Runtime Verification

Grigore Roşu

University of Illinois at Urbana-Champaign

Contents

Topics to cover:

- Safety properties and their monitoring. How many safety properties are there? Can they all be monitored? Complexity of monitoring in different formalism: LTL, RE, ERE, CFG, SRS. Both dynamic properties, like above, and static properties (e.g., complex heap patterns).
- Event/Trace observation. How to observe the execution of a program? Instrumentation vs. runtime environment.
- Monitor synthesis. Generating optimal monitors for several formalisms: LTL, FT/PT-LTL, RE, ERE, CFG, SRS, PT-CaRet, Allen TL.
- Parametric property monitoring. How to deal with multiple instances of monitors?
- Predictive runtime analysis. Vector clock vs SMT-based techniques.
- Static analysis to improve runtime verification. Improve runtime overhead by not instrumenting what is unnecessary. Improve prediction capability by looking beyond the trace.
- Semantics-based Runtime Verification. Defining a formal language semantics. Using a semantics to do symbolic execution and runtime verify properties. Ultimate goal: verify programs by exhaustive runtime verification.

Chapter 1

Introduction

Begin of intro stuff for chapter on safety

From SACS: ***Abstract sacs:*** *This paper addresses the problem of runtime verification from a foundational perspective, answering questions like "Is there a consensus among the various definitions of a safety property?" (Answer: Yes), "How many safety properties exist?" (Answer: As many as real numbers), "How difficult is the problem of monitoring a safety property?" (Answer: Arbitrarily complex), "Is there any formalism that can express all safety properties?" (Answer: No), etc. Various definitions of safety properties as sets of execution traces have been proposed in the literature, some over finite traces, others over infinite traces, yet others over both finite and infinite traces. By employing cardinality arguments and a novel notion of* persistence*, this paper first establishes the existence of bijective correspondences between the various notions of safety property. It then shows that safety properties can be characterized as "always past" properties. Finally, it proposes a general notion of* monitor*, which allows to show that safety properties correspond precisely to the monitorable properties, and then to establish that monitoring a safety property is arbitrarily hard.*

From safety: ***Abstract safety:*** *Various definitions of safety properties as sets of execution traces have been introduced in the literature, some over finite traces, others over infinite traces, yet others over both finite and infinite traces. By employing cardinality arguments, this paper first shows that these notions of safety are ultimately equivalent, by showing each of them to have the cardinal of the continuum. It is then shown that all safety properties can be characterized as "always past" properties, and then that the problem of monitoring a safety property can be arbitrarily hard. Finally, two decidable specification formalisms for safety properties are discussed, namely extended regular expressions and past time LTL. It is shown that monitoring the former requires non-elementary space. An optimal monitor synthesis algorithm is given for the latter; the generated monitors run in space linear with the number of temporal operators and in time linear with the size of the formula.*

A *safety property* is a behavioral property which, once violated, cannot be satisfied anymore. For example, a property "always $x > 0$" is violated when $x \leq 0$ is observed for the first time; this safety property remains violated even though eventually $x > 0$ might hold. That means that one can identify each safety property with a set of "bad" finite execution traces, with the intuition that once one of those is reached the safety property is violated.

There are several apparently different ways to formalize safety. Perhaps the most immediate one is to complement the "bad traces" above and thus to define a safety property as a prefix-closed property over finite traces (containing the "good traces") – by "property" in this paper we mean a set of finite or infinite traces. Inspired by Lamport [11], Alpern and Schneider [4] define safety properties over infinite traces as ones with the property that if an infinite trace is unacceptable then there must be some finite prefix of it which is already unacceptable, in the sense that there is no acceptable infinite completion of it. Is there any relationship between these two definitions of safety? We show rather indirectly that there is, by showing that their corresponding sets of safety properties have the cardinal c of the continuum (i.e., the cardinal of $\mathbb{R}$, the set of real numbers), so there exists some bijective mapping between the two. Unfortunately, the existence of such a bijection is

as little informative as the existence of a bijection between the real numbers and the irrational numbers. To capture the relationship between finite- and infinite-trace safety properties in a meaningful way, we introduce a subset of finite-trace safety properties, called *persistent*, and then construct an explicit bijection between that subset and the infinite-trace safety properties. Interestingly, over finite traces there are as many safety properties as unrestricted properties (finite-traces are enumerable and $\mathcal{P}(\mathbb{N})$ is in bijection with $\mathbb{R}$), while over infinite traces there are c safety properties versus 2^c unrestricted properties (infinite traces are in bijection with $\mathbb{R}$).

It is also common to define safety properties as properties over both finite and infinite traces, the intuition for the finite traces being that of unfinished computations. For example, Lamport [12] extends the notion of infinite-trace safety properties to properties over both finite and infinite traces, while Schneider et al. [17, 8] give an alternative definition of safety over finite and infinite traces, called "execution monitoring". One immediate technical advantage of allowing both finite and infinite traces is that one can define prefix-closed properties. We indirectly show that prefix-closeness is not a sufficient condition to define safety properties when infinite traces are also allowed, by showing that there are 2^c prefix-closed properties versus, as expected, "only" c safety properties.

Another common way to specify safety properties is as "always past" properties, that is, as properties containing only words whose finite prefixes satisfy a given property. If P is a property on finite prefixes, then we write $\Box P$ for the "always P" safety property containing the words with prefixes in P. We show that specifying safety properties as "always past" properties is fully justified by showing that, for each of the three types of traces (finite, infinite, and both), the "always past" properties are precisely the safety properties as defined above. It is common to specify P using some logical formalism, for example past time linear temporal logic (past LTL) [13]; for example, one can specify "a before b" in past LTL as the formula $b \rightarrow \diamond a$.

The problem of monitoring safety properties is also investigated in this paper. Since there are as many safety properties as real numbers, it is not unexpected that some of them can be very hard to monitor. We show that the problem of monitoring a safety property is arbitrarily hard, by showing that it reduces to deciding membership of natural

numbers to a set of natural numbers. In particular, we can associate a safety property to any degree in the arithmetic hierarchy as well as to any complexity class in the decidable universe, whose monitoring is as hard as that degree or complexity class.

From SACS: *This paper makes three novel contributions, two technical and another pedagogical. On the technical side, it first introduces the notion of a* persistent *safety property, which appears to be the right finite-trace correspondent of an infinite-trace safety property, and uses it to show the cardinal equivalence of the various notions of safety property encountered in the literature. Also on the technical side, it rigorously defines the problem of monitoring a safety property, and it shows that it can be arbitrarily hard. On the pedagogical side, this paper offers the first comprehensive study and uniform presentation of safety properties and of their monitoring.*

From safety: *In practice not all ($c = |\mathbb{R}|$) safety properties are meaningful, but only those ($\aleph_0 = |\mathbb{N}|$) which are specifiable using formal specification languages or logics of interest. We also investigate the problem of monitoring safety properties expressed using two common formalisms, namely regular expressions extended with complement, also called extended regular expressions (ERE), and LTL. It is known that both formalisms allow polynomial finite-trace membership checking algorithms [9, 15] if one has random access to the trace, but that both require exponential space if the trace can only be analyzed online [14, 10]. It is also known that LTL can indeed be monitored in exponential space [7] and so is claimed[1] for EREs in [14]. We show that the claim in [14] is, unfortunately, wrong, by showing that ERE monitoring requires non-elementary space. To do so, we propose for any $n \in \mathbb{N}$ a safety property P_n whose monitoring requires space non-elementary in n, as well as an ERE of size $O(n^3)$. Since the known monitoring algorithms for LTL in its full generality are asymptotically optimal, what is left to do is to consider important fragments of LTL. We focus on the "always past" fragment and give a monitor synthesis algorithm that takes formulae φ and generate monitors for them that need $O(k)$ total space and $O(|\varphi|)$ time to process each event, where k is the number of past operators in φ. This improves over the best known algorithm that needs space $O(|\varphi|$ (and same time).*

End of intro stuff for chapter on safety

Chapter 2

Background, Preliminaries, Notations

Add some structure to this chapter

We let $\mathbb{N}$ denote the set of natural numbers including 0 but excluding the infinity symbol ∞ and let $\mathbb{N}_\infty$ denote the set $\mathbb{N} \cup \{\infty\}$. We also let $\mathbb{Q}$ denote the set of rational numbers and $\mathbb{R}$ the set of real numbers; as for natural numbers, the "∞" subscript can also be added to $\mathbb{Q}$ and $\mathbb{R}$ for the corresponding extensions of these sets. $\mathbb{Q}^+$ and $\mathbb{R}^+$ denote the sets of strictly positive (0 not included) rational and real numbers, respectively.

We fix a set Σ of elements called *events* or *states*. We call words in $\Sigma^\star$ *finite traces* and those in Σ^ω *infinite traces*. If $u \in \Sigma^\star \cup \Sigma^\omega$ then u_i is the i-th state or event that appears in u. We call *finite-trace properties* sets $P \subseteq \Sigma^*$ of finite traces, *infinite-trace properties* sets $P \subseteq \Sigma^\omega$ of infinite traces, and just *properties* sets $P \subseteq \Sigma^* \cup \Sigma^\omega$ of finite or infinite traces. If the finite or infinite aspect of traces is understood from context, then we may call any of the types or properties above just *properties*. We may write $P(w)$ for a property P and a (finite or infinite) trace w whenever $w \in P$. Traces and properties are more commonly called *words* and *languages*, respectively, in the literature; we prefer to call them traces and properties to better reflect the intuition that our target application is monitoring and system observance, not formal languages. We take, however, the liberty to also call them words and languages whenever that terminology seems more appropriate.

In some cases states can be simply identified with their names, or labels, and specifications of properties on traces may just refer to those labels. For

example, the regular expression $(s_1 \cdot s_2)^\star$ specifies all those finite traces starting with state s_1 and in which states s_1 and s_2 alternate. In other cases, one can think of states as sets of atomic predicates, that is, predicates that hold in those states: if s is a state and a is an atomic predicate, then we say that $a(s)$ is true iff a "holds" in s; thus, if all it matters with respect to states is which predicates hold and which do not hold in each state, then states can be faithfully identified with sets of predicates. We prefer to stay loose with respect to what "holds" means, because, depending on the context, it can mean anything. In conventional software situations, atomic predicates can be: boolean expressions over variables of the program, their satisfaction being decided by evaluating them in the current state of the program; or whether a function is being called or returned from; or whether a particular variable is being written to; or whether a particular lock is being held by a particular thread; and so on. In the presence of atomic predicates, specifications of properties on traces typically only refer to the atomic predicates. For example, the property "always a before b", that is, those traces containing no state in which b holds that is not preceded by some state in which a holds (for example, a can stand for "authentication" and b for "resource access"), can be expressed in LTL as the formula $\Box(b \rightarrow \diamond a)$.

Let us recall some basic notions and notations from formal languages, temporarily using the consecrated terminology of "words" and "languages" instead of traces and properties. For an alphabet Σ, let $\mathcal{L}_\Sigma$ be the set of languages over Σ, i.e., the powerset $\mathcal{P}(\Sigma^\star)$. By abuse of language and notation, let $\emptyset$ be the empty language $\{\}$ and ϵ the language containing only the empty word, $\{\epsilon\}$. If $L_1, L_2 \in \mathcal{L}_\Sigma$ then $L_1 \cdot L_2$ is the language $\{\alpha_1\alpha_2 \mid \alpha_1 \in L_1 \text{ and } \alpha_2 \in L_2\}$. Note that $L \cdot \emptyset = \emptyset \cdot L = \emptyset$ and $L \cdot \epsilon = \epsilon \cdot L = L$. If $L \in \mathcal{L}_\Sigma$ then $L^\star$ is $\{\alpha_1\alpha_2 \cdots \alpha_n \mid n \geq 0 \text{ and } \alpha_1, \alpha_2, \ldots, \alpha_n \in L\}$ and $\neg L$ is $\Sigma^\star - L$.

We next recall some notions related to cardinality. If A is any set, we let $|A|$ denote the *cardinal* of A, which expresses the size of A. When A is finite, $|A|$ is precisely the number of elements of A and we call it a *finite cardinal*. Infinite sets can have different cardinals, called *transfinite* or even *infinite*. For example, natural numbers $\mathbb{N}$ have the cardinal $\aleph_0$ (pronounced "aleph zero") and real numbers $\mathbb{R}$ have the cardinal c, also called the *cardinal of the continuum*. Two sets A and B are said to have the same cardinal, written $|A| = |B|$, iff there is some bijective mapping between the two. We write $|A| \leq |B|$ iff there is some injective mapping from A to B.

The famous *Cantor-Bernstein-Schroeder theorem* states that if $|A| \leq |B|$

and $|B| \leq |A|$ then $|A| = |B|$. In other words, to show that there is some bijection between sets A and B, it suffices to find an injection from A to B and an injection from B to A. The two injections need not be bijections. For example, the inclusion of the interval $(0, 1)$ in $\mathbb{R}^+$ is obviously an injection, so $|(0, 1)| \leq |\mathbb{R}^+|$. On the other hand, the function $x \rightsquigarrow x/(2x + 1)$ from $\mathbb{R}^+$ to $(0, 1)$ (in fact its codomain is the interval $(0, 1/2)$) is also injective, so $|\mathbb{R}^+| \leq |(0, 1)|$. Neither of the two injective functions is bijective, yet by the Cantor-Bernstein-Schroeder theorem there is some bijection between $(0, 1)$ and $\mathbb{R}^+$, that is, $|(0, 1)| = |\mathbb{R}^+|$. We will use this theorem to relate the various types of safety properties; for example, we will show that there is an injective function from safety properties over finite traces to safety properties over infinite traces and another injective function in the opposite direction. Unfortunately, the Cantor-Bernstein-Schroeder theorem is existential: it only says that some bijection exists between the two sets, but it does not give us an explicit bijection. Since the visualization of a concrete bijection between different sets of safety properties can be very meaningful, we will avoid using the Cantor-Bernstein-Schroeder theorem when we can find an explicit bijection between two sets of safety properties.

