Unité Mixte de Recherche 5104 CNRS - INPG - UJF
Centre Equation
2, avenue de VIGNATE
F-38610 GIERES
tel : +33 456 52 03 40
fax : +33 456 52 03 50
http://www-verimag.imag.fr

# Extending the Safety-Progress Classification of Properties in a Runtime Verification Context

*Yliès Falcone, Laurent Mounier, Jean-Claude Fernandez, Jean-Luc Richier*

**Verimag Research Report n$^o$ TR-2009-5**

May 22, 2009

# Extending the Safety-Progress Classification of Properties in a Runtime Verification Context

*Yliès Falcone, Laurent Mounier, Jean-Claude Fernandez, Jean-Luc Richier*

May 22, 2009

**Abstract**

This paper revisit and extends results about the Safety-Progress classification of properties introduced by Chang, Manna, and Pnueli [1]. Our work is motivated by runtime verification, as so we believe that this general classification is a good basis for specifying properties. In runtime verification, a major and distinguishing feature is the interest of finite execution sequences and their validation of properties. Indeed, finite execution sequences are often abstract representation of incremental chunks of a program execution. These executions sequences are fed to a monitor, *i.e.* a mechanism designed to state appraisal wrt. a desired property under scrutiny.
We show in this paper, that the four views originally dedicated to infinitary properties can be uniformly extended to finitary ones.

**Keywords:** *r*-property, safety-progress, hierarchy, runtime verification, Streett, DFA, safety, guarantee, response, persistence

**Notes**:

**How to cite this report:**

@techreport { ,
title = { Extending the Safety-Progress Classification of Properties in a Runtime Verification Context},
authors = { Yliès Falcone, Laurent Mounier, Jean-Claude Fernandez, Jean-Luc Richier},
institution = { Verimag Research Report },
number = {TR-2009-5},
year = { 2009},
note = { }
}

# Contents



# 1 Introduction

This paper extends results about the safety-progress [1, 2, 3] classification of properties. In the original papers this classification introduced a hierarchy between properties defined as *infinite* execution sequences. We extend the classification to deal with finite-length execution sequences. As so we revisit this classification for *r*-properties.

**The safety-progress classification: initial version** The safety-progress classification is an alternative to the more classical safety-liveness [4, 5] dichotomy. Unlike this later, the safety-progress classification is a hierarchy and not a partition. It provides a finer-grain classification, and the properties of each class are characterized according to four *views* [1]: a language-theoretic view, a topological view, a temporal logic view, and an automata-based view. The language-theoretic view describes the hierarchy according to the way each class can be constructed from sets of finite sequences. The topological view characterizes the classes as sets with topological properties. The third vision links the classes to their expression in temporal logic. At last, the automata-view gives syntactic characterization on the automata recognizing the properties of a given class.

Later [6], this hierarchy was characterized in terms of Büchi, co-Büchi, and Streett acceptance conditions. A second characterization consisted in an Until-Release hierarchy: a logical (LTL) view in which properties are classified according to the alternation depth of the until and release operators.

**Contribution** In this paper we revisit the safety-progress classification in a runtime verification fashion. The usual words are here execution sequences produced by an underlying program. And the properties are used for specification. In order to deal with finite sequences, we will introduce *r*-properties able to describe finitary and infinitary properties. Furthermore, we will consider here only the language-theoretic and the automata views dedicated to *r*-properties.

The contributions of this report are as follows:

- Language-theoretic view: we give a formal definitions of all operators of the language-theoretic view introduced in [2]. Moreover, we introduce two operators for response and persistence finite sequences. Those definitions are consistent with their infinitary corresponding ones.
- Automata view: we introduce a finite-sequence acceptance criterion for Streett automata. This criterion is compatible with operators producing finite sequences in the language-theoretic view. Moreover, we introduce a transformation over Deterministic Finite-state Automata [7] so as to produce Streett automata. Similarly to the operators in the language-theoretic view those transformations are specific to each class of properties.

**Paper Organization** The remainder of this report is organized as follows. The Sect. 2 introduces some preliminaries and notations. In Sect. 3 we introduce the hierarchy informally. The language-theoretic view is studied in Sect. 4. In Sect. 5 we deal with the automata view of the hierarchy. Some concluding remarks are given in Sect. 6.

