume that σ is accepted by every simple reactivity automaton $\mathcal{M}_1, \ldots, \mathcal{M}_m$, and let X be the infinite visitation set of $\mathcal{M}$ over σ. We will show that X is good.

Let the height of X be h. Note that X can only contain sets of height not exceeding h and can only be contained in sets of height not lower than h.

If $h = 0$ then X cannot contain any bad set, in particular it cannot be bad itself. Thus, for the case $h = 0$, X is good.

Consider an $h > 0$. Since σ is accepted by $\mathcal{M}_h$, X either contains some set in $\mathcal{H}_{\geq h}$ or is contained in some set in $\mathcal{H}_{<h}$. Since the latter is impossible, there exists some good set G_1 of height h which is contained in X. If $h = m$ then X must be good, otherwise it participates in a chain of length $m + 1$. If $h < m$ then σ is also accepted by $\mathcal{M}_{h+1}$ which implies that either X contains some good set of height not smaller than $h + 1$, or it is contained in some good set of height smaller than $h + 1$. The only possibility is that X is contained in some good set G_2 of height h.

It follows that X is bracketed between G_1 and G_2, i.e.,

$$G_1 \subseteq X \subseteq G_2,$$

where both G_1 and G_2 are good sets of height h. The set X cannot be bad because, if it were, then G_2 would have contained an alternating chain of length $h+1$. ▟

This concludes the proof of Theorem 5.1. ▟

Now it is easy to see that the hierarchies of m-obligation and m-reactivity properties are strict, and that in particular, the languages O_m and $\mathcal{E}_m$ defined in subsection 2.5 are m'-obligation and m'-reactivity properties respectively, only for $m' \geq m$.

We will show that any automaton accepting O_m must contain a state which has alternation at least m. Consider the computation $\widehat{\sigma}_1^B : \langle 0, 1^{m-1}\rangle^\omega$. Since $\widehat{\sigma}_1^B \notin O_m$, $vinf(\widehat{\sigma}_1^B)$ must be a bad cycle, and consequently there is some $\sigma_1^B \prec \widehat{\sigma}_1^B$ for which $\widehat{\delta}(\sigma_1^B)$ is bad. Now, consider the computation $\widehat{\sigma}_1^G : \sigma_1^B \cdot \langle 2, 1^{m-1}\rangle^\omega$ which is in O_m. There is some σ_1^G, $\sigma_1^B \prec \sigma_1^G \prec \widehat{\sigma}_1^G$, for which $\widehat{\delta}(\sigma_1^G)$ is good, and, since $\widehat{\delta}(\sigma_1^G)$ is reachable from $\widehat{\delta}(\sigma_1^B)$, it has alternation of size at least 1. Then $\widehat{\sigma}_2^B : \sigma_1^G \cdot \langle 1, 0, 1^{m-2}\rangle^\omega$ must be bad, so there is some σ_2^B, $\sigma_1^G \prec \sigma_2^B \prec \widehat{\sigma}_1^B$, for which $\widehat{\delta}(\sigma_2^B)$ is bad. But $\widehat{\sigma}_2^G : \sigma_2^B \cdot \langle 1, 2, 1^{m-2}\rangle^\omega \in O_m$, and therefore there is some $\sigma_2^G \prec \widehat{\sigma}_2^G$ for which $\widehat{\delta}(\sigma_2^G)$ is good, with alternation 2. Clearly we can continue this construction until we obtain $\widehat{\sigma}_m^G$, which is in O_m and must visit a good state with alternation at least m.

To show that $\mathcal{E}_m$ is (at least) an m-reactivity property is a more difficult task. An infinite word $\alpha \in \Sigma_m^\omega$ is called *beyond-n persistent* if for every $i > n$, $\pi_i(\alpha) \in [0..2]^* 1^\omega$.

Proposition 5.8 *The hierarchy of m-reactivity properties is strict.*

Proof: It is clear that $\mathcal{E}_m$ is specifiable by automata. We will show that $\mathcal{E}_m$ is an m'-reactivity property for no $m' < m$ by showing that every automaton specifying $\mathcal{E}_m$ has alternating chains of length m. By Lemma 5.5, this implies that $m' \geq m$.