If A is a set of cardinal α, then 2^α is the cardinal of $\mathcal{P}(A)$, the power set of A (the set of subsets of A). It is known that $2^{\aleph_0} = c$, that is, there are as many sets of natural numbers as real numbers. The famous, still unanswered *continuum hypothesis*, states that there is no set whose size is strictly between $\aleph_0$ and c; more generally, it states that, for any transfinite cardinal α, there is no proper cardinal between α and 2^α. If A and B are infinite sets, then $|A| + |B|$ and $|A| \cdot |B|$ are the cardinals of the sets $A \cup B$ and $A \times B$, respectively. An important property of transfinite cardinals is that of *absorption* – the larger cardinal absorbs the smaller one: if α and β are transfinite cardinals such that $\alpha \leq \beta$, then $\alpha + \beta = \alpha \cdot \beta = \beta$; in particular, $c \cdot 2^c = 2^c$. Besides sets of natural numbers, there are several other important sets that have cardinal c: streams (i.e., infinite sequences) of Booleans, streams of reals, non-empty closed or open intervals of reals, as well as the sets of all open or closed sets of reals, respectively (Exercise 1).

For our purposes, if Σ is an enumerable set of states, then $\Sigma^\star$ is also enumerable, so it has cardinal $\aleph_0$. Also, if $|\Sigma| \leq c$, in particular if it is finite, then Σ^ω has the cardinal c, because it is equivalent to streams of states. We can then immediately infer that the set of finite-trace properties over Σ has cardinal $2^{\aleph_0} = c$, while the set of infinite-trace properties has cardinal 2^c.

Exercises

Exercise 1 *Show that each of the following sets have cardinal c: streams (i.e., infinite sequences) of Booleans; streams of natural numbers; streams of real numbers; closed intervals of real numbers; open intervals of real numbers; closed sets of real numbers; open sets of real numbers.*

Chapter 3

Safety Properties

Intuitively, a safety property of a system is one stating that the system cannot "go wrong", or, as Lamport [11] put it, that the "bad thing" never happens. In other words, in order for a system to violate a safety property, it should eventually "go wrong" or the "bad thing" should eventually happen. There is a very strong relationship between safety properties and runtime monitoring: if a safety property is violated by a running system, then the violation should happen *during* the execution of the system, in a finite amount of time, so a monitor for that property observing the running system should be able to detect the violation; an additional point in the favor of monitoring is that, if a system violates a safety property at some moment during its execution, then there is no way for the system to continue its execution to eventually satisfy the property, so a monitor needs not wait for a better future once it detects a bad present/past.

State properties or assertions that need only the current state of the running system to check whether they are violated or not, such as "no division by 0", or "x positive", or no deadlock, are common safety properties; once violated, one can stop the computation or take corrective measures. However, there are also interesting safety properties that involve more than one state of the system, such as "if one uses resource x then one must have authenticated at some moment in the past", or "any start of a process must be followed by a stop within 10 units of time", or "take command from user only if the user has logged in at some moment in the past and has not logged out since then", etc. Needless to say that the atomic events, or states, which form execution traces on which safety properties are defined, can be quite abstract: not all the details of a system execution are relevant

for the particular safety property of interest. In the context of monitoring, these relevant events or states can be extracted by means of appropriate instrumentation of the system. For example, runtime monitoring systems such as Tracematches [3] and MOP [6] use aspect-oriented technology to "hook" relevant observation points and appropriate event filters in a system.

It is customary to define safety properties as properties over *infinite traces*, to capture the intuition that they are defined for systems that can potentially run forever, such as reactive systems. A point in favor of infinite traces is that finite traces can be regarded as special cases of infinite traces, namely ones that "stutter" indefinitely in their last state (see, for example, Abadi and Lamport [1, 2]). Infinite traces are particularly desirable when one specifies safety properties using formalisms that have infinite-trace semantics, such as linear temporal logics or corresponding automata.

While "infinity" is a convenient abstraction that is relatively broadly-accepted nowadays in mathematics and in theoretical foundations of computer science, there is no evidence so far that a system can have an infinite-trace behavior (we have not seen any). A disclaimer is in place here: we do *not* advocate finite-traces as a foundation for safety properties; all we try to do is to argue that, just because they can be seen as a special case of infinite traces, finite traces are not entirely uninteresting. For example, a safety property associated to a one-time-access key issued to a client can be "activate, then use at most once, then close". Using regular patterns over the alphabet of relevant events $\Sigma = \{activate, use, close\}$, this safety property can be expressed as "$activate \cdot (\epsilon + use) \cdot close$"; any trace that is not a prefix of the language of this regular expression violates the property, including any other activation or use of the key after it was closed. While these finite-trace safety properties can easily be expressed as infinite-trace safety properties, we believe that that would be more artificial than simply accepting that in practice we deal with many finite-trace safety properties.

In this section we discuss various approaches to formalize safety properties and show that they are ultimately directly or indirectly equivalent. We categorize them into finite-trace safety properties, infinite-trace safety properties, and finite- and infinite-trace safety properties:

1. Section 3.1 defines safety properties over finite traces as prefix closed properties. A subset of finite-trace safety properties, that we call *persistent*, contain only traces that "have a future" within the property, that is, finite traces that can be continued into other finite traces that are also in the safety property. Persistent safety properties appear to

be the right finite-trace variant that corresponds faithfully to the more conventional infinite-trace safety properties. Even though persistent safety properties form a proper subset of finite-trace safety properties and each finite-trace safety property has a largest persistent safety property included in it, we show that there is in fact a bijection between safety properties and persistent safety properties by showing them both to have the cardinal of the continuum c.

2. In Section 3.2, we consider two standard infinite-trace definitions of a safety property, one based on the intuition that violating behaviors must manifest so after a finite number of events and the other based on the intuition of a safety property as a closed set in an appropriate topology over infinite-traces. We show them both equivalent to persistent safety properties over finite traces, by constructing an explicit bijection (as opposed to using cardinality arguments and infer the existence of a bijection); consequently, infinite-trace safety properties also have the cardinal of the continuum c. Since closed sets of real numbers are in a bijective correspondence with the real numbers, we indirectly rediscover Alpern and Schneider's result [4] stating that infinite-trace safety properties correspond to closed sets in infinite-trace topology.

3. Section 3.3 considers safety properties defined over both finite and infinite traces. We discuss two definitions of such safety properties encountered in the literature, and, using cardinality arguments, we show their equivalence with safety properties over only finite traces. In particular, safety properties over finite and infinite traces also have the cardinality of the continuum c. We also show that prefix-closeness is not a sufficient condition to characterize (not even bijectively) such safety properties, by showing that there are significantly more (2^c) prefix-closed properties over finite and infinite traces than safety properties.

Therefore, each of the classes of safety properties is in bijection with the real numbers. Since there are so many safety properties, we can also insightfully conclude that there is *no* enumerable mechanism to define all the safety properties, because $\aleph_0 \lneq c$. Therefore, particular logical or syntactic recursive formalisms can only define *some* of the safety properties, but not all of them.

3.1 Finite Traces

One of the most common intuitions for a safety property is as a prefix-closed set of finite traces. This captures best the intuition that once something bad happened, there is no way to recover: if $w \notin P$ then there is no u such that $P(wu)$, which is equivalent to saying that if $P(wu)$ then $P(w)$, which is equivalent to saying that P is prefix closed. From a monitoring perspective, a prefix closed property can be regarded as one containing all the good (complete or partial) behaviors of the observed system: once a state is encountered that does not form a good behavior together with the previously observed states, then a violation can be reported.

Definition 1 *Let* $\mathit{prefixes} : \Sigma^\star \to \mathcal{P}(\Sigma^\star)$ *be the prefix function returning for any finite trace all its prefixes, and let* $\mathit{prefixes} : \mathcal{P}(\Sigma^\star) \to \mathcal{P}(\Sigma^\star)$ *be its corresponding closure operator that takes sets of finite traces and closes them under prefixes.*

Note that $\mathsf{prefixes} : \mathcal{P}(\Sigma^\star) \to \mathcal{P}(\Sigma^\star)$ is indeed a closure operator (Exercise 2): it is extensive ($P \subseteq \mathsf{prefixes}(P)$), monotone ($P \subseteq P'$ implies $\mathsf{prefixes}(P) \subseteq \mathsf{prefixes}(P')$), and idempotent ($\mathsf{prefixes}(\mathsf{prefixes}(P)) = \mathsf{prefixes}(P)$).

Definition 2 *Let* $\mathit{Safety}^\star$ *be the set of finite-trace prefix-closed properties, that is, the set* $\{P \in \mathcal{P}(\Sigma^\star) \mid P = \mathit{prefixes}(P)\}$. *In other words,* $\mathit{Safety}^\star$ *is the set of fixed points of the prefix operator* $\mathit{prefixes} : \mathcal{P}(\Sigma^\star) \to \mathcal{P}(\Sigma^\star)$.

The star superscript in $\mathsf{Safety}^\star$ reflects that its traces are finite; in the next section we will define a set Safety^ω of infinite-trace safety properties. Since $\mathsf{prefixes}(P) \in \mathsf{Safety}^\star$ for any $P \in \mathcal{P}(\Sigma^*)$, we can assume from here on that $\mathsf{prefixes} : \mathcal{P}(\Sigma^\star) \to \mathcal{P}(\Sigma^\star)$ is actually a function $\mathcal{P}(\Sigma^*) \to \mathsf{Safety}^\star$.

Example 1 Consider the one-time-access key safety property discussed above, saying that a client can "activate, then use at most once, and then close" the key. If $\Sigma = \{\mathit{activate}, \mathit{use}, \mathit{close}\}$, then this safety property can be expressed as the finite set of finite words

$$\{\epsilon, \mathit{activate}, \mathit{activate\ close}, \mathit{activate\ use}, \mathit{activate\ use\ close}\}$$

No other behavior is allowed. Now suppose that the safety policy is extended to allow multiple uses of the key once activated, but still no further events

once it is closed. The extended safety property has now infinitely many finite-traces:

$$\{\epsilon\} \cup \{activate\} \cdot \{use^n \mid n \in \mathbb{N}\} \cdot \{\epsilon, close\}.$$

Note that this property is indeed prefix-closed. A monitor in charge of online checking this safety property would report a violation if the first event is not *activate*, or if it encounters any second *activate* event, or if it encounters any event after a *close* event is observed, including another *close* event.

It is interesting to note that this finite-trace safety property encompasses both finite and infinite aspects. For example, it does not preclude behaviors in which one sees an *activate* event and then an arbitrary number of *use* events; *use* events can persist indefinitely after an *activate* event without violating the property. On the other hand, once a *close* event is encountered, no other event can be further seen. We will shortly see that the safety property above properly includes the *persistent* safety property $\{\epsilon\} \cup \{activate\, use^n \mid n \in \mathbb{N}\}$, which corresponds to the infinite-trace safety property $\{activate\, use^\omega\}$. □

While prefix closeness seems to be the right requirement for a safety property, one can argue that it is not sufficient. For example, in the context of reactive systems that supposedly run forever, one may think of a safety property as one containing safe finite traces, that is, ones for which the reactive system can always find a way to continue its execution safely. The definition of safety properties above includes, among other safety properties, the empty set of traces as well as all prefix-closed *finite* sets of finite traces; any reactive system will eventually violate such safety properties, so one can say that the definition of safety property above is too generous.

We next define *persistent safety properties* as ones that always allow a future; intuitively, an observed reactive system that is in a safe state can always (if persistent enough) find a way to continue its execution to a next safe state. This notion is reminiscent of "feasibility", a semantic characterization of fairness in [5], and of "machine closeness" [1, 16], also used in the context of fairness.

Definition 3 *Let* $\mathsf{PersistentSafety}^\star$ *be the set of finite-trace* persistent safety properties*, that is, safety properties* $P \in \mathsf{Safety}^\star$ *such that if* $P(w)$ *for some* $w \in \Sigma^\star$ *then there is some* $a \in \Sigma$ *such that* $P(wa)$*.*

If a persistent safety property is non-empty, then note that it must contain an infinite number of words. The persistency aspect of a finite-trace safety

property can be regarded, in some sense, as a liveness argument. Indeed, assuming that it is a "good thing" for a trace to be indefinitely continued, then a persistent safety property is one in which the "good thing" always eventually happens. If one takes the liberty to regard "stuck" computations as unfair, then the persistency aspect above can also be regarded as a fairness argument.

Another way to think of persistent safety properties is as a means to refer to infinite behaviors by means of finite traces. This view is, in some sense, dual to the more common approach to regard finite behaviors as infinite behaviors that stutter infinitely in a "last" state (see, for example, Abadi and Lamport [1, 2] for a formalization of such last-state infinite stuttering).

Note that if Σ is a degenerate set of events containing only one element, that is, if $|\Sigma| = 1$, then $|\mathsf{Safety}^\star| = \aleph_0$ and $|\mathsf{PersistentSafety}^\star| = 2$; indeed, if $\Sigma = \{a\}$ then $\mathsf{Safety}^\star$ contains precisely the finite properties $a^{\leq n} = \{a^i \mid 0 \leq i \leq n\}$ for each $n \in \mathbb{N}$ plus the infinite property $\{a^n \mid n \in \mathbb{N}\}$, so a total of $\aleph_0 + 1 = \aleph_0$ properties, while $\mathsf{PersistentSafety}^\star$ contains only two properties, namely $\emptyset$ and $\{a^n \mid n \in \mathbb{N}\}$. The case when there is only one event or state in Σ is neither interesting nor practical. Therefore, from here on in this paper we take the liberty to assume that $|\Sigma| \geq 2$. Since in practice Σ contains states or events generated by a computer, for simplicity in stating some of the subsequent results, we also take the liberty to assume that $|\Sigma| \leq \aleph_0$; therefore, Σ can be any finite or recursively enumerable set, including $\mathbb{N}$, $\mathbb{N}_\infty$, $\mathbb{Q}$, etc., but cannot be $\mathbb{R}$ or any set "larger" than $\mathbb{R}$. With these assumptions, it follows that $|\Sigma^\star| = \aleph_0$ (finite words are recursively enumerable) and $|\Sigma^\omega| = c$ (infinite streams have the cardinality of the continuum).