# 2 Preliminaries and notations

This section introduces some preliminary notations, namely the notions of *program execution sequences* and *program properties*.

## 2.1 Sequences and execution sequences

**Sequences and execution sequences.** Considering a finite set of elements $E$, we define notations about sequences of elements belonging to $E$. A sequence $\sigma$ containing elements of $E$ is formally defined by a total function $\sigma : I \to E$ where $I$ is either the interval $[0, n-1]$ for some $n \in \mathbb{N}$, or $\mathbb{N}$ itself (the set of natural numbers). We denote by $E^*$ the set of finite sequences over $E$ (partial function from $\mathbb{N}$), by $E^+$ the set of non-empty finite sequences over $E$, and by $E^\omega$ the set of infinite sequences over $E$. The set $E^\infty = E^* \cup E^\omega$ is the set of all sequences over $E$. The empty sequence of $E$ is denoted by $\epsilon_E$ or $\epsilon$ when clear from context. The length (number of elements) of a finite sequence $\sigma$ is noted $|\sigma|$ and the $(i+1)$-th element of $\sigma$ is denoted by $\sigma_i$. For two sequences $\sigma \in E^*, \sigma' \in E^\infty$, we denote by $\sigma \cdot \sigma'$ the concatenation of $\sigma$ and $\sigma'$, and by $\sigma \prec \sigma'$ the fact that $\sigma$ is a strict prefix of $\sigma'$ (resp. $\sigma'$ is a strict suffix of $\sigma$). The sequence $\sigma$ is said to be a strict prefix of $\sigma' \in \Sigma^\infty$ when $\forall i \in \{0, \ldots, |\sigma| - 1\} \cdot \sigma_i = \sigma'_i$ and $|\sigma| < |\sigma'|$. When $\sigma' \in E^*$, we note $\sigma \preceq \sigma' \stackrel{\text{def}}{=} \sigma \prec \sigma' \vee \sigma = \sigma'$. For $\sigma \in E^\infty$ and $n \in \mathbb{N}$, $\sigma_{\ldots n}$ is the sub-sequence containing the $n+1$ first elements of $\sigma$. Also, when $|\sigma| > n$, the subsequence $\sigma_{n\ldots}$ is the sequence containing all elements of $\sigma$ but the $n$ first ones. For $i, j \in \mathbb{N}$ with $i \leq j$, we denote by $\sigma_{i\ldots j}$ the subsequence of $\sigma$ containing the $(i+1)$-th to the $(j+1)$-th (included) elements.

A program $\mathcal{P}$ is considered as a generator of execution sequences. We are interested in a restricted set of operations the program can perform. These operations influence the truth value of properties the program is supposed to fulfill. Such execution sequences can be made of access events on a secure system to its ressources, or kernel operations on an operating system. In a software context, these events may be abstractions of relevant instructions such as variable modifications or procedure calls. We abstract these operations by a finite set of *events*, namely a vocabulary $\Sigma$. We denote by $\mathcal{P}_\Sigma$ a program for which the vocabulary is $\Sigma$. The set of execution sequences of $\mathcal{P}_\Sigma$ is denoted by $Exec(\mathcal{P}_\Sigma) \subseteq \Sigma^\infty$. This set is *prefix-closed*, that is $\forall \sigma \in Exec(\mathcal{P}_\Sigma), \sigma' \in \Sigma^* \cdot \sigma' \preceq \sigma \Rightarrow \sigma' \in Exec(\mathcal{P}_\Sigma)$. In the remainder of this article, we consider a vocabulary $\Sigma$.