Let $\mathcal{M}$ be an automaton specifying $\mathcal{E}_m$. We will show that $\mathcal{M}$ has alternating chains of length m. This is shown by proving the following two claims by induction on $n = 1, \ldots, m$:

- Claim $C_1[n]$ — For every finite word α, there exists a beyond-n persistent infinite word β such that $vinf(\alpha \cdot \beta)$ is a bad cycle of height at least n.

- Claim $C_2[n]$ — For every finite word α, there exists a beyond-n persistent infinite word γ such that $vinf(\alpha \cdot \gamma)$ is a good cycle of height at least n.

Base case

Claim $C_1[1]$ is shown by taking β to be $\langle 0, 1^{m-1}\rangle^\omega$. Since $\pi_1(\alpha \cdot \beta) \in [0..2]^* 0^\omega$, $\alpha \cdot \beta$ does not belong to $\mathcal{E}_m$ and should be rejected by $\mathcal{M}$. Consequently, $vinf(\alpha \cdot \beta)$ is a bad cycle which is therefore of height at least 1. It is also obvious that β is beyond-1 persistent.

Induction step — $C_1[n]$ to $C_2[n]$

Consider a given finite word α. Apply claim $C_1[n]$ to α to obtain an infinite beyond-n persistent word β_1, such that $B_1 = vinf(\alpha \cdot \beta_1)$ is a bad cycle of height at least n. Let $\widehat{\beta}_1$ be a finite prefix of β_1 that causes $\mathcal{M}$ to move from $\widehat{\delta}(\alpha)$ along a path that visits each state of B_1 at least once.

Apply claim $C_1[n]$ to $\alpha \cdot \widehat{\beta}_1 \cdot \langle 2^n, 1^{m-n}\rangle$. This produces a beyond-n persistent infinite word β_2, such that $B_2 = vinf(\alpha \cdot \widehat{\beta}_1 \cdot \langle 2^n, 1^{m-n}) \cdot \beta_2\rangle$ is a bad cycle of height at least n. As before, let $\widehat{\beta}_2$ be a finite prefix of β_2 that causes $\mathcal{M}$ to move from $\widehat{\delta}(\alpha \cdot \widehat{\beta}_1 \cdot \langle 2^n, 1^{m-n}\rangle)$ along a path that visits each state of B_2 at least once.

In this way we construct an infinite beyond-n persistent word

$$\sigma \quad : \quad \alpha \cdot \widehat{\beta}_1 \cdot \langle 2^n, 1^{m-n} \rangle \cdot \widehat{\beta}_2 \cdot \langle 2^n, 1^{m-n} \rangle \cdots$$

Since for every $i \leq n$, $\pi_i(\sigma) \in ([0..2]^*2)^\omega$ and, for every $i > n$, $\pi_i(\sigma) \in [0..2]^*1^\omega$, the word σ belongs to $\mathcal{E}_m$. On the other hand, the segment of the run of $\mathcal{M}$ over σ corresponding to each $\widehat{\beta}_j$ visits each state of the bad cycle B_j at least once. Since there are only finitely many different bad cycles, there exists some B_k that is contained in $\mathit{vinf}(\sigma)$. Since each B_j is of height at least n, it follows that $\mathit{vinf}(\sigma)$ is a good cycle of height at least n. Consequently we take the word γ promised by claim $C_2[n]$ to be

$$\widehat{\beta}_1 \cdot \langle 2^n, 1^{m-n} \rangle \cdot \widehat{\beta}_2 \cdot \langle 2^n, 1^{m-n} \rangle \cdots.$$

Induction step — $C_2[n]$ to $C_1[n+1]$, for $n < m$

Consider a given finite word α. Apply claim $C_2[n]$ to α to obtain an infinite beyond-n persistent word γ_1, such that $G_1 = \mathit{vinf}(\alpha \cdot \gamma_1)$ is a good cycle of height at least n. Let $\widehat{\gamma}_1$ be a finite prefix of γ_1 that causes $\mathcal{M}$ to move from $\widehat{\delta}(\alpha)$ along a path that visits each state of G_1 at least once.