Proposition 1 *$\mathsf{Safety}^\star$ and $\mathsf{PersistentSafety}^\star$ are closed under union; $\mathsf{Safety}^\star$ is also closed under intersection.*

Proof: The union and the intersection of prefix-closed properties is also prefix-closed. Also, the union of persistent prefix-closed properties is also persistent. □

The intersection of persistent safety properties may not be persistent:

Example 2 Let Σ be the set $\{0, 1\}$. Let $P = \{1^m \mid m \in \mathbb{N}\}$ and $P' = \{\epsilon\} \cup \{10^m \mid m \in \mathbb{N}\}$ be two persistent safety properties, where ϵ is the empty word (the word containing no letters). Then $P \cap P'$ is the finite safety property $\{\epsilon, 1\}$, which is not persistent. If one thinks that this happened because $P \cap P'$ does not contain any proper (i.e., non-empty)

persistent property, then one can take instead the persistent safety properties $P = \{0^n \mid n \in \mathbb{N}\} \cdot \{1^m \mid m \in \mathbb{N}\}$ and $P' = \{0^n \mid n \in \mathbb{N}\} \cdot (\{\epsilon\} \cup \{10^m \mid m \in \mathbb{N}\}$, whose intersection is the safety property $\{0^n \mid n \in \mathbb{N}\} \cup \{0^n 1 \mid n \in \mathbb{N}\}$. This safety property is not persistent because its words ending in 1 cannot persist, but it contains the proper persistent safety property $\{0^n \mid n \in \mathbb{N}\}$. □

Therefore, we can associate to any safety property in $\mathsf{Safety}^\star$ a largest persistent safety property in $\mathsf{PersistentSafety}^\star$, by simply taking the union of all persistent safety properties that are included in the original safety property (the empty property is one of them, the smallest):

Definition 4 *For a safety property $P \in \mathsf{Safety}^\star$, let $P^\circ \in \mathsf{PersistentSafety}^\star$ be the largest persistent safety property with $P^\circ \subseteq P$.*

The following example shows that one may need to eliminate infinitely many words from a safety property in order to obtain a persistent safety property:

Example 3 Let $\Sigma = \{0, 1\}$ and let P be the safety property $\{0^n \mid n \in \mathbb{N}\} \cup \{0^n 1 \mid n \in \mathbb{N}\}$. Then P° can contain no word ending with a 1 and can contain all the words of 0's. Therefore, $P^\circ = \{0^n \mid n \in \mathbb{N}\}$. □

Finite safety properties obviously cannot contain any non-empty persistent safety property, that is, $P^\circ = \emptyset$ if P is finite. But what if P is infinite? Is it always the case that it contains a non-empty persistent safety property? Interestingly, it turns out that this is true if and only if Σ is finite:

Proposition 2 *If Σ is finite and P is a safety property containing infinitely many words, then $P^\circ \neq \emptyset$.*

Proof: For each letter $a \in \Sigma$, let us define the *derivative of P wrt a*, written $\delta_a(P)$, as the language $\{w \in \Sigma^\star \mid aw \in P\}$. Since

$$P = \{\epsilon\} \cup \bigcup_{a \in \Sigma} \{a\} \cdot \delta_a(P)$$

since Σ is finite, and since P is infinite, it follows that there is some $a_1 \in \Sigma$ such that $\delta_{a_1}(P)$ is infinite; note that $a_1 \in P$ since P is prefix closed. Similarly, since $\delta_{a_1}(P)$ is infinite, there is some $a_2 \in \Sigma$ such that $\delta_{a_2}(\delta_{a_1}(P))$ is infinite and $a_1 a_2 \in P$. Iterating this reasoning, we can find some $a_n \in \Sigma$

for each $n \in \mathbb{N}$, such that $a_1 a_2 \dots a_n \in P$ and $\delta_{a_n}(\cdots(\delta_{a_2}(\delta_{a_1}(P)))\cdots)$ is infinite, that is, the set $\{w \in \Sigma^\star \mid a_1 a_2 \dots a_n w \in P\}$ is infinite. It is now easy to see that the set $\{a_1 a_2 \dots a_n \mid n \in \mathbb{N}\} \subseteq P$ is persistent. Therefore, $P^\circ \neq \emptyset$. □

The following example shows that Σ must indeed be finite in order for the result above to hold:

Example 4 Consider some infinite set of events or states Σ. Then we can label distinct elements in Σ with distinct labels in $\mathbb{N} \cup \{\infty\}$. We only need these elements from Σ; therefore, without loss of generality, we can assume that $\Sigma = \mathbb{N} \cup \{\infty\}$. Let P be the safety property

$$\{\epsilon\} \cup \{\infty\, n\, (n-1) \dots (m+1)\, m \mid 0 \leq m \leq n+1\},$$

where ϵ is the empty word (the word containing no letters) and $n \dots (n+1)$ is also the empty word for any $n \in \mathbb{N}$. Then P° is the empty property. Indeed, note that any persistent safety property P' included in P cannot have traces ending in 0, because those cannot be continued into other traces in P; since P' cannot contain traces ending in 0, it cannot contain traces ending in 1 either, because such traces can only be continued with a 0 letter into traces in P, but those traces have already been decided that cannot be part of P'; inductively, one can show that P' can contain no words ending in letters that are natural numbers in $\mathbb{N}$. Since the only trace in P ending in ∞ is ∞ itself and since ∞ can only be continued with a natural number letter into a trace in P but such trace cannot belong to P', we deduce that P' can contain no word with letters in Σ. In particular, P° must be empty. □

Even though we know that the largest persistent safety property P° included into a safety property P always exists because $\mathsf{PersistentSafety}^\star$ is closed under union, we would like to have a more constructive way to obtain it. A first and obvious thing to do is to eliminate from P all the "stuck" computations, that is, those which cannot be added any new state to obtain a trace that is also in P. This removal step does not destroy the prefix-closeness of P, but it may reveal new computations which are stuck. By iteratively eliminating all the computations that get stuck in a finite number of steps, one would expect to obtain a persistent safety property, namely precisely P°. It turns out that this is indeed true only if Σ is finite. If that is the case, then the following can also be used as an alternative definition of P°:

Proposition 3 *Given safety property $P \in \mathsf{Safety}^\star$, then let P^- be the property $\{w \in P \mid (\exists a \in \Sigma)\, wa \in P\}$. Also, let $\{P_i \mid i \in \mathbb{N}\}$ be properties defined as $P_0 = P$ and $P_{i+1} = P_i^-$ for all $i \geq 0$. Then $P^\circ = \bigcap_{i \geq 0} P_i$ whenever Σ is finite.*

Proof: It is easy to see that if P is prefix-closed then $P^- \subseteq P$ is also prefix-closed, so P^- is also a property in $\mathsf{Safety}^\star$. Therefore, the properties P_i form a sequence $P = P_0 \supseteq P_1 \supseteq P_2 \supseteq \cdots$ of increasingly smaller safety properties.

Let us first prove that $\bigcap_{i \geq 0} P_i$ is a persistent safety property. Assume by contradiction that for some $w \in \bigcap_{i \geq 0} P_n$ there is no $a \in \Sigma$ such that $wa \in \bigcap_{i \geq 0} P_i$. In other words, we can find for each $a \in \Sigma$ some $i_a \geq 0$ such that $wa \notin P_{i_a}$. Since Σ is finite, we can let i be the largest among the natural numbers $i_a \in \mathbb{N}$ for all $a \in \Sigma$. Since $P_i \subseteq P_{i_a}$ for all $a \in \Sigma$, it should be clear that there is no $a \in \Sigma$ such that $wa \in P_i$, which means that $w \notin P_{i+1}$. This contradicts the fact that $w \in \bigcap_{i \geq 0} P_i$. Therefore, $\bigcap_{i \geq 0} P_i \in \mathsf{PersistentSafety}^\star$.

Let us now prove that $\bigcap_{i \geq 0} P_i$ is the largest persistent safety property included in P. Let P' be any persistent safety property included in P. We show by induction on i that $P' \subseteq P_i$ for all $i \in \mathbb{N}$. The base case, $P' \subseteq P_0$, is obvious. Suppose that $P' \subseteq P_i$ for some $i \in \mathbb{N}$ and let $w \in P'$. Since P' is persistent, there is some $a \in \Sigma$ such that $wa \in P' \subseteq P_i$, which means that $w \in P_{i+1}$. Since w was chosen arbitrarily, it follows that $P' \subseteq P_{i+1}$. Therefore, $P' \subseteq \bigcap_{i \geq 0} P_i$. □

We next show that the finiteness of Σ was a necessary requirement in order for the result above to hold. In other words, we show that if Σ is allowed to be infinite then we can find a safety property $P \in \mathsf{Safety}^\star$ over Σ such that $P^\circ \in \mathsf{PersistentSafety}^\star$ and $\bigcap_{i \geq 0} P_i \in \mathsf{Safety}^\star$ are distinct. Since we showed in the proof of Proposition 3 that any persistent safety property P' is included in $\bigcap_{i \geq 0} P_i$, it follows that $P^\circ \subseteq \bigcap_{i \geq 0} P_i$. Since P° is the largest persistent safety property included in P, one can easily show that $P^\circ = (\bigcap_{i \geq 0} P_i)^\circ$. Therefore, it suffices to find a safety property P such that $\bigcap_{i \geq 0} P_i$ is not persistent, which is what we do in the next example:

Example 5 Consider the safety property P over infinite $\Sigma = \mathbb{N} \cup \{\infty\}$ discussed in Example 4, namely $\{\epsilon\} \cup \{\infty\, n\, (n-1) \ldots (m+1)\, m \mid 0 \leq m \leq n+1\}$. Then one can easily show by induction on $i \in \mathbb{N}$ that the properties P_i defined in Proposition 3 are the sets $\{\epsilon\} \cup \{\infty\, n\, (n-1) \ldots (m+1)\, m \mid i \leq m \leq n+1\}$; in other words, each P_i excludes from P all the words whose

last letters are smaller than i when regarded as natural numbers. Then the intersection $\bigcap_{i\geq 0} P_i$ contains no trace ending in a natural number; the only possibility left is then $\bigcap_{i\geq 0} P_i = \{\epsilon, \infty\}$, which is different from $P^\circ = \emptyset$ (see Example 4).

One may argue that $P^\circ \neq \bigcap_{i\geq 0} P_i$ above happened precisely because P° was empty. One can instead pick the safety property $Q = \{0^n \mid n \in \mathbb{N}\} \cdot P$. Then one can show following the same idea as in Example 4 that $Q^\circ = \{0^n \mid n \in \mathbb{N}\}$. Further, one can show that $Q_i = \{0^n \mid n \in \mathbb{N}\} \cdot P_i$, so $\bigcap_{i\geq 0} Q_i = \{0^n \mid n \in \mathbb{N}\} \cup \{0^n\infty \mid n \in \mathbb{N}\}$, which is different from Q°. $\square$

Persistency is reminiscent of "feasibility" introduced by Apt et al. [5] in the context of fairness, and of "machine closeness" introduced by Abadi and Lamport [1, 2] (see also Schneider [16]) in the context of refinement. Let us use the terminology "machine closeness": a property L (typically a liveness or a fairness property) is *machine closed* for a property M (typically given as the language of some state machine) iff L does not prohibit any of the observable runtime behaviors of M, that is, iff $\mathsf{prefixes}(M) = \mathsf{prefixes}(M \cap L)$; for example, if M is the total property (i.e., every event is possible at any moment, i.e., $M = \Sigma^\star$) and L is the property stating that "always eventually event a", then any prefix of M can be continued to obtain a property satisfying L. Persistency is related to machine closeness in that a safety property P is persistent if and only if P° is machine closed for P. In other words, there is nothing P can do in a finite amount of time that P° cannot do. However, there is a caveat here: since liveness and fairness are inherently infinite-trace notions, machine closeness (or feasibility) have been introduced in the context of infinite-traces. On the other hand, persistency makes sense only in the context of finite traces.

It is clear that $\mathsf{PersistentSafety}^\star$ is properly included in $\mathsf{Safety}^\star$. Yet, we next show that, surprisingly, there is a bijective correspondence between $\mathsf{Safety}^\star$ and $\mathsf{PersistentSafety}^\star$, both having the cardinal of the continuum:

Theorem 1 $|\mathit{PersistentSafety}^\star| = |\mathit{Safety}^\star| = c$.

Proof: Since $\Sigma^\star$ is recursively enumerable and since $2^{\aleph_0} = c$, we can readily infer that $|\mathsf{PersistentSafety}^\star| \leq |\mathsf{Safety}^\star| \leq |\mathcal{P}(\Sigma^\star)| = c$.