## 2.2 Properties

**Properties as sets of execution sequences.** A *finitary property* (resp. an *infinitary property*, a *property*) is a subset of execution sequences of $\Sigma^*$ (resp. $\Sigma^\omega$, $\Sigma^\infty$). Considering a given finite (resp. infinite, finite or infinite) execution sequence $\sigma$ and a property $\phi$ (resp. $\varphi$, $\theta$), when $\sigma \in \phi$, noted $\phi(\sigma)$ (resp. $\sigma \in \varphi$, noted $\varphi(\sigma)$, $\sigma \in \theta$, noted $\theta(\sigma)$), we say that $\sigma$ *satisfies* $\phi$ (resp. $\varphi$, $\theta$). A consequence of this definition is that properties we will consider are restricted to *single* execution sequences, excluding specific properties defined on powersets of execution sequences (like fairness, for instance). Moreover, for a finitary property $\phi$ and an execution sequence $\sigma \in \Sigma^\infty$, we denote by $\mathrm{Pref}_\prec(\phi, \sigma)$ the set of all (strict) prefixes of $\sigma$ satisfying $\phi$, *i.e.* $\mathrm{Pref}_\prec(\phi, \sigma) = \{\sigma' \in \phi \mid \sigma' \prec \sigma\}$. The longest prefix of $\sigma$ satisfying $\phi$ (noted $\mathrm{Max}(\mathrm{Pref}_\prec(\phi, \sigma))$) is the maximal element regarding $\prec$ if $\mathrm{Pref}_\prec(\phi, \sigma) \neq \emptyset$. Given a property $\phi \subseteq \Sigma^*$ and an execution sequence $\sigma \in \Sigma^*$, a straightforward property of the set $Pref_\prec(\phi, \sigma)$ is that $\forall a \in \Sigma, \forall \sigma' \in \Sigma^* \big(\sigma' \prec \sigma \wedge \sigma' = Max(Pref_\prec(\phi, \sigma)) \wedge \neg\phi(\sigma) \Rightarrow \sigma' = Max(Pref_\prec(\phi, \sigma \cdot a))\big)$ $\forall a \in \Sigma, \neg\phi(\sigma) \Rightarrow Max(Pref_\prec(\phi, \sigma \cdot a)) = Max(Pref_\prec(\phi, \sigma))$.

Runtime verificationa mechanisms run with the underlying program under scrutiny, it should be able to decide about the truth value of a property regarding the current produced execution sequence. The principle of analyzing execution sequence at runtime restricts the kind of property our monitors can analyze. Indeed,

the analysis of runtime execution monitor is, by definition, restricted to one execution sequence. Such a fact implies that the security automaton cannot decide about property involving several executions of the monitored program. An example of such a kind of property is for instance a fairness property.

**Runtime properties.** In this paper we are interested in runtime properties. As stated in the introduction, we consider finite and infinite execution sequences (that a program may produce), runtime verification properties should characterize satisfaction for both kinds of sequence in a uniform way. As so, We introduce *r*-properties (runtime properties) as pairs $(\phi, \varphi) \subseteq \Sigma^* \times \Sigma^\omega$. Intuitively, the finitary property $\phi$ represents the desirable property that finite execution sequences should fulfill, whereas the infinitary property $\varphi$ is the expected property for infinite execution sequences. The definition of negation of a *r*-property follows from definition of negation for finitary and infinitary properties. For a *r*-property $(\phi, \varphi)$, we define $\overline{(\phi, \varphi)}$ as $(\overline{\phi}, \overline{\varphi})$. Boolean combinations of *r*-properties are defined in a natural way. For $* \in \{\cup, \cap\}$, $(\phi_1, \varphi_1) * (\phi_2, \varphi_2) = (\phi_1 * \phi_2, \varphi_1 * \varphi_2)$. Considering an execution sequence $\sigma \in Exec(\mathcal{P}_\Sigma)$, we say that $\sigma$ satisfies $(\phi, \varphi)$ when $\sigma \in \Sigma^* \wedge \phi(\sigma) \vee \sigma \in \Sigma^\omega \wedge \varphi(\sigma)$. For a *r*-property $\Pi = (\phi, \varphi)$, we note $\Pi(\sigma)$ when $\sigma$ satisfies $(\phi, \varphi)$.

# 3 Informal description

The safety-progress classification is made of four basic classes over execution sequences. Informally, the classes were defined as follows:

- *safety* properties are the properties for which whenever a sequence satisfies a property, *all its prefixes* satisfy this property.
- *guarantee* properties are the properties for which whenever a sequence satisfies a property, *there are some prefixes* (at least one) satisfying this property.
- *response* properties are the properties for which whenever a sequence satisfies a property, *an infinite number of its prefixes* satisfy this property.
- *persistence* properties are the properties for which whenever a sequence satisfies a property, *all its prefixes* continuously satisfy this property from a certain point.

Furthermore, two extra classes can be defined as (finite) boolean combinations (union and intersection) of basic classes.