Apply claim $C_2[n]$ to $\alpha \cdot \widehat{\gamma}_1 \cdot \langle 0^{n+1}, 1^{m-n-1} \rangle$. This produces a beyond-n persistent infinite word γ_2, such that $G_2 = \mathit{vinf}(\alpha \cdot \widehat{\gamma}_1 \cdot \langle 0^{n+1}, 1^{m-n-1}) \cdot \gamma_2 \rangle$ is a good cycle of height at least n. As before, let $\widehat{\gamma}_2$ be a finite prefix of γ_2 that causes $\mathcal{M}$ to move from $\widehat{\delta}(\alpha \cdot \widehat{\gamma}_1 \cdot \langle 0^{n+1}, 1^{m-n-1} \rangle)$ along a path that visits each state of G_2 at least once.

In this way we construct an infinite beyond-$(n+1)$ persistent word

$$\sigma \quad : \quad \alpha \cdot \widehat{\gamma}_1 \cdot \langle 0^{n+1}, 1^{m-n-1} \rangle \cdot \widehat{\gamma}_2 \cdot \langle 0^{n+1}, 1^{m-n-1} \rangle \cdots$$

Since $\pi_{n+1}(\sigma) \in [0..2]^*(2^*0)^\omega$, the word σ does not belong to $\mathcal{E}_m$. On the other hand, the segment of the run of $\mathcal{M}$ over σ corresponding to each $\widehat{\gamma}_j$ visits each state of the good cycle G_j at least once. Since there are only finitely many different good cycles, there exists some G_k that is contained in $\mathit{vinf}(\sigma)$. Since each G_j is of height at least n, it follows that $\mathit{vinf}(\sigma)$ is a bad cycle of height at least $n+1$. Consequently we take the word β promised by claim $C_1[n+1]$ to be

$$\widehat{\gamma}_1 \cdot \langle 0^{n+1}, 1^{m-n-1} \rangle \cdot \widehat{\gamma}_2 \cdot \langle 0^{n+1}, 1^{m-n-1} \rangle \cdots. \quad \blacksquare$$

5.2 Deciding the Type of a Property

In this section we consider the following problem:

> **Problem:** Given a Streett automaton $\mathcal{M}$, decide whether the property specified by this automaton is a κ-property, where $\kappa \in \{$ *safety*, *guarantee*, *m-obligation response*, *persistence*, *m-reactivity* $\}$.

The following proposition gives an answer to this general question:

Proposition 5.9 *It is decidable whether a given Streett automaton specifies a κ-property, where $\kappa \in \{$ safety, guarantee, m-obligation, response, persistence, m-reactivity $\}$.*

Again, for the basic classes, the answer has been given in [Lan69]. For the case of reactivity, as well as the complete hierarchy below, it is provided by Wagner in [Wag79].

In the context of specification, this question was considered in [AS87], where a decision procedure was given for safety and liveness.

Since the decision procedures we are interested in are relatively simple, we present them below.

A set of automaton-states $A \subset Q$ is defined to be *closed* if for every $q \in A$ and $s \in \Sigma$, $\delta(q, s) \in A$. The *closure* $\widehat{A}$ of a set of automaton-states is the smallest closed set containing A.

Let $\mathcal{M}$ be an m-automaton. We follow the proof of Theorem 5.1 and define a state to be *good* if it is a member of a good (accepting) cycle, and *bad* if it is a member of a bad (rejecting) cycle. Let G and B denote the sets of good and bad states, respectively.

- Checking for a *safety* property.
  $\mathcal{M}$ specifies a safety property iff $\widehat{B} \cap G = \emptyset$.

- Checking for a *guarantee* property.
  $\mathcal{M}$ specifies a guarantee property iff $\widehat{G} \cap B = \emptyset$.