Let us now define an injective function φ from the open interval of real numbers $(0, 1)$ to $\mathsf{PersistentSafety}^\star$. Since $|\Sigma| \geq 2$, let us distinguish two different elements in Σ and let us label them $\overline{0}$ and $\overline{1}$. For a real $r \in (0, 1)$, let $\varphi(r)$ be the set $\{\overline{\alpha} \mid \alpha \in \{0, 1\}^\star \text{ and } 0.\alpha < r\}$, where $0.\alpha$

is the (rational) number in $(0,1)$ whose decimals in binary representation are α, and where $\overline{\alpha}$ is the word in $\Sigma^\star$ corresponding to α. Note that the set $\varphi(r) \in \mathcal{P}(\Sigma^\star)$ is prefix-closed for any $r \in (0,1)$, and that if $w \in \varphi(r)$ then also $w\overline{0} \in \varphi(r)$ (the latter holds since, by real numbers conventions, $0.\alpha = 0.\alpha 0$), so $\varphi(r) \in \mathsf{PersistentSafety}^\star$. Since the set of rationals with finite number of decimals in binary representation is dense in $\mathbb{R}$ (i.e., it intersects any open interval in $\mathbb{R}$) and in particular in the interval $(0,1)$, it follows that the function $\varphi : (0,1) \to \mathsf{PersistentSafety}^\star$ is injective: indeed, if $r_1 \neq r_2 \in (0,1)$, say $r_1 < r_2$, then there is some $\alpha \in \{0,1\}^\star$ such that $r_1 < 0.\alpha < r_2$, so $\varphi(r_1) \neq \varphi(r_2)$. Since the interval $(0,1)$ has the cardinal of the continuum c, the existence of the injective function φ implies that $c \leq |\mathsf{PersistentSafety}^\star|$. By the Cantor-Bernstein-Schroeder theorem it follows that $|\mathsf{PersistentSafety}^\star| = |\mathsf{Safety}^\star| = c$. □

From safety: *The proof above could have been rearranged to avoid the need to use the set PersistentSafety*. However, we prefer to keep it for two reasons:*

1. *For finite-traces, persistent safety properties appear to be more natural in the context of reactive systems than just prefix closed properties;*
2. *Persistent safety properties play a technical bridge role in the next section to show that the infinite-trace safety properties also have the cardinal c.*

With regards to finite-traces, persistent safety properties appear to be more natural in the context of reactive systems than just prefix-closed properties. Also, persistent safety properties play a technical bridge role in the next section to show that the infinite-trace safety properties also have the cardinal c.

3.2 Infinite Traces

The finite-trace safety properties defined above, persistent or not, rely on the intuition of a correct prefix: a safety property is identified with the set of all its finite prefixes. In the case of a persistent safety property, each "informal" infinite acceptable behavior is captured by its infinite set of finite prefixes. Even though persistent safety properties appear to capture well in a finite-trace setting the intuition of safety in the context of (infinite-trace) reactive

systems, one could argue that it does not say anything about unacceptable infinite traces. Indeed, one may think that persistent safety properties do not capture the intuition that if an infinite trace is unacceptable then there must be some finite prefix of it which is already unacceptable. In this section we show that there is in fact a bijection between safety properties over infinite traces and persistent safety properties over finite traces as we defined them in the previous section.

We start by extending the $\mathsf{prefixes}$ function to infinite traces:

Definition 5 *Let* $\mathit{prefixes} : \Sigma^\omega \to \mathcal{P}(\Sigma^\star)$ *be the function returning for any infinite trace* u *all its finite prefixes* $\mathit{prefixes}(u)$*, and let* $\mathit{prefixes} : \mathcal{P}(\Sigma^\omega) \to \mathcal{P}(\Sigma^\star)$ *be its corresponding extension to sets of infinite traces.*

Note that $\mathsf{prefixes}(S) \in \mathsf{PersistentSafety}^\star$ for any $S \in \mathcal{P}(\Sigma^\omega)$, so $\mathsf{prefixes}$ is in fact a function $\mathcal{P}(\Sigma^\omega) \to \mathsf{PersistentSafety}^\star$.

The definition of safety properties over infinite traces below appears to be the most used definition of a safety property in the literature; at our knowledge, it was formally introduced by Alpern and Schneider [4], but they credit the insights of their definition to Lamport [11].

Definition 6 *Let* Safety^ω *be the set of infinite-trace properties* $Q \in \mathcal{P}(\Sigma^\omega)$ *s.t.: if* $u \notin Q$ *then there is a finite trace* $w \in \mathit{prefixes}(u)$ *s.t.* $wv \notin Q$ *for any* $v \in \Sigma^\omega$*.*

In other words, if an infinite behavior violates the safety property then there is some finite-trace "violation threshold"; once the violation threshold is reached, there is no chance to recover.

The following proposition can serve as an alternative and more compact definition of Safety^ω:

Proposition 4 $\mathit{Safety}^\omega = \{Q \in \mathcal{P}(\Sigma^\omega) \mid u \in Q \text{ iff } \mathit{prefixes}(u) \subseteq \mathit{prefixes}(Q)\}$.

Proof: Since $u \in Q$ implies $\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(Q)$, the only thing left to show is that $Q \in \mathsf{Safety}^\omega$ iff "$\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(Q)$ implies $u \in Q$"; the latter is equivalent to "$u \notin Q$ implies $\mathsf{prefixes}(u) \not\subseteq \mathsf{prefixes}(Q)$", which is further equivalent to "$u \notin Q$ implies there is some $w \in \mathsf{prefixes}(u)$ s.t. $w \notin \mathsf{prefixes}(Q)$", which is indeed equivalent to $Q \in \mathsf{Safety}^\omega$. □

Another common intuition for safety properties over infinite traces is as *closed* sets in the topology corresponding to Σ^ω. Alpern and Schneider captured formally this intuition for the first time in [4]; then it was used as a convenient definition of safety by Abadi and Lamport [1, 2] among others:

Definition 7 *An infinite sequence $u^{(1)}$, $u^{(2)}$, ..., of infinite traces in Σ^ω* converges to $u \in \Sigma^\omega$, *or u is a* limit of $u^{(1)}$, $u^{(2)}$, ...,, *written $u = \lim_i u^{(i)}$, iff for all $m \geq 0$ there is an $n \geq 0$ such that $u_1^{(i)} u_2^{(i)} \dots u_m^{(i)} = u_1 u_2 \dots u_m$ for all $i \geq n$. If $Q \in \mathcal{P}(\Sigma^\omega)$ then $\overline{Q}$, the* closure *of Q, is the set $\{\lim_i u^{(i)} \mid u^{(i)} \in Q \text{ for all } i \in \mathbb{N}\}$.*

It can be easily shown that the overline closure above is indeed a closure operator on Σ^ω, that is, it is extensive ($Q \subseteq \overline{Q}$), monotone ($Q \subseteq Q'$ implies $\overline{Q} \subseteq \overline{Q'}$), and idempotent ($\overline{\overline{Q}} = \overline{Q}$); see Exercise 6.

Definition 8 *Let $\mathsf{Safety}^\omega_{\lim}$ be the set of properties $\{Q \in \mathcal{P}(\Sigma^\omega) \mid Q = \overline{Q}\}$.*

As expected, the two infinite-trace safety property definitions are equivalent; we have not found any formal proof in the literature, so for the sake of completeness we give a simple proof here:

Proposition 5 $\mathsf{Safety}^\omega_{\lim} = \mathsf{Safety}^\omega$.

Proof: All we need to prove is that for any $Q \in \mathcal{P}(\Sigma^\omega)$ and any $u \in \Sigma^\omega$, $\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(Q)$ iff $u = \lim_i u^{(i)}$ for some infinite sequence of infinite traces $u^{(1)}$, $u^{(2)}$, ... in Q. If $\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(Q)$ then one can find for each $i \geq 0$ some $u^{(i)} \in Q$ such that $u_1 u_2 \dots u_i = u_1^{(i)} u_2^{(i)} \dots u_i^{(i)}$, so for each $m \geq 0$ one can pick $n = m$ such that $u_1 u_2 \dots u_m = u_1^{(i)} u_2^{(i)} \dots u_m^{(i)}$ for all $i \geq n$, so $u = \lim_i u^{(i)}$. Conversely, if $u = \lim_i u^{(i)}$ for some infinite sequence of infinite traces $u^{(1)}$, $u^{(2)}$, ... in Q, then for any $m \geq 0$ there is some $n \geq 0$ such that $u_1 u_2 \dots u_m = u_1^{(n)} u_2^{(n)} \dots u_m^{(n)}$, that is, for any prefix of u there is some $u' \in Q$ having the same prefix, that is, $\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(Q)$. □

The next result establishes the relationship between infinite-trace safety properties and finite-trace persistent safety properties, by proposing a concrete bijective mapping relating the two (as opposed to using cardinality arguments to indirectly show only the existence of such a mapping). Therefore, there is also a bijective correspondence between safety properties over infinite traces and the real numbers:

Note there are 2^c properties over Σ^ω

Theorem 2 $|\mathsf{Safety}^\omega| = |\mathsf{PersistentSafety}^\star| = c$.

Proof: We show that there is a bijective function between the two sets of safety properties. Recall that $\mathsf{prefixes}(S) \in \mathsf{PersistentSafety}^\star$ for any $S \in \mathcal{P}(\Sigma^\omega)$, that is, that $\mathsf{prefixes}$ is a function $\mathcal{P}(\Sigma^\omega) \to \mathsf{PersistentSafety}^\star$. Let $\mathsf{prefixes} : \mathsf{Safety}^\omega \to \mathsf{PersistentSafety}^\star$ be the restriction of this prefix function to Safety^ω. Let us also define a function $\omega : \mathsf{PersistentSafety}^\star \to \mathsf{Safety}^\omega$ as follows: $\omega(P) = \{u \in \Sigma^\omega \mid \mathsf{prefixes}(u) \subseteq P\}$. This function is well-defined: if $u \notin \omega(P)$ then by the definition of $\omega(P)$ there is some $w \in \mathsf{prefixes}(u)$ such that $w \notin P$; since $w \in \mathsf{prefixes}(wv)$ for any $v \in \Sigma^\omega$, it follows that $wv \notin \omega(P)$ for any $v \in \Sigma^\omega$.

We next show that $\mathsf{prefixes}$ and ω are inverse to each other. Let us first show that $\mathsf{prefixes}(\omega(P)) = P$ for any $P \in \mathsf{PersistentSafety}^\star$. The inclusion $\mathsf{prefixes}(\omega(P)) \subseteq P$ follows by the definition of $\omega(P)$: $\mathsf{prefixes}(u) \subseteq P$ for any $u \in \omega(P)$. The inclusion $P \subseteq \mathsf{prefixes}(\omega(P))$ follows from the fact that P is a persistent safety property: for any $w \in P$ one can iteratively build an infinite sequence v_1, v_2, ..., such that $wv_1,\ wv_1v_2,\ ... \in P$, so $wv_1v_2... \in \omega(P)$. Let us now show that $\omega(\mathsf{prefixes}(Q)) = Q$ for any $Q \in \mathsf{Safety}^\omega$. The inclusion $Q \subseteq \omega(\mathsf{prefixes}(Q))$ is immediate. For the other inclusion, let $u \in \omega(\mathsf{prefixes}(Q))$, that is, $\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(Q)$. Suppose by contradiction that $u \notin Q$. Then there is some $w \in \mathsf{prefixes}(u)$ such that $wv \notin Q$ for any $v \in \Sigma^\omega$. Since $w \in \mathsf{prefixes}(u)$ and $\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(Q)$, it follows that $w \in \mathsf{prefixes}(Q)$, that is, that there is some $u' \in Q$ such that $u' = wv$ for some $v \in \Sigma^\omega$. This contradicts the fact that $wv \notin Q$ for any $v \in \Sigma^\omega$. Consequently, $u \in Q$.

The second part follows by Theorem 1. □

3.3 Finite and Infinite Traces

It is also common to define safety properties as properties over both finite and infinite traces, the intuition for the finite traces being that of unfinished computations. For example, Lamport [12] extends the notion of safety in Definition 6 to properties over both finite and infinite traces, while Schneider et al [17, 8] give an alternative definition of safety over finite and infinite traces. We define both approaches shortly and then show their equivalence and their bijective correspondence with real numbers. Before that, we argue that the mix of finite and infinite traces is less trivial than it may appear, by showing that there are significantly more prefix closed properties than in the case when only finite traces were considered.

Definition 9 *Let $\mathsf{PrefixClosed}^{\star,\omega}$ be the set of prefix-closed sets of finite and infinite traces: for $Q \subseteq \Sigma^\star \cup \Sigma^\omega$, $Q \in \mathsf{PrefixClosed}^{\star,\omega}$ iff $\mathsf{prefixes}(Q) \subseteq Q$.*

Also, let $\mathsf{PersistentPrefixClosed}^{\star,\omega}$ *be the set of* persistent *prefix-closed sets of finite and infinite traces: for* $Q \in \mathsf{PrefixClosed}^{\star,\omega}$*, it is the case that* $Q \in \mathsf{PersistentPrefixClosed}^{\star,\omega} \iff$ *if* $Q(w)$ *for some* $w \in \Sigma^\star$ *then that there is some* $a \in \Sigma$ *such that* $Q(wa)$*.*

The next result says that there is a bijective correspondence between prefix-closed and persistent prefix-closed properties also in the case of finite and infinite traces, but that there are exponentially more such properties than in the case of just finite traces:

Proposition 6 $|\mathsf{PersistentPrefixClosed}^{\star,\omega}| = |\mathsf{PrefixClosed}^{\star,\omega}| = 2^c$*.*

Proof: We show $2^c \leq |\mathsf{PersistentPrefixClosed}^{\star,\omega}| \leq |\mathsf{PrefixClosed}^{\star,\omega}| \leq 2^c$, where the middle inequality is immediate. For $2^c \leq |\mathsf{PersistentPrefixClosed}^{\star,\omega}|$, let us define $\varphi : \mathcal{P}((0,1)) \to \mathsf{PersistentPrefixClosed}^{\star,\omega}$ as

$$\varphi(R) = \bigcup_{0.\alpha \in R} \{\overline{\alpha}\} \cup \mathsf{prefixes}(\overline{\alpha})$$

where we assume for any real number in the interval $(0,1)$ its decimal binary representation $0.\alpha$ with $\alpha \in \{0,1\}^\omega$ (if the number is rational then α may contain infinitely many ending 0's), and $\overline{\alpha}$ is the infinite trace in Σ^ω replacing each 0 and 1 in α by $\overline{0}$ and $\overline{1}$, respectively, where $\overline{0}$ and $\overline{1}$ are two arbitrary but fixed distinct elements in Σ (recall that $|\Sigma| \geq 2$. Note that $\varphi(R)$ is well-defined: it is clearly prefix-closed and it is also persistent because its finite traces are exactly prefixes of infinite traces, so they admit continuations in $\varphi(R)$. It is easy to see that φ is injective. Since $|(0,1)| = c$, we conclude that $2^c \leq |\mathsf{PersistentPrefixClosed}^{\star,\omega}|$.