- The *obligation class* can be defined as the class obtained by boolean combination of safety and guarantee properties.
- The *reactivity class* can be defined as the class obtained by boolean combination of response and persistence properties. This is the more general class containing all linear temporal properties [1].

The following example introduces informally the aforementioned properties. In Example 4.2, we formalize those properties into *r*-properties.

EXAMPLE 3.1 *Let consider an operating system where a given operation* op *is allowed only when an authorization* auth *has been granted before. The operator is also endowed with three pimitives related to authentication:* r_auth *(requesting authentication),* g_auth *(granting authentication),* d_auth *(denying authentication). Then,*

- *the property* $\Pi_1$ *stating that "each occurence of* op *should be preceded by a distinct occurence of* g_auth*" is a* safety *property;*
- *the property* $\Pi_2$ *stating that "In this session, the user should perform an authorization request* r_auth *which should be eventually followed by a grant (*g_auth*) or a deny (*d_auth*)" is a* guarantee *property;*
- *the property* $\Pi_3$ *stating that "the system should run forever, unless a* d_auth *is issued and then the user should be disconnected (*disco*) and the system should terminate (*end*)" is an* obligation *property;*

- *the property* $\Pi_4$ *stating that "each occurrence of* r_auth *should be first written in a log file and then answered either with a* g_auth *or a* d_auth *without any occurrence of* op *in the meantime" is a* response *property;*
- *the property* $\Pi_5$ *stating that "after a* d_auth*, a (forbidden) use of operation* op *should imply that at some point any future call to* r_auth *will always result in a* d_auth *answer" is a* persistence *property.*

# 4 The language-theoretic view of *r*-properties

In the language-theoretic view of the hierarchy, *r*-properties are pairs of sets of execution sequence.

## 4.1 Building finitary and infinitary properties from finitary ones

The language-theoretic view of the safety-progress classification is based on the construction of infinitary-properties and finitary-properties from finitary ones. It relies on the use of four operators $A, E, R, P$ (building infinitary properties) and four operators $A_f, E_f, R_f, P_f$ (building finitary properties) applying to finitary properties. In the original classification of Mana and Pnueli, operators $A, E, R, P, A_f, E_f$ were introduced. In this paper, we add operators $R_f$ and $P_f$ and give a formal definition of all operators.

Let $\psi$ a finitary property over $\Sigma$.

- $A(\psi)$ consists of all infinite words $\sigma$ such that *all* prefixes of $\sigma$ belong to $\psi$.
  Formally $A(\psi) = \{\sigma \in \Sigma^\omega \mid \forall \sigma' \in \Sigma^*, \sigma' \prec \sigma \Rightarrow \psi(\sigma')\}$.
- $E(\psi)$ consists of all infinite words $\sigma$ such that *some* prefixes of $\sigma$ belong to $\psi$.
  Formally $E(\psi) = \{\sigma \in \Sigma^\omega \mid \exists \sigma' \in \Sigma^*, \sigma' \prec \sigma \wedge \psi(\sigma')\}$.
- $R(\psi)$ consists of all infinite words $\sigma$ such that *infinitely many* prefixes of $\sigma$ belong to $\psi$.
  Formally $R(\psi) = \{\sigma \in \Sigma^\omega \mid \forall \sigma' \in \Sigma^*, \exists \sigma'' \in \Sigma^*, \sigma' \prec \sigma'' \prec \sigma \wedge \psi(\sigma'')\}$.
- $P(\psi)$ consists of all infinite words $\sigma$ such that *all but finitely many* prefixes of $\sigma$ belong to $\psi$.
  Formally $P(\psi) = \{\sigma \in \Sigma^\omega \mid \exists \sigma' \in \Sigma^*, \forall \sigma'' \in \Sigma^*, \sigma' \prec \sigma'' \prec \sigma \Rightarrow \psi(\sigma'')\}$.

Operators $A_f, E_f, R_f, P_f$ build finitary properties from finitary ones.