- Checking for an m-obligation property.
  $\mathcal{M}$ specifies an m-obligation property iff the length of every alternation

  $$q_0, \ldots, b_1, \ldots, g_1, \ldots, g_{n-1}, \ldots, b_n, \ldots, q$$

  as defined in the proof of Theorem 5.1, does not exceed m, i.e., $n \leq m$.

The following are direct consequences of the proof of Theorem 5.1.

- Checking for a *response* property.
  $\mathcal{M}$ specifies a response property iff for every good cycle G and every cycle $A \supseteq G$, A is a good cycle.

- Checking for a *persistence* property.
  $\mathcal{M}$ specifies a persistence property iff for every good cycle G and every cycle $A \subseteq G$, A is a good cycle.

- Checking for an *m-reactivity* property.
  $\mathcal{M}$ specifies an m-reactivity property iff the length n of every alternating chain

  $$B_1 \subset G_1 \subset B_2 \subset G_2 \subset \cdots \subset G_n,$$

  does not exceed m.

5.3 Connections Between Temporal Logic and Automata

Temporal logic and Streett automata have been considered as alternatives for specifying properties of programs. A comparison of their expressive power is considered next.

Theorem 5.10 *A property that is specifiable by a canonical κ-formula is specifiable by a κ-automaton, $\kappa \in \{$ safety, guarantee, m-obligation, response, persistence, m-reactivity $\}$.*

This is based on the following construction, studied in [LPZ85] and [Zuc86].

For every finite set of past formulas $p_1, \ldots, p_k$ it is possible to construct an automaton $\mathcal{M}$ with states Q and designated subsets $F_1, \ldots, F_k \subseteq Q$. The automaton $\mathcal{M}$ has the property that for each $i = 1, \ldots, k$, each computation σ, and each position $j \geq 0$:

$$\widehat{\delta}(\sigma[0 \ldots j]) \in F_i \quad \textit{iff} \quad (\sigma, j) \models p_i$$

Thus the automaton identifies, while reading σ up to position j, which p_i's hold at j.

Using this basic construction, it is straightforward to build a κ-automaton corresponding to a κ-formula.

For example, for the simple reactivity formula $\Box\Diamond\, p_1 \vee \Diamond\Box\, p_2$, let the automaton mentioned above be $\langle Q, q_o, \delta\rangle$ with the designated sets F_1 and F_2. Then the corresponding simple reactivity automaton is:

$$\langle Q, q_0, \delta, \{(F_1, F_2\}\rangle$$

In the other direction, not every property specifiable by an automaton can be specified in temporal logic. Only a restricted class of automata, called *counter-free* automata (see [MP71]) can be translated into temporal logic. This phenomena was first pointed out in [Wol83]. A Streett automaton is defined to be *counter-free* if there do not exist finite words $\sigma_0, \sigma \in \Sigma^+$ and some $n > 1$ such that $\widehat{\delta}(\sigma_0 \cdot \sigma^n) = \widehat{\delta}(\sigma_0)$ but $\widehat{\delta}(\sigma_0 \cdot \sigma) \neq \widehat{\delta}(\sigma_0)$. This condition prevents counter-free automata from counting occurrences of σ modulo n. It has been shown in [Zuc86] (additional details are provided in [MP90a]) that:

Proposition 5.11 *A property is specifiable by temporal logic iff it is specifiable by a counter-free automaton.*

In particular, it can be shown that the κ-automata constructed in the proof of Theorem 5.10 are all counter-free. This result can be used to provide a translation from counter-free κ-automata to κ-formulas.

Theorem 5.12 *A property that is specifiable by a counter-free κ-automaton is specifiable by a canonical κ-formula, where $\kappa \in \{$ safety, guarantee, obligation, response, persistence, reactivity $\}$.*

The translation is essentially the one studied in [Zuc86]. It is based on the construction of a past formula φ_q for each $q \in Q - \{q_0\}$ of a given counter-free automaton. The formula φ_q characterizes all the finite computations leading from q_0 to q, i.e., for each computation σ and position $j \geq 0$:

$$\widehat{\delta}(\sigma[0..j]) = q \quad \textit{iff} \quad (\sigma, j) \models \varphi_q$$