To show $|\mathsf{PrefixClosed}^{\star,\omega}| \leq 2^c$, note that any property in $\mathsf{PrefixClosed}^{\star,\omega}$ is a union of a subset in $\Sigma^\star$ and a subset in Σ^ω, so $|\mathsf{PrefixClosed}^{\star,\omega}| \leq 2^{|\Sigma^\star|} \cdot 2^{|\Sigma^\omega|}$. Since $|\Sigma^\star| = \aleph_0$, $|\Sigma^\omega| = c$, $2^{\aleph_0} = c$, and $c \cdot 2^c = 2^c$ (by absorption of transfinite cardinals), we get that $|\mathsf{PrefixClosed}^{\star,\omega}| \leq 2^c$. □

The fact that properties in $\mathsf{PersistentPrefixClosed}^{\star,\omega}$ contain also infinite traces was crucial in showing the injectivity of φ in the proof above. A similar construction for the finite trace setting does *not* work. Indeed, if one tries to define a function $\varphi : \mathcal{P}((0,1)) \to \mathsf{PersistentSafety}^\star$ as $\varphi(R) = \bigcup_{0.\alpha \in R} \mathsf{prefixes}(\overline{\alpha})$, then one can show it well-defined but cannot show it injective: e.g., $\varphi((0,0.5)) = \varphi((0,0.5])$.

Since safety properties over finite and infinite traces are governed by the same intuitions as safety properties over only finite or over only infinite

traces, the result above tells us that prefix closeness is not a sufficient condition to properly capture the safety properties. Schneider [17] proposes an additional condition in the context of his EM (execution monitoring) framework, namely that if an infinite trace is not in the property, then there is a finite prefix of it which is not in the property either. It is easy to see that this additional condition is equivalent to saying that an infinite trace is in the property whenever all its finite prefixes are in the property, which allows us to compactly define safety properties over finite and infinite traces in the EM style as follows:

Definition 10 $\mathit{Safety}_{\mathsf{EM}}^{\star,\omega} = \{Q \subseteq \Sigma^\star \cup \Sigma^\omega \mid u \in Q \textit{ iff } \mathit{prefixes}(u) \subseteq Q\}$.

Note that $\mathsf{Safety}_{\mathsf{EM}}^{\star,\omega} \subset \mathsf{PrefixClosed}^{\star,\omega}$. We will shortly show that $\mathsf{Safety}_{\mathsf{EM}}^{\star,\omega}$ is in fact much smaller than $\mathsf{PrefixClosed}^{\star,\omega}$, by showing that $|\mathsf{Safety}_{\mathsf{EM}}^{\star,\omega}| = c$.

The consecrated definition of a safety property in the context of both finite and infinite traces is perhaps the one proposed by Lamport in [12], which relaxes the one in Definition 6 by allowing u to range over both finite and infinite traces:

Definition 11 *Let* $\mathit{Safety}^{\star,\omega}$ *be the set of finite- and infinite-trace properties* $\{Q \subseteq \Sigma^\star \cup \Sigma^\omega \mid u \notin Q \Rightarrow (\exists w \in \mathit{prefixes}(u))\,(\forall v \in \Sigma^\star \cup \Sigma^\omega)\; wv \notin Q\}$

Schneider informally stated in [17] that the two definitions of safety above are equivalent. It is not hard to show it formally:

Proposition 7 $\mathit{Safety}_{\mathsf{EM}}^{\star,\omega} = \mathit{Safety}^{\star,\omega}$.

Proof: First note that $\mathsf{Safety}^{\star,\omega} \subseteq \mathsf{PrefixClosed}^{\star,\omega}$: if $wu \in Q \in \mathsf{Safety}^{\star,\omega}$ and $w \notin Q$ then there is some $w' \in \mathsf{prefixes}(w)$, say $w = w'w''$, such that $w'v \notin Q$ for any v, in particular $w'w''u \notin Q$, which contradicts $wu \in Q$.

$\mathsf{Safety}^{\star,\omega} \subseteq \mathsf{Safety}_{\mathsf{EM}}^{\star,\omega}$: let $Q \in \mathsf{Safety}^{\star,\omega}$ and $u \in \Sigma^\star \cup \Sigma^\omega$ s.t. $\mathsf{prefixes}(u) \subseteq Q$; if $u \notin Q$ then there is some $w \in \mathsf{prefixes}(u)$ s.t. $wv \notin Q$ for any v, in particular for v the empty word, that is, $w \notin Q$, which contradicts $\mathsf{prefixes}(u) \subseteq Q$.

$\mathsf{Safety}_{\mathsf{EM}}^{\star,\omega} \subseteq \mathsf{Safety}^{\star,\omega}$: let $u \notin Q \in \mathsf{Safety}_{\mathsf{EM}}^{\star,\omega}$; then $\mathsf{prefixes}(u) \not\subseteq Q$, that is, there is some $w \in \mathsf{prefixes}(u)$ s.t. $w \notin Q$; since Q is prefix-closed, it follows that $wv \notin Q$ for any $v \in \Sigma^\star \cup \Sigma^\omega$. □

We next show that there is a bijective correspondence between the safety properties over finite or infinite traces above and the finite trace safety properties in Section 3.1:

Theorem 3 $|\mathit{Safety}^{\star,\omega}| = |\mathit{Safety}^{\star,\omega}_{\mathsf{EM}}| = |\mathit{Safety}^{\star}| = c$.

Proof: $\mathsf{Safety}^{\star} \subset \mathsf{Safety}^{\star,\omega}_{\mathsf{EM}}$ since the properties in $\mathsf{Safety}^{\star,\omega}_{\mathsf{EM}}$ are prefix-closed, so $|\mathsf{Safety}^{\star}| \leq |\mathsf{Safety}^{\star,\omega}_{\mathsf{EM}}|$.

Since the functions $\mathsf{prefixes} : \mathcal{P}(\Sigma^{\star}) \to \mathcal{P}(\Sigma^{\star})$ and $\mathsf{prefixes} : \mathcal{P}(\Sigma^{\omega}) \to \mathcal{P}(\Sigma^{\star})$ have actual co-domains $\mathsf{Safety}^{\star}$ and $\mathsf{PersistentSafety}^{\star}$, respectively, they can be organized as a function $\mathsf{prefixes} : \mathsf{Safety}^{\star,\omega}_{\mathsf{EM}} \to \mathsf{Safety}^{\star}$. Let us show that this function is injective. Let us assume $Q \neq Q' \in \mathsf{Safety}^{\star,\omega}_{\mathsf{EM}}$, say $u \in Q$ and $u \notin Q'$, s.t. $\mathsf{prefixes}(Q) = \mathsf{prefixes}(Q')$. Since $u \in Q \in \mathsf{Safety}^{\star,\omega}_{\mathsf{EM}}$ it follows that $\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(Q) \subseteq Q$, which implies that $\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(Q') \subseteq Q'$; since $Q' \in \mathsf{Safety}^{\star,\omega}_{\mathsf{EM}}$, it follows that $u \in Q'$, contradiction. Therefore, $\mathsf{prefixes} : \mathsf{Safety}^{\star,\omega}_{\mathsf{EM}} \to \mathsf{Safety}^{\star}$ is injective, which proves that $\mathsf{Safety}^{\star,\omega}_{\mathsf{EM}} \leq \mathsf{Safety}^{\star}$.

The rest follows by Proposition 7 and Theorem 1. □

3.4 "Always Past" Characterization

Another common way to specify safety properties is by giving an arbitrary property on finite traces, not necessarily prefix closed, and then to require that any acceptable behavior must have all its finite prefixes in the given property. A particularly frequent case is when one specifies the property of the finite-prefixes using the past-time fragment of linear temporal logics (LTL). For example, Manna and Pnueli [13] call the resulting "always (past LTL)" properties *safety formulae*; many other authors, including ourselves, adopted the terminology "safety formula" from Manna and Pnueli, although some qualify it as "LTL safety formula". An example of an LTL safety formula is "always (b implies eventually in the past a)", written using LTL notation as "$\Box(b \to \diamond a)$"; here the past time formula "$b \to \diamond a$" compactly specifies all the finite-traces

$$\{wsw's' \mid w, w' \in \Sigma^{\star},\ s, s' \in \Sigma,\ a(s) \text{ and } b(s') \text{ hold}\} \bigcup$$
$$\{ws \mid w \in \Sigma^{\star},\ s \in \Sigma,\ b(s) \text{ does not hold}\}.$$

From safety: *We will investigate the case when safety properties are expressed as LTL safety formulae, as well as optimal monitoring techniques for such safety properties, in Section* **??**.

In the remainder of this section we assume that the past time prefix properties are given as ordinary sets of finite-traces (so we make abstraction of how

these properties are expressed) and show not only that the resulting "always past" properties are safety properties, but also that any safety properties can be expressed as an "always past" property. This holds for all the variants of safety properties (i.e., over finite traces, over infinite traces, or over both finite and infinite traces).

Definition 12 *Let $P \subseteq \Sigma^\star$ be any property over finite traces. Then we define the "always past" property $\Box P$ as follows:*

(finite traces) $\{w \in \Sigma^\star \mid \mathsf{prefixes}(w) \subseteq P\}$*; and*
(infinite traces) $\{u \in \Sigma^\omega \mid \mathsf{prefixes}(u) \subseteq P\}$*; and*
(finite and infinite traces) $\{u \in \Sigma^\star \cup \Sigma^\omega \mid \mathsf{prefixes}(u) \subseteq P\}$.

Let $\mathsf{Safety}^\star_\Box$, $\mathsf{Safety}^\omega_\Box$ and $\mathsf{Safety}^{\star,\omega}_\Box$ be the corresponding sets of properties.

From safety: *In Section ?? we show that the language $\mathcal{L}(\Box\varphi)$, that corresponds to the LTL safety formula $\Box\varphi$ for φ some past-time LTL formula, is a property in $\mathsf{Safety}^\omega_\Box$. If one was interested in a finite-trace or in a both finite and infinite trace semantics of LTL, then one could have shown that $\mathcal{L}(\Box\varphi) \in \mathsf{Safety}^\star_\Box$ or that $\mathcal{L}(\Box\varphi) \in \mathsf{Safety}^{\star,\omega}_\Box$.*

Intuitively, one can regard the square "$\Box$" as a closure operator. Technically, it is not precisely a closure operator because it does not operate on the same set: it takes finite-trace properties to any of the three types of properties considered. Since prefixes takes properties back to finite-trace properties, we can show the following result saying that the square is a "closure operator via prefixes", and that safety properties are precisely the sets of words which are closed this way:

Proposition 8 *The following hold for all three types of safety properties:*

- $\Box(\mathsf{prefixes}(\Box P)) = \Box P$ *for any* $P \subseteq \Sigma^\star$*;*
- Q *is a safety property iff* $\Box(\mathsf{prefixes}(Q)) = Q$.

Proof: Left as an exercise to the reader. See Exercise 5. □

We next show that the "always past" properties are all safety properties and, moreover, that any safety property can be expressed as an "always past" property:

Theorem 4 *The following hold:*

- $\mathit{Safety}^{\star}_{\Box} = \mathit{Safety}^{\star}$,
- $\mathit{Safety}^{\omega}_{\Box} = \mathit{Safety}^{\omega}$, *and*
- $\mathit{Safety}^{\star,\omega}_{\Box} = \mathit{Safety}^{\star,\omega}$.

Therefore, each of the "always past" safety properties have the cardinal c.

Proof: We prove each of the equalities by double inclusion.

$\mathsf{Safety}^{\star}_{\Box} \subseteq \mathsf{Safety}^{\star}$. It is true because any property $\Box P$ in $\mathsf{Safety}^{\star}_{\Box}$ is prefix-closed.

$\mathsf{Safety}^{\star} \subseteq \mathsf{Safety}^{\star}_{\Box}$. If $P \in \mathsf{Safety}^{\star}$ then we claim that $P = \Box P$, so $P \in \mathsf{Safety}^{\star}_{\Box}$. Indeed, since P is prefix-closed, $\mathsf{prefixes}(w) \subseteq P$ for any $w \in P$, so $w \in \Box P$; also, since $w \in \mathsf{prefixes}(w)$, it follows that for any $w \in \Box P$, $w \in P$.

$\mathsf{Safety}^{\omega}_{\Box} \subseteq \mathsf{Safety}^{\omega}$. Let $\Box P$ be an "always past" property in $\mathsf{Safety}^{\omega}_{\Box}$, and let u be an infinite trace in Σ^{ω} such that $u \notin \Box P$. Then it follows that $\mathsf{prefixes}(u) \not\subseteq P$, that is, there is some $w \in \mathsf{prefixes}(u)$ such that $w \notin P$. Since $w \in \mathsf{prefixes}(wv)$ for any $v \in \Sigma^{\omega}$, it means that there is no $v \in \Box P$ such that $\mathsf{prefixes}(wv) \subseteq P$, that is, there is no $v \in \Sigma^{\omega}$ such that $wv \in \Box P$. Therefore, $\Box P \in \mathsf{Safety}^{\omega}$.

$\mathsf{Safety}^{\omega} \subseteq \mathsf{Safety}^{\omega}_{\Box}$. If $Q \in \mathsf{Safety}^{\omega}$ then we claim that $Q = \Box\mathsf{prefixes}(Q)$. The inclusion $Q \subseteq \Box\mathsf{prefixes}(Q)$ is clear, because $u \in Q$ implies $\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(Q)$. For the other inclusion, note that if $\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(Q)$ for some $u \in \Sigma^{\omega}$, then u must be in Q: if $u \notin Q$ then by the definition of $Q \in \mathsf{Safety}^{\omega}$, there is some $w \in \mathsf{prefixes}(u)$ which cannot be completed into an infinite trace in Q, which contradicts $\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(Q)$.