- $A_f(\psi)$ consists of all finite words $\sigma$ such that *all* prefixes of $\sigma$ belong to $\psi$.
  Formally $A_f(\psi) = \{\sigma \in \Sigma^* \mid \forall \sigma' \in \Sigma^*, \sigma' \preceq \sigma \Rightarrow \psi(\sigma')\}$. One can observe that $A_f(\psi) = \psi$ if $\psi$ is prefix-closed and $\emptyset$ else.
- $E_f(\psi)$ consists of all finite words $\sigma$ such that *some* prefixes of $\sigma$ belong to $\psi$.
  Formally $E_f(\psi) = \{\sigma \in \Sigma^* \mid \exists \sigma' \in \Sigma^*, \sigma' \preceq \sigma \wedge \psi(\sigma')\}$. One can observe that $E_f(\psi) = \psi \cdot \Sigma^*$.
- $R_f(\psi)$ consists of all finite words $\sigma$ such that $\psi(\sigma)$ and there exists a continuation $\sigma'$ of $\sigma$ also belonging to $\psi$.
  Formally $R_f(\psi) = \{\sigma \in \Sigma^* \mid \psi(\sigma) \wedge \exists \sigma' \in \Sigma^* \cdot \sigma \prec \sigma' \wedge \psi(\sigma')\}$.
- $P_f(\psi)$ consists of all finite words $\sigma$ belonging to $\psi$ s.t. there exists an extension $\sigma'$ of $\sigma$ s.t. each extension $\sigma''$ of $\sigma'$ belongs to $\psi$.
  Formally $P_f(\psi) = \{\sigma \in \Sigma^* \mid \psi(\sigma) \wedge \exists \sigma' \in \Sigma^* \cdot \sigma \preceq \sigma' \wedge \forall \sigma'', \sigma' \preceq \sigma'' \Rightarrow \psi(\sigma'')\}$.

Based on these operators, each class can be seen from the language-theoretic view.

DEFINITION 4.1 *A r-property* $\Pi = (\phi, \varphi)$ *is defined to be*

- *A* safety *r-property if* $\Pi = (A_f(\psi), A(\psi))$ *for some finitary property* $(\psi)$*. That is, all prefixes of a finite word* $\sigma \in \phi$ *or of an infinite word* $\sigma \in \varphi$ *belong to* $\psi$*.*

- *A* guarantee *r-property if* $\Pi = (E_f(\psi), E(\psi))$ *for some finitary property* $\psi$*. That is, each finite word* $\sigma \in \phi$ *or infinite word* $\sigma \in \varphi$ *is guaranteed to have some prefixes (at least one) belonging to* $\psi$*.*

- *A* response *r-property if* $\Pi = (R_f(\psi), R(\psi))$ *for some finitary property* $\psi$*. That is, each infinite word* $\sigma \in \varphi$ *recurrently has (infinitely many) prefixes belonging to* $\psi$*.*

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

*In all cases, we say that* $\Pi$ *is built over* $\psi$*. Furthermore, obligation (resp. reactivity) r-properties are obtained by boolean combinations of safety and guarantee (resp. response and persistence) r-properties.*

Given a set of events $\Sigma$, we note $\mathrm{Safety}(\Sigma)$ (resp. $\mathrm{Guarantee}(\Sigma)$, $\mathrm{Obligation}(\Sigma)$, $\mathrm{Response}(\Sigma)$, $\mathrm{Persistence}(\Sigma)$) the set of safety (resp. guarantee, obligation, response, persistence) *r*-properties defined over $\Sigma$.

We expose some straightforward consequences of definitions of safety and guarantee *r*-properties.

PROPERTY 4.1 (CLOSURE OF *r*-PROPERTIES) *Considering an r-property* $\Pi = (\phi, \varphi)$ *defined over an alphabet* $\Sigma$ *built from a finitary property* $\psi$*, the following facts hold:*

- *If* $\Pi$ *is a safety r-property, all prefixes of a sequence belonging to* $\Pi$ *also belong to* $\Pi$*. That is,* $\forall \sigma \in \Sigma^\infty, \Pi(\sigma) \Rightarrow \forall \sigma' \prec \sigma, \Pi(\sigma')$.