For example, the formula corresponding to the (counter-free) reactivity automaton $\langle Q, q_0, \delta, \{(R, P)\}\rangle$ is:

$$\Box\Diamond\Big(\bigvee_{q\in R} \varphi_q\Big) \quad \vee \quad \Diamond\Box\Big(\bigvee_{q\in P} \varphi_q\Big)$$

The above two-way translation, subject to counter-freedom, provides a standard reduction of results about automata into corresponding results about temporal logic. We can use this reduction to prove the missing direction in the proof of Theorem 4.3, which can be stated as:

A κ-property that is specifiable by a temporal formula is specifiable by a canonical κ-formula.

Proof: Let φ be the formula specifying Π. Using the translation described in Theorem 5.10, we construct an automaton $\mathcal{M}_\varphi$ specifying the κ-property Π which, by Proposition 5.11, is counter-free. By Theorem 5.1 we can construct a κ-automaton $\widehat{\mathcal{M}}$ that specifies the same property. A close examination of the transformations used in the proof of Theorem 5.1 shows that they preserve counter-freedom. Consequently, by Theorem 5.12, we can construct a canonical κ-formula $\varphi_{\widehat{\mathcal{M}}}$ specifying Π. ▪

References

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

[AS87] B. Alpern and F.B. Schneider. Recognizing safety and liveness. *Dist. Comp.*, 2:117–126, 1987.

[AS89] B. Alpern and F.B. Schneider. Verifying temporal properties without temporal logic. *ACM Trans. Prog. Lang. Sys.*, 11:147–167, 1989.

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

[Kam85] M. Kaminski. A classification of ω-regular languages. *Theor. Comp. Sci.*, 36:217–229, 1985.

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

[Lam83] L. Lamport. What good is temporal logic. In R.E.A. Mason, editor, *Proc. IFIP 9th World Congress*, pages 657–668. North-Holland, 1983.

[Lan69] L.H. Landweber. Decision problems for ω-automata. *Math. Sys. Theory*, 4:376–384, 1969.

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

[Man74] Z. Manna. *Mathematical Theory of Computation*. McGraw-Hill, 1974.

[MP71] R. McNaughton and S. Papert. *Counter Free Automata*. MIT Press, 1971.

[MP81] Z. Manna and A. Pnueli. Verification of concurrent programs: The temporal framework. In R.S. Boyer and J.S. Moore, editors, *The Correctness Problem in Computer Science*, pages 215–273. Academic Press, London, 1981.

[MP83] Z. Manna and A. Pnueli. How to cook a temporal proof system for your pet language. In *Proc. 10th ACM Symp. Princ. of Prog. Lang.*, pages 141–154, 1983.

[MP84] Z. Manna and A. Pnueli. Adequate proof principles for invariance and liveness properties of concurrent programs. *Sci. Comp. Prog.*, 32:257–289, 1984.

[MP87] Z. Manna and A. Pnueli. Specification and verification of concurrent programs by ∀-automata. In *Proc. 14th ACM Symp. Princ. of Prog. Lang.*, pages 1–12, 1987.

[MP89] Z. Manna and A. Pnueli. The anchored version of the temporal framework. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, pages 201–284. Lec. Notes in Comp. Sci. 354, Springer-Verlag, 1989.

[MP90a] O. Maler and A. Pnueli. Tight bounds on the complexity of cascaded decomposition of automata. In *Proc. 31th IEEE Symp. Found. of Comp. Sci.*, pages 672–682, 1990.

[MP90b] Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *Proc. 9th ACM Symp. Princ. of Dist. Comp.*, pages 377–408, 1990.

[MP91] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer Verlag, New York, 1991.

[OL82] S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Trans. Prog. Lang. Sys.*, 4:455–495, 1982.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. Found. of Comp. Sci.*, pages 46–57, 1977.

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

[Str82] R.S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Inf. and Cont.*, 54:121–141, 1982.

[Wag79] K. Wagner. On ω-regular sets. *Inf. and Cont.*, 43:123–177, 1979.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Inf. and Cont.*, 56:72–99, 1983.

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