$\mathsf{Safety}^{\star,\omega}_{\Box} \subseteq \mathsf{Safety}^{\star,\omega}$. By Proposition 7, it suffices to show that $\mathsf{Safety}^{\star,\omega}_{\Box} \subseteq \mathsf{Safety}^{\star,\omega}_{\mathsf{EM}}$. Let $\Box P$ be an "always past" property in $\mathsf{Safety}^{\star,\omega}_{\Box}$, and let $u \in \Sigma^{\star} \cup \Sigma^{\omega}$ such that $\mathsf{prefixes}(u) \subseteq \mathsf{prefixes}(\Box P)$. Since $\mathsf{prefixes}(\Box P) \subseteq P$, it follows that $u \in \Box P$; therefore, $\Box P \in \mathsf{Safety}^{\star,\omega}_{\mathsf{EM}}$.

$\mathsf{Safety}^{\star,\omega} \subseteq \mathsf{Safety}^{\star,\omega}_{\Box}$. It is straightforward to see that $Q \in \mathsf{Safety}^{\star,\omega}_{\mathsf{EM}}$ implies $Q = \Box\mathsf{prefixes}(Q)$.

The cardinality part follows by Theorems 1, 2, and 3. □

Proposition 8 and Theorem 4 give yet another characterization for safety properties over any of the three combinations of traces, namely one in the style of the equivalent formulation of safety over infinite traces in Proposition 4: Q is a safety property iff it contains precisely the words whose prefixes are in $\mathsf{prefixes}(Q)$.

Exercises

Exercise 2 *The* $\mathsf{prefixes}\colon \mathcal{P}(\Sigma^\star) \to \mathcal{P}(\Sigma^\star)$ *in Definition 1 is a closure operator: it is extensive (*$P \subseteq \mathsf{prefixes}(P)$*), monotone (*$P \subseteq P'$ *implies* $\mathsf{prefixes}(P) \subseteq \mathsf{prefixes}(P')$*), and idempotent (*$\mathsf{prefixes}(\mathsf{prefixes}(P)) = \mathsf{prefixes}(P)$*).*

Exercise 3 *(Counter-)Example 4 showed that the finiteness of* Σ *was necessary in order for Proposition 2 to hold, by defining a property* P *over* $\Sigma = \mathbb{N} \cup \{\infty\}$ *in which all non-empty words start with* ∞*. Can we remove* ∞ *from* Σ *and from all the words in* P*? Why, or why not?*

Exercise 4 *Same like Exercise 3, but for Example 5 instead of Example 4.*

Exercise 5 *Prove Proposition 8.*

Exercise 6 *The "closure under limits" operation in Definition 7 is indeed a closure operator on* Σ^ω*: it is extensive (*$Q \subseteq \overline{Q}$*), monotone (*$Q \subseteq Q'$ *implies* $\overline{Q} \subseteq \overline{Q'}$*), and idempotent (*$\overline{\overline{Q}} = \overline{Q}$*).*

Chapter 4

Monitoring

In this section we give yet another characterization of safety properties, namely as monitorable properties. Specifically, we formally define a monitor as a (possibly infinite) state machine without final states but with a partial transition function, and then we show that safety properties are precisely the properties that can be monitored with such monitors. We then elaborate on the problem of defining the complexity of monitoring a safety property, discussing some pitfalls and guiding principles, and show that monitoring a safety property can be an arbitrarily hard problem. Finally, we give a more compact and mathematical equivalent definition of a monitor, which may be useful in further foundational efforts in this area.

Relate our definition of a monitor with Schneider's security automata

---

4.1 Specifying Safety Properties as Monitors

Safety properties are difficult to work with as flat sets of finite or infinite words, not only because they can contain infinitely many words, but also because such a flat representation is inconvenient for further analysis. It is important therefore to *specify* safety properties using formalisms that are easier to represent and reason about.

From safety: *The next sections in this paper investigate several dedicated formalisms that proved to be convenient in specifying safety, such as finite state machines, regular expressions and temporal logics, together with corresponding limitations and efficient monitor synthesis techniques.*

Formalisms known to be useful for specifying safety properties include regular expressions and temporal logics, which can be efficiently translated into finite-state machines which can then be used as monitors. In this section we formalize the intuitive notion of a *monitor* as a special state machine and give yet another characterization of safety properties, namely as *monitorable properties*. Since monitorable properties are completely defined by their monitors, it follows that *all* safety properties can be specified by their corresponding monitors.

Recall that we work under the assumption that Σ is a set of events or program states such that $|\Sigma| \leq \aleph_0$.

Definition 13 *A* Σ-monitor*, or just a* monitor *(when* Σ *is understood), is a triple* $\mathcal{M} = (S, s_0, M : S \times \Sigma \rightharpoonup S)$*, where* S *is a set of* states*,* $s_0 \in S$ *is the* initial state*, and* M *is a deterministic partial* transition *function.*

Therefore, a monitor as defined above is nothing but a deterministic state machine without final states. Moreover, the set of states is allowed to be infinite, and the transition function has no complexity requirements (it can even be undecidable). We could have defined monitors to be standard state machines, but the subsequent technical developments would have been slightly more involved.

From safety: *In fact, we aim at shortly giving an even more compact definition of a monitor, that we will call* canonical monitor*, which appears to be sufficient to capture any safety property.*

The intuition for a monitor is the expected one: the monitor is driven by events generated by the observed program (the letters in Σ)—each newly received event drives the monitor from its current state to some other state, as indicated by the transition function M; if the monitor ever gets stuck, that is, if the transition function M is undefined on the current state and the current event, then the monitored property is declared violated at that point by the monitor.

For any partial function $M : S \times \Sigma \rightharpoonup S$, we obey the following common notational convention. If $s \in S$ and $w = w_1 w_2 \dots w_k \in \Sigma^\star$, we write "$M(s,w)\downarrow$" whenever $M(s,w)$ is defined, that is, whenever $M(s,w_1)$ and $M(M(s,w_1),w_2)$ and ... and $M(...(M(s,w_1),w_2)...,w_k)$ are all defined, which is nothing but only saying that $M(...(M(s,w_1),w_2)...,w_k)$ is defined. If we write $M(s,w) = s'$ for some $s' \in S$, then, as expected, we mean that $M(...(M(s,w_1),w_2)...,w_k)$ is defined and equal to s'.

A monitor specifies a finite-trace property, an infinite-trace property, as well as a finite- and infinite-trace property:

Definition 14 *Given a monitor $\mathcal{M} = (S, s_0, M : S \times \Sigma \rightharpoonup S)$, we define the following properties:*

- $\mathcal{L}^\star(\mathcal{M}) = \{w \in \Sigma^\star \mid M(s_0,w)\downarrow\}$,
- $\mathcal{L}^\omega(\mathcal{M}) = \{u \in \Sigma^\omega \mid M(s_0,w)\downarrow \text{ for all } w \in \mathsf{prefixes}(u)\}$, *and*
- $\mathcal{L}^{\star,\omega}(\mathcal{M}) = \mathcal{L}^\star(\mathcal{M}) \cup \mathcal{L}^\omega(\mathcal{M})$.

We call $\mathcal{L}^\star(\mathcal{M})$ the finite-trace property specified by $\mathcal{M}$, *call $\mathcal{L}^\omega(\mathcal{M})$ the* infinite-trace property specified by $\mathcal{M}$, *and call $\mathcal{L}^{\star,\omega}(\mathcal{M})$ the* finite- and infinite-trace property specified by $\mathcal{M}$. *Also, we let*

$$\mathcal{S}_\mathcal{M} = \{s \in S \mid (\exists w \in \Sigma^\star)\, M(s_0,w) = s\}$$

denote the set of reachable states *of $\mathcal{M}$.*

A *monitorable* property is a property which can be specified by a monitor. We next capture this intuitive notion formally:

Definition 15 *For a property $P \subseteq \Sigma^\star \cup \Sigma^\omega$, we let $\mathsf{Monitors}(P)$ be the set of monitors $\{\mathcal{M} \mid \mathcal{L}^{\star,\omega}(\mathcal{M}) = P\}$. If $\mathsf{Monitors}(P) \neq \emptyset$ then P is called* monitorable *and the elements of $\mathsf{Monitors}(P)$ are called* monitors of P. *We define the following classes of properties:*

- $\mathsf{Monitorable}^\star = \{P \subseteq \Sigma^\star \mid P \text{ monitorable}\}$,
- $\mathsf{Monitorable}^\omega = \{P \subseteq \Sigma^\omega \mid P \text{ monitorable}\}$, *and*
- $\mathsf{Monitorable}^{\star,\omega} = \{P \subseteq \Sigma^\star \cup \Sigma^\omega \mid P \text{ monitorable}\}$.

The notion of persistence can also be adapted to monitors:

Definition 16 *A monitor* $\mathcal{M} = (S, s_0, M : S \times \Sigma \rightharpoonup S)$ *is* persistent *iff for any reachable state* $s \in \mathcal{S}_{\mathcal{M}}$*, there is an* $a \in \Sigma$ *such that* $M(s, a)\downarrow$*. Let*

- $\mathsf{PersistentMonitorable}^\star = \{\mathcal{L}^\star(\mathcal{M}) \mid \mathcal{M}$ *persistent*$\}$

be the set of finite-trace properties monitorable by persistent monitors.

Our next goal is to show that each monitor admits a largest persistent "submonitor". To formalize it, we lift the conventional partial order relation on partial functions to monitors:

Definition 17 *If* $\mathcal{M}_1 = (S, s_0, M_1 : S \times \Sigma \rightharpoonup S)$ *and* $\mathcal{M}_2 = (S, s_0, M_2 : S \times \Sigma \rightharpoonup S)$ *are two monitors sharing the same states and initial state, then let* $\mathcal{M}_1 \sqsubseteq \mathcal{M}_2$*, read* $\mathcal{M}_1$ *a* submonitor *of* $\mathcal{M}_2$*, iff for any* $s \in S$ *and any* $a \in \Sigma$*, if* $M_1(s, a)$ *is defined then* $M_2(s, a)$ *is also defined and* $M_2(s, a) = M_1(s, a)$*.*

The above can be easily generalized to allow $\mathcal{M}_1$ to only have a subset of the states of $\mathcal{M}_2$, but we found that generalization unnecessary so far.

The above partial-order on monitors allows us to use conventional mathematics to obtain the largest persistent sub-monitor of a monitor:

Proposition 9 $(\{\mathcal{K} \mid \mathcal{K} \sqsubseteq \mathcal{M}$ *and* $\mathcal{K}$ *persistent*$\}, \sqsubseteq)$ *is a complete (join) semilattice for any monitor* $\mathcal{M}$*.*

Proof: If $\{\mathcal{K}_i = (S, s_0, K_i : S \times \Sigma \rightharpoonup S) \in \mathcal{M}\}_{i \in I}$ is a set of persistent monitors, then their supremum (or join) is the monitor $\mathcal{K} = (S, s_0, K : S \times \Sigma \rightharpoonup S)$ where $K(s, a) = s'$ iff there is some $i \in I$ such that $K_i(s, a) = s'$. It is easy to see that $\mathcal{K}$ is a well-defined monitor and that it is persistent. □

Since complete semilattices have maximum elements, the following definition is fully justified:

Definition 18 *For any monitor* $\mathcal{M} = (S, s_0, M : S \times \Sigma \rightharpoonup S)$*, we let* $\mathcal{M}^\circ = (S, s_0, M^\circ : S \times \Sigma \rightharpoonup S)$ *be the* $\sqsubseteq$*-maximal element of the complete lattice* $(\{\mathcal{K} \mid \mathcal{K} \sqsubseteq \mathcal{M}$ *and* $\mathcal{K}$ *persistent*$\}, \sqsubseteq)$*.*

We next show that, as expected, there is a tight relationship between persistent safety properties (Definition 3) and persistent canonical monitors.

Proposition 10 *Let* $\mathcal{M} = (S, s_0, M : S \times \Sigma \rightharpoonup S)$*. Then the following hold:*

- $\mathcal{L}^\omega(\mathcal{M}) = \mathcal{L}^\omega(\mathcal{M}^\circ)$,

- $\mathcal{L}^\star(\mathcal{M}^\circ) = \mathcal{L}^\star(\mathcal{M})^\circ$, *and*
- $\mathcal{M}$ *persistent iff* $\mathcal{L}^\star(\mathcal{M})$ *persistent.*

Proof: The first property can be shown by the following sequence of equivalences: $u \in \mathcal{L}^\omega(\mathcal{M})$ iff $M(s_0, w) \downarrow$ for all $w \in \mathsf{prefixes}(u)$, iff there is some persistent monitor $\mathcal{K} \sqsubseteq \mathcal{M}$ such as $K(s_0, w) \downarrow$ for all $w \in \mathsf{prefixes}(u)$, iff $M^\circ(s_0, w) \downarrow$ for all $w \in \mathsf{prefixes}(u)$, iff $u \in \mathcal{L}^\omega(\mathcal{M}^\circ)$.

The second property can be shown as follows: $w \in \mathcal{L}^\star(\mathcal{M}^\circ)$ iff $M^\circ(s_0, w) \downarrow$, iff there is some $u \in \mathcal{L}^\omega(\mathcal{M}^\circ)$ such that $w \in \mathsf{prefixes}(u)$ (because $\mathcal{M}^\circ$ is persistent), iff there is some $u \in \mathcal{L}^\omega(\mathcal{M})$ such that $w \in \mathsf{prefixes}(u)$ (by the first property), iff there is some $u \in \Sigma^\omega$ such that $w \in \mathsf{prefixes}(u) \subseteq \mathcal{L}^\star(\mathcal{M})$, iff there is some $u \in \Sigma^\omega$ such that $w \in \mathsf{prefixes}(u) \subseteq \mathcal{L}^\star(\mathcal{M})^\circ$ (because $\mathsf{prefixes}(u)$ is a persistent safety property), iff $w \in \mathcal{L}^\star(\mathcal{M})^\circ$.