  *Indeed, we have either* $\phi(\sigma)$ *or* $\varphi(\sigma)$*, i.e. all prefixes* $\sigma'$ *of* $\sigma$ *belong to* $\psi$*. Necessarily, all prefixes* $\sigma''$ *of* $\sigma'$ *also belong to* $\psi$*, that is* $\psi(\sigma'')$*. By definition, that means* $\sigma' \in A_f(\psi)$*, i.e.* $\phi(\sigma')$ *and* $\Pi(\sigma')$*.*

- *If* $\Pi$ *is a guarantee r-property, all continuations of a finite sequence belonging to* $\Pi$ *also belong to* $\Pi$*. That is,* $\forall \sigma \in \Sigma^*, \Pi(\sigma) \Rightarrow \forall \sigma' \in \Sigma^\infty, \Pi(\sigma \cdot \sigma')$.

  *Indeed,* $\Pi(\sigma)$ *implies that* $\sigma$ *has at least one prefix* $\sigma_0 \preceq \sigma$ *belonging to* $\psi$*:* $\sigma \in E_f(\psi)$*. Then, any continuation of* $\sigma$ *built using any finite or infinite sequence* $\sigma'$ *has at least the same prefix belonging to* $\psi$*. If* $\sigma' \in \Sigma^*$*, we have* $\sigma_0 \preceq \sigma \preceq \sigma \cdot \sigma'$ *and* $\sigma \cdot \sigma' \in E_f(\psi)$*. If* $\sigma' \in \Sigma^\omega$*, we have* $\sigma_0 \preceq \sigma \prec \sigma \cdot \sigma'$ *and* $\sigma \cdot \sigma' \in E(\psi)$*.*

We illustrate in the following example the construction of infinitary properties from finitary ones for each of the four operators (finite and infinite).

EXAMPLE 4.1 (CONSTRUCTION OF INFINITARY AND FINITARY PROPERTIES FROM FINITARY ONES) *We use regular-expressions to define properties.*

- *For the finitary property* $\psi = \epsilon + a^+ \cdot b^*$*,* $A_f(\psi) = \epsilon + a^+ \cdot b^*$*,* $A(\psi) = a^\omega + a^+ \cdot b^\omega$*,* $(A_f(\psi), A(\psi))$ *is a safety r-property. This language contains all the words that have either only occurrences of* $a$ *or a finite number of occurrences of* $a$ *(at least one) followed only by occurrences of* $b$*.*

- *For the finitary property* $\psi = a^+ \cdot b^*$*,* $E_f(\psi) = a^+ \cdot b^* \cdot \Sigma^*$*,* $E(\psi) = a^+ \cdot b^* \cdot \Sigma^\omega$*,* $(E_f(\psi), E(\psi))$ *is a guarantee r-property.*

- *For the finitary property* $\psi = \Sigma^* \cdot b$*,* $R_f(\psi) = (\Sigma^* \cdot b)^+$*,* $R(\psi) = (\Sigma^* \cdot b)^\omega$*,* $(R_f(\psi), R(\psi))$ *is a response r-property. This language contains all the words that have infinitely many occurrences of* $b$*.*

- *For the finitary property* $\psi = \Sigma^* \cdot b$*,* $P_f(\psi) = \Sigma^* \cdot b^+$*,* $P(\psi) = \Sigma^* \cdot b^\omega$*,* $(P_f(\psi), P(\psi))$ *is a persistence r-property. This language contains all the words that, from a certain point on, contain only occurrences of* $b$*.*

EXAMPLE 4.2 (*r*-PROPERTIES) *Properties of Example 3.1 can be formalized as r-properties as follows.*

- *the property* $\Pi_1$ *can be expressed as a* safety *r-property built over* $\psi_1 = (g\_auth^+ \cdot op)^* \cdot g\_auth^*$ *with* $\Sigma = \{g\_auth, op\}$

- *the property* $\Pi_2$ *can be expressed as a* guarantee *r-property built over* $\psi_2 = (\Sigma \setminus \{r\_auth\})^* \cdot r\_auth \cdot (\Sigma \setminus \{g\_auth, d\_auth\})^* \cdot (g\_auth + d\_auth)$ *with* $\Sigma = \{g\_auth, d\_auth, r\_auth\}$
- *the property* $\Pi_3$ *can be expressed as an* obligation *r-property built over* $\psi_3 = (\Sigma \setminus \{d\_auth, end\})^*$, *and* $\psi'_3 = (\Sigma \setminus \{d\_auth, end\})^* \cdot d\_auth \cdot (\Sigma \setminus \{disco\})^* \cdot disco$ *with* $\Sigma = \{end, d\_auth, disco\}$; *then* $\Pi_3$ *is* $(A_f(\psi_3), A(\psi_3)) \vee (E_f(\psi'_3), E(\psi'_3))$.
- *the property* $\Pi_4$ *can be expressed as a* response *r-property built over* $\psi_4 = r\_auth \cdot log^+ \cdot (d\_auth + g\_auth)$ *with* $\Sigma = \{g\_auth, d\_auth, op, log\}$
- *the property* $\Pi_5$ *can be expressed as a* persistence *r-property built over* $\psi_5 = (d\_auth \cdot g\_auth)^* . d\_auth \cdot (\Sigma \setminus \{r\_auth, d\_auth\})^* \cdot op \cdot \big(g\_auth^* + (r\_auth \cdot (\Sigma \setminus \{d\_auth\})^+)\big)^*$ *with* $\Sigma = \{g\_auth, d\_auth, op\}$

### 4.2 About *r*-properties

The following lemma (inspired from [1]) provides a decomposition of each obligation *r*-properties in a normal form.

LEMMA 4.1 *Any obligation r-property can be represented as the intersection*

$$\bigcap_{i=1}^{n} (\text{Safety}_i \cup \text{Guarantee}_i)$$

*for some* $n > 0$, *where* $\text{Safety}_i$ *and* $\text{Guarantee}_i$ *are respectively safety and guarantee r-properties. We refer to this presentation as the conjunctive normal form of obligation r-properties.*

When an *r*-property $\Pi$ is expressed as $\cap_{i=1}^{k}(\text{Safety}_i \cup \text{Guarantee}_i)$, $\Pi$ is said to be a k-obligation *r*-property. The set of $k$-obligation *r*-properties ($k \geq 1$) is denoted $Obligation_k$. Similar definitions and properties hold for reactivity *r*-properties which are expressed by combination of response and persistence *r*-properties.

## 5 The automata view of *r*-properties

For each class of the safety-progress classification it is possible to syntactically characterize a recognizing automaton. We define a variant of deterministic and complete Streett automata (introduced in [8] and used in [1]) for property recognition. These automata process events and decide properties of interest. We add to original Streett automata a finite-sequence recognizing criterion in such a way that these automata uniformly recognize *r*-properties.

DEFINITION 5.1 (STREETT $m$-AUTOMATON) *A deterministic Streett* $m$*-automaton is a tuple* $(Q, q_{\text{init}}, \Sigma, \longrightarrow, \{(R_1, P_1), \ldots, (R_m, P_m)\})$ *defined relatively to a set of events* $\Sigma$. *The set* $Q$ *is the set of automaton states,* $q_{\text{init}} \in Q$ *is the initial state. The function* $\longrightarrow$: $Q \times \Sigma \to Q$ *is the transition function. In the following, for* $q, q' \in Q, e \in \Sigma$ *we abbreviate* $\longrightarrow (q, e) = q'$ *by* $q \xrightarrow{e} q'$. *The set* $\{(R_1, P_1), \ldots, (R_m, P_m)\}$ *is the set of accepting pairs, for all* $i \leq m$, $R_i \subseteq Q$ *are the sets of recurrent states, and* $P_i \subseteq Q$ *are the sets of persistent states.*

We refer to an automaton with $m$ accepting pairs as a $m$-automaton. When $m = 1$, a 1-automaton is also called a *plain*-automaton, and we refer to $R_1$ and $P_1$ as $R$ and $P$. In the following $\mathcal{A} = (Q^{\mathcal{A}}, {q_{\text{init}}}^{\mathcal{A}}, \Sigma, \longrightarrow_{\mathcal{A}}, \{(R_1, P_1), \ldots, (R_m, P_m)\})$ designates a Streett $m$-automaton.

For $\sigma \in \Sigma^{\infty}$, the *run* of $\sigma$ on $\mathcal{A}$ is the sequence of states involved by the execution of $\sigma$ on $\mathcal{A}$. It is formally defined as $run(\sigma, \mathcal{A}) = q_0 \cdot q_1 \cdots$ where $\forall i \cdot (q_i \in Q^{\mathcal{A}} \wedge q_i \xrightarrow{\sigma_i}_{\mathcal{A}} q_{i+1}) \wedge q_0 = {q_{\text{init}}}^{\mathcal