Finally, the third property is an immediate consequence of the second, noticing that $\mathcal{M}$ is persistent iff it is equal to $\mathcal{M}^\circ$, and that $\mathcal{L}^\star(\mathcal{M})$ is persistent iff it is equal to $\mathcal{L}^\star(\mathcal{M})^\circ$. □

Theorem 5 *The following hold:*

- $\mathit{Monitorable}^\star = \mathit{Safety}^\star$,
- $\mathit{Monitorable}^\omega = \mathit{Safety}^\omega$,
- $\mathit{Monitorable}^{\star,\omega} = \mathit{Safety}^{\star,\omega}$, *and*
- $\mathit{PersistentMonitorable}^\star = \mathit{PersistentSafety}^\star$.

Proof: First, note that the following hold for any monitor $\mathcal{M}$:

- $\mathcal{L}^\star(\mathcal{M}) \in \mathsf{Safety}^\star$,
- $\mathcal{L}^\omega(\mathcal{M}) \in \mathsf{Safety}^\omega$, and
- $\mathcal{L}^{\star,\omega}(\mathcal{M}) \in \mathsf{Safety}^{\star,\omega}$.

These all follow by Theorem 4: taking P in Definition 12 to be the property $\{w \in \Sigma^\star \mid M(s_0, w) \downarrow\}$, then $\Box P$ over finite traces is precisely $\mathcal{L}^\star(\mathcal{M})$, over infinite traces is precisely $\mathcal{L}^\omega(\mathcal{M})$, and over finite and infinite traces is precisely $\mathcal{L}^{\star,\omega}(\mathcal{M})$, so the three languages are in $\mathsf{Safety}^\star_\Box$, $\mathsf{Safety}^\omega_\Box$, and $\mathsf{Safety}^{\star,\omega}_\Box$, respectively. Therefore, $\mathsf{Monitorable}^\star \subseteq \mathsf{Safety}^\star$, $\mathsf{Monitorable}^\omega \subseteq \mathsf{Safety}^\omega$, and $\mathsf{Monitorable}^{\star,\omega} \subseteq \mathsf{Safety}^{\star,\omega}$.

Second, note that we can associate a default monitor $\mathcal{M}_P$ to any finite-trace property $P \subseteq \Sigma^\star$, namely $(S_P, \epsilon, M_P : S_P \times \Sigma \rightharpoonup S_P)$, where $S_P = \mathsf{prefixes}(P)$, ϵ is the empty word, and $M_P(w, a)$ is defined iff $wa \in \mathsf{prefixes}(P)$, and in that case $M_P(w, a) = wa$. Moreover, it is easy to check that

- $\mathcal{L}^\star(\mathcal{M}_P) = \{w \in \Sigma^\star \mid \mathsf{prefixes}(w) \subseteq P\} = \Box P$ (over finite traces) ,
- $\mathcal{L}^\omega(\mathcal{M}_P) = \{u \in \Sigma^\omega \mid \mathsf{prefixes}(u) \subseteq P\} = \Box P$ (over infinite traces),
- $\mathcal{L}^{\star,\omega}(\mathcal{M}_P) = \{u \in \Sigma^\star \cup \Sigma^\omega \mid \mathsf{prefixes}(w) \subseteq P\} = \Box P$ (over both finite and infinite traces).

Since P was chosen arbitrarily, it follows then by Theorem 4 that $\mathsf{Safety}^\star \subseteq \mathsf{Monitorable}^\star$, $\mathsf{Safety}^\omega \subseteq \mathsf{Monitorable}^\omega$, and $\mathsf{Safety}^{\star,\omega} \subseteq \mathsf{Monitorable}^{\star,\omega}$.

Finally, the equality $\mathsf{PersistentMonitorable}^\star = \mathsf{PersistentSafety}^\star$ follows by the first fact and by Proposition 10. □

4.2 Complexity of Monitoring a Safety Property

We here address the problem of defining the complexity of monitoring. Before we give our definition, let us first discuss some pitfalls in defining this notion. Our definition for the complexity of monitoring resulted as a consequence of trying to avoid these pitfalls. Let P be a safety property.

> **Pitfall 1.**
>
> *The complexity of monitoring P is nothing but the complexity of checking, for an input word $w \in \Sigma^\star$, whether $w \in \mathsf{prefixes}(P)$.*

This would be an easy to formulate decision problem, but, unfortunately, does not capture well the intuition of monitoring, because it does not require that the word w be processed incrementally, as its letters become available from the observed system. Incremental processing of letters can make a huge difference in both how complex monitoring is and how monitoring complexity can be defined. For example, it is well-known that the membership problem of a finite word to the language of an extended regular expression (ERE), i.e., a regular expression extended with complement operators, is a polynomial problem (the classic algorithm by Hopcroft and Ullman [9] runs in space $O(m^2 \cdot n)$ and time $O(m^3 \cdot n)$, where m is the size of the word and n that of the expression). However,

---

From safety: *as shown in Section* **??**

---

there are EREs defining safety properties whose monitoring requires non-elementary space and time. Of course, this non-elementary lower-bound is expressed only as a function of the size of the ERE representing the safety property; it does not take into account the size of the monitored trace. This leads us to our first guiding principle:

Principle 1.

The complexity of monitoring a safety property P should depend only upon P, not upon the trace being monitored.

Indeed, since monitoring is a process that involves potentially unbounded traces, if the complexity of monitoring a property P were expressed as a function of the execution trace as well, then that complexity measure would be close to meaningless in practice, because monitoring reactive systems would have unbounded complexity. For example, consider an operating system monitoring some safety property on how its resources are being used by the various running processes; what one would like to know here is what is the runtime overhead of monitoring that safety property at each relevant event, and not the obvious fact that the more the operating system runs the larger the total runtime overhead is.

Nevertheless, one can admittedly argue that it would still be useful to know how complex the monitoring of P against a given finite trace w is, in terms of both the size of (some representation of) P and the size of w; however, this is nothing but a conventional membership test decision problem, that has nothing to do with monitoring. If one picks some arbitrary off-the-shelf efficient algorithm for membership testing and uses that at each newly received event on the existing finite execution trace, then one may obtain a "monitoring" algorithm whose complexity to process each event grows in time, as events are processed. In the context of monitoring a reactive system, that means that eventually the monitoring process may become unfeasible, regardless of how many resources are initially available and regardless of how efficient the membership testing algorithm is. What one needs in order for the monitoring process to stay feasible regardless of how many events are observed, is a special membership algorithm that processes each event as received and whose state or processing time does not increase potentially unbounded as events are received. Therefore, one needs an algorithm which, if it takes resources R to check w, then it takes at most $R + \Delta$ to check a one-event continuation wa of w, where Δ *does not depend on* w. In other words, one needs a *monitor for* P *of complexity* Δ.

Pitfall 2.

P is typically infinite, so the complexity of monitoring P should be a function of the size of some finite specification, or representation, of P.

Indeed, since Principle 1 tells us that the complexity of monitoring P is a function of P only and not of the monitored trace, one may be tempted to conclude that it is a function of the *size* of some convenient encoding of P. There are at least two problems with this approach, that we discuss below.

- One problem is that the same property P can be specified in many different ways as a structure of finite size; for example, it can be specified as a regular expression, as an extended regular expression, as a temporal logic formula, as an ordinary automaton, as a push-down automaton, etc. These formalisms may represent P as specifications of quite different sizes. Which is the most appropriate? It is, nevertheless, interesting and important to study the complexity of monitoring safety properties expressed using different specification formalisms, as a function of the property representation size, because that can give us an idea of the amount of resources needed to monitor a particular specification. However, one should be aware that such a complexity measure is an attribute of the corresponding specification formalisms, not of the specified property itself. Indeed, the higher this complexity measure for a particular formalism, the higher the encoding strength of safety properties in that formalism: for example, the complexity of monitoring safety properties expressed as EREs is non-elementary in the size of the original ERE, while the complexity of monitoring the same property expressed as an ordinary regular expression is linear in the size of the regular expression. Does that mean that one can monitor safety properties expressed as regular expressions non-elementarily more efficiently than one can monitor safety properties expressed as EREs? Of course not, because EREs and regular expressions have the same expressiveness, so they specify exactly the same safety properties. All it means is that EREs can express safety-properties non-elementarily more compactly than ordinary regular expressions.

- Another problem with this approach is that apparently appropriate representations of P may be significantly larger than it takes to monitor P. One may say, for example, that, whenever possible, a natural way to specify a particular safety property is as a finite-state machine, e.g.,

as a monitor like in Definition 13 . To be more concrete, consider that the safety property P_n saying "every 2^n-th event is a" is specified as a monitor of 2^n states that transits with any event from each state to the next one, except for the 2^n-th state, which has only one transition, with event a, back to state 1. Therefore, the size of this representation of P_n is $\Omega(2^n)$. Assuming that each state takes n bits of storage (for example, assume that states are exactly the binary encodings of the numbers 1, 2, 3, ..., 2^n) and that the next state can be calculated from the current state in linear complexity with the size of the state (which is true in our scenario), then it is clear that the actual complexity of monitoring P_n is $O(n)$. If the complexity of monitoring P_n were a function of the size of the specification of P_n, then one could wrongly conclude that the complexity of monitoring "every 2^n-th event is a" is $O(2^n)$.

Therefore, a safety property P has an inherent complexity w.r.t. monitoring, complexity which has nothing to do with how P is represented, or encoded, or specified. It is that inherent complexity attribute of safety properties that we are after here. From the discussion above, we draw our second guiding principle:

Principle 2.

The monitoring complexity of a safety property P is an attribute of P alone, not a function of the size of some adhoc representation of P.

By Theorem 5, safety properties are precisely those properties that are monitorable, that is, those properties P for which there are (finite-state or not) monitors $\mathcal{M} = (S, s_0, M : S \times \Sigma \rightharpoonup S)$ whose (finite-trace, infinite-trace, or finite- and infinite-trace—this depends upon the type of P) language is precisely P. Any algorithm, program or system that one may come up with to be used as a monitor for P, can be organized as a monitor of the form $\mathcal{M} = (S, s_0, M : S \times \Sigma \rightharpoonup S)$ for P. Consequently, the complexity of monitoring P cannot be smaller than the functional complexity of the partial function $M : S \times \Sigma \rightharpoonup S$) corresponding to some "best" monitor $\mathcal{M}$ for P; if there are no additional restrictions, then by "best" monitor we mean the one whose functional complexity of M is smallest. In particular, if there is no monitor for P whose transition partial function M is decidable, then we can safety say that the problem of monitoring P is undecidable. This discussion leads to the following:

Pitfall 3.

The complexity of monitoring P is the functional complexity of function M, where $\mathcal{M} = (S, s_0, M : S \times \Sigma \rightharpoonup S)$ is the "best" monitor for P.

Since safety properties are precisely the monitorable properties, this appears to be a very natural definition for the complexity of monitoring. While the functional complexity of the monitor function is indeed important because it directly influences the efficiency of monitoring, it is *not* a sufficient measure for the complexity of monitoring. That is because the functional complexity of M only says how complex M is in terms of the size of *its input*; it does not say anything about how large the state of the monitor can grow in time. For example, the rewriting-based monitoring algorithm for EREs from [14], whose states are EREs and whose transition is a derivative operation of functional complexity $O(n^2)$ taking an ERE of size n into an ERE of size $O(n^2)$. It would be very misleading to say that the complexity of monitoring EREs is $O(n^2)$, because it may sound much better than it actually is: the n^2 factor accumulates as events are processed. Any monitor for EREs, including the one based on derivatives, eventually requires non-elementary resources (in the size of the ERE) to process a new event.

Therefore, while the complexity of the function M being executed at each newly received event by a monitor $\mathcal{M}$ is definitely a necessary and important factor to be considered when defining the complexity of monitoring using $\mathcal{M}$, it is not sufficient. One also needs to take into account the size of the input that is being passed to the monitoring function, that is, the size of the monitor state together with the size of the received event. In particular, a monitor storing all the observed trace has unbounded complexity, say ∞, even though its monitoring function has trivial complexity (e.g., the "event storing" function has linear complexity). More generally, if a property admits no finite-state monitor, than we'd like to say that its monitoring complexity is ∞: indeed, for any monitor for such a property and for any amount of resources R, there is some sequence of events that would lead the monitor to a state that needs more than R resources to be stored or computed. These observations lead us to the following:

Principle 3. The complexity of monitoring P is a function of both the functional complexity of M and of the size of the states in S, where $\mathcal{M} = (S, s_0, M : S \times \Sigma \rightharpoonup S)$ is an appropriately chosen ("best") monitor for P.

We next follow the three principles above and derive our definition for the complexity of monitoring a safety property P. Before that, let us first define the complexity of monitoring a safety property using a particular monitor for that property, or in other words, let us first define the complexity of a monitor.

During a monitoring session using a monitor, at any moment in time one needs to store at least one state, namely the state that the monitor is currently in. When receiving a new event, the monitor launches its transition function on the current state and the received input. Therefore, the (worst-case) complexity of monitoring with $\mathcal{M} = (S, s_0, M : S \times \Sigma \rightharpoonup S)$ could be defined as

$$\max\{FC(M(s,a)) \mid s \in S, a \in \Sigma\},$$

where $FC(M(s,a))$ is the functional complexity of evaluating M on state s and event a, as a function of the sizes of s and a. In other words, the worst-case monitoring complexity of a particular monitor is the maximal functional complexity that its transition function has on any state and any input; this functional complexity is expressed as a function of the size of the pair (state,event). In order for such a definition to make sense formally, one would need to define or axiomatize the size of monitor states and the size of events. Since in order to distinguish N elements one needs $\log(N)$ space, we deduce that one needs at least $\log(|S|)$ space to store the state of the monitor in its worst-case monitoring scenario (each state in S is reachable).

Definition 19 *Given a monitor $\mathcal{M} = (S, s_0, M : S \times \Sigma \rightharpoonup S)$, we define the complexity of monitoring $\mathcal{M}$, written $C_{Mon}(\mathcal{M})$, as the function*

$$FC(M)(\log|S|) : \mathbb{N} \to \mathbb{N},$$

which is the "uncurried" version applied on $\log|S|$ of the worst-case functional complexity $FC(M) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ of the partial function M as a function of the size of the pair (state,event) being passed to it.

We assume that the complexity of monitoring a safety property P is the worst-case complexity of monitoring it using some appropriate, "best" monitor for P:

$$\min\{\max\{FC(M(s,a)) \mid s \in S, a \in \Sigma\} \mid \mathcal{M} = (S, s_0, M) \in \mathsf{Monitors}(P)\},$$

> **From safety:** *where $FC(M(s,a))$ is the functional complexity of evaluating M on state s and event a, as a function of the sizes of s and a. In other words, we assume that the complexity of monitoring a safety property P is the worst-case complexity of monitoring it using some appropriate, "best" monitor for P. The worst-case monitoring complexity of a particular monitor is the maximal functional complexity that its transition function has on any state and any input; this functional complexity is expressed as a function of the size of the pair (state,event). Therefore, in order for such a definition to make sense formally, one would need to define or axiomatize the size of monitor states and the size of events.*

This gives us the following:

Definition 20 *We let*

$$C_{Mon}(P) = \min\{FC(M) \circ \langle \log(|S|), 1_\Sigma \rangle \mid \mathcal{M} = (S, s_0, M) \in \mathsf{Monitors}(P)\}$$

be the complexity of monitoring a safety property P.

4.3 Monitoring Safety Properties is Arbitrarily Hard

We show that the problem of monitoring a safety property can be arbitrarily complex. The previous section tells us that there are as many safety properties as real numbers. Therefore, it is not surprising that some of them can be very hard or impossible to monitor. In this section we formalize this intuitive argument. Our approach is to show that we can associate a safety property P_S to any set of natural numbers S, such that monitoring that safety property is as hard as checking membership of arbitrary natural numbers to S. The result then follows from the fact that checking memberships of natural numbers to sets of natural numbers is a problem that can be arbitrarily complex.

Theorem 1 indirectly says that we can associate a persistent safety property to any set of natural numbers (sets of natural numbers are in a bijective correspondence with the real numbers). However, it is not clear how that safety property looks and neither how to monitor it. We next give a more concrete mapping from sets of natural numbers to (persistent) safety properties and show that monitoring the property is equivalent to

testing membership to the set. It suffices to assume that Σ contains only two elements, say $\Sigma = \{0, 1\}$.

Definition 21 *Let $P\_ : \mathcal{P}(\mathbb{N}) \to \mathsf{PersistentSafety}^\star$ be the mapping defined as follows: for any $S \subseteq \mathbb{N}$, let P_S be the set $1^\star \cup \{1^k 0 \mid k \in S\} \cdot \{0, 1\}^\star$.*

It is easy to see that P_S is a persistent safety property over finite traces. Also, it is easy to see that the bijection in the proof of Theorem 2 associates to P_S the safety property over infinite traces $1^\omega \cup \{1^k 0 \mid k \in S\} \cdot \{0, 1\}^\omega$.

Let us now investigate the problem of monitoring P_S.

Proposition 11 *For any $S \subseteq \mathbb{N}$, monitoring P_S is equivalent to deciding membership of natural numbers to S.*

Proof: If M_S is an oracle deciding membership of natural numbers to S, that is, if $M_S(n)$ is true iff $n \in S$, then one can build a monitor for P_S as follows: for a given trace, incrementally read and count the number of prefix 1's; if no 0 is ever observed then monitor indefinitely without reporting any violation; when a first 0 is observed, if any, ask if $M(k)$, where k is the number of 1's observed; if $M(k)$ is false, then report violation; if $M(k)$ is true, then continue monitoring indefinitely and never report violation. It is clear that this is indeed a monitor for P_S.

Conversely, if we had any monitor for P_S then we could build a decision procedure for membership to S as follows: given $k \in \mathbb{N}$, send to the monitor a sequence of k ones followed by a 0; if the monitor reports violation then deduce that $k \notin S$; if the monitor does not report violation, then deduce that $k \in S$. It is clear that this is a decision procedure for membership to S.

The proof works for both persistent safety properties over finite traces and for safety properties over infinite traces. □

The claim in the title of this section follows now from the fact that the set S of natural numbers can be chosen so that its membership problem is arbitrarily complex. For example, since there are as many subsets of natural numbers as real numbers while there are only as many Turing machines as natural numbers, it follows that there are many (exponentially) more sets of natural numbers that are not recognized by Turing machines than those that are. In particular, there are sets of natural numbers corresponding to any degree in the arithmetic hierarchy, i.e., to predicates $A(k)$ of the form $(Q_1 k_1)(Q_2 k_2) \cdots (Q_n k_n)\ R(k, k_1, k_2, \cdots, k_n)$, where Q_1, Q_2, ..., Q_n are alternating (universal or existential) quantifiers and R is a recursive/decidable

relation: for A such a predicate, let S_A be the set of natural numbers $\{k \mid A(k)\}$. Recall that if Q_1 is $\forall$ then A is called a Π_n property, while if Q_1 is $\exists$ then A is called a Σ_n property. In particular, $\Sigma_0 = \Pi_0$ and they contain precisely the recursive/decidable properties, Σ_1 contains precisely the class of recursively enumerable problems, Π_1 contains precisely the co-recursively enumerable problems, etc.; a standard Π_2 problem is TOTALITY: given $k \in \mathbb{N}$, is it true that Turing machine with Gödel number k terminates on all inputs? Since each level in the arithmetic hierarchy contains problems strictly harder than problems on the previous layer (because $\Sigma_n \cup \Pi_n \subsetneq \Sigma_{n+1} \cap \Pi_{n+1}$), the arithmetic hierarchy gives us a universe of safety properties whose monitoring can be arbitrarily hard.

Within the decidable fragment, as expected, monitoring safety properties can also have any complexity. Indeed, pick for example any NP-complete problem and let S be the set of inputs (coded as natural numbers) for which the problem has a positive answer; then, as explained in the proof of Proposition 11, monitoring P_S against input 1^k0 is equivalent to deciding membership of k to S, which is further equivalent to answering the NP-complete problem on input k. Of course, in practice a particular (implementation of a) monitor can be more complex than the corresponding membership problem; for example, monitors corresponding to NP-complete problems are most likely exponential. Also, note that a monitor for P_S needs not necessarily do its complex computation on an input 1^k0 when it encounters the 0. It can perform intermediate computations as it reads the prefix 1's and thus pay a lesser computational price when the 0 is encountered. What Proposition 11 says is that the *total* complexity to process the input 1^k0 can be no lower than the complexity of checking whether $k \in S$.

4.4 Canonical Monitors

We conclude this section with an alternative definition of a monitor, called *canonical monitor*, which is more compact than our previous definition and which appears to be sufficient to capture any safety property. We do not make any use of this alternative definition in this paper, but it may serve as a basis for further foundational endeavors in this area.

The set of states S of a monitor $(S, s_0, M : S \times \Sigma \rightharpoonup S)$ are typically enumerable, so they can be very well replaced with natural numbers. Moreover, the initial state s_0 can be encoded, by convention, as the first natural number, 0. A monitor then becomes nothing but a partial function $\mathbb{N} \times \Sigma \rightharpoonup \mathbb{N}$. We

therefore rightfully call these particular monitors *canonical*:

Definition 22 *A* canonical Σ-monitor *is a partial function* $\mathcal{N} : \mathbb{N} \times \Sigma \rightharpoonup \mathbb{N}$. *Let* $\mathcal{S}_{\mathcal{N}} = \{n \mid (\exists w)\, \mathcal{N}(0, w) = n\}$ *be the* states *of* $\mathcal{N}$. *As before, let*

- $\mathcal{L}^{\star}(\mathcal{N}) = \{w \in \Sigma^{\star} \mid \mathcal{N}(0, w) \downarrow\}$,
- $\mathcal{L}^{\omega}(\mathcal{N}) = \{u \in \Sigma^{\omega} \mid \mathcal{N}(0, w) \downarrow \text{ for all } w \in \mathsf{prefixes}(u)\}$, *and*
- $\mathcal{L}^{\star,\omega}(\mathcal{N}) = \mathcal{L}^{\star}(\mathcal{N}) \cup \mathcal{L}^{\omega}(\mathcal{N})$.

Although the set of states S in a monitor $(S, s_0, M : S \times \Sigma \rightharpoonup S)$ is allowed to have any cardinal while the states in canonical monitors are restricted to natural numbers, it turns out that canonical monitors can in fact express all monitorable properties:

Proposition 12 *A property* $P \subseteq \Sigma^{\star}$ *(resp.* $P \subseteq \Sigma^{\omega}$, *resp.* $P \in \Sigma^{\star} \cup \Sigma^{\omega}$*) is monitorable iff there is some canonical monitor* $\mathcal{N}$ *such that* $P = \mathcal{L}^{\star}(\mathcal{N})$ *(resp.* $P = \mathcal{L}^{\omega}(\mathcal{N})$, *resp.* $P = \mathcal{L}^{\star,\omega}(\mathcal{N})$*).*

Proof: Since any canonical monitor is a monitor, it follows that any property specifiable by a canonical monitor is indeed monitrable. For the converse, let P be a property monitorable by some monitor $\mathcal{M} = (S, s_0, M : S \times \Sigma \rightharpoonup S)$. Since $|\Sigma| \leq \aleph_0$, we can enumerate all the states of $\mathcal{M}$ that can be reached from s_0 with its transition function M. There are many different ways to do this (e.g., in breadth-first order, in depth-first order, etc.), but these are all ultimately irrelevant. If we let $S^r = \{s_0, s_1, s_2, \ldots\}$ denote the resulting set of reachable states, then it is easy to first note that the monitor $\mathcal{M}^r = (S^r, s_0, M : S^r \times \Sigma \rightharpoonup S^r)$ specifies the same property P as $\mathcal{M}$, and second note that $\mathcal{M}^r$ specifies the same property as the canonical monitor $\mathcal{N} : \mathbb{N} \times \Sigma \rightharpoonup \mathbb{N}$ defined by $\mathcal{N}(i, a) = j$ iff $M(s_i, a) = s_j$. □

Exercises

Exercise 7 *Define a canonical monitor for the property*

> *"A file can only be accessed if it is open."*

That is, the file can only be accessed if it was opened at some moment in the past and it was not closed since then. Suppose Σ *consists of the events/actions* $\{o, a, c\}$, *where* o *stands for "file open",* a *for "file access", and* c *for "file close".*

Chapter 5

Event/Trace Observation

Chapter 6

Monitor Synthesis

Chapter 7

Parametric Property Monitoring

Chapter 8

Predictive Runtime Analysis

Chapter 9

Static Analysis to Improve Runtime Verification

Chapter 10

Semantics-Based Runtime Verification

Chapter 11

Conclusion and Future Work

11.1 Safety Properties and Monitoring

Chapters 3 and 4 presented a comprehensive study of safety properties and of their monitoring, using a uniform formalism and notation. Technically, there were two novel contributions. First, it introduced the notion of a *persistent* safety property, which is the finite-trace correspondent of an infinite-trace safety property, and used it to show the cardinal equivalence of the various notions of safety property encountered in the literature. Second, it rigorously defined the problem of monitoring a safety property, and it showed that it can be arbitrarily hard. These results established a firm foundation for studying safety properties and corresponding monitors and algorithms for various domains of interest, where requirements can be expressed using domain-specific formalisms, such as future-time and past-time temporal logics, context-free grammars, push-down automata, and so on.

Bibliography

[1] Martín Abadi and Leslie Lamport. The existence of refinement mappings. In *LICS*, pages 165–175. IEEE Computer Society, 1988.

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

[3] Chris Allan, Pavel Avgustinov, Aske Simon Christensen, Laurie Hendren, Sascha Kuzins, Ondrej Lhoták, Oege de Moor, Damien Sereni, Ganesh Sittampalam, and Julian Tibble. Adding trace matching with free variables to AspectJ. In Richard P. Gabriel, editor, *ACM Conference on Object-Oriented Programming, Systems and Languages (OOPSLA)*, pages 345–364. ACM Press, 2005.

[4] Bowen Alpern and Fred B. Schneider. Defining liveness. *IPL*, 21(4):181–185, 1985.

[5] Krzysztof R. Apt, Nissim Francez, and Shmuel Katz. Appraising fairness in languages for distributed programming. In *POPL*, pages 189–198, 1987.

[6] Feng Chen, Marcelo D'Amorim, and Grigore Roşu. Checking and correcting behaviors of Java programs at runtime with Java-MOP. In *RV'05*, volume 144(4) of *ENTCS*, 2005.

[7] Marcelo d'Amorim and Grigore Roşu. Efficient monitoring of ω-languages. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *LNCS*, pages 364–378, 2005.

[8] Kevin W. Hamlen, J. Gregory Morrisett, and Fred B. Schneider. Computability classes for enforcement mechanisms. *ACM Trans. Program. Lang. Syst.*, 28(1):175–205, 2006.

[9] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[10] O. Kupferman and M. Y. Vardi. Model Checking of Safety Properties. In *Proc. of CAV'99: Conference on Computer-Aided Verification*, Trento, Italy, 1999.

[11] Leslie Lamport. Proving the correctness of multiprocess programs. *IEEE Trans. Software Eng.*, 3(2):125–143, 1977.

[12] Leslie Lamport. Logical foundation. In M. W. Alford, J. P. Ansart, G. Hommel, L. Lamport, B. Liskov, G. P. Mullery, F. B. Schneider, M. Paul, and H. J. Siegert, editors, *Distributed systems: Methods and tools for specification. An advanced course*, volume 190 of *LNCS*, pages 119–130. Springer-Verlag, 1985.

[13] Zohar Manna and Amir Pnueli. *Temporal verification of reactive systems: safety.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[14] G. Roşu and M. Viswanathan. Testing extended regular language membership incrementally by rewriting. In *RTA'03*, volume 2706 of *LNCS*. Springer, 2003.

[15] Grigore Roşu and Klaus Havelund. Rewriting-based techniques for runtime verification. *Automated Software Engineering*, 12(2):151–197, 2005.

[16] Fred B. Schneider. *On Concurrent Programming.* Springer, 1997.

[17] Fred B. Schneider. Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50, 2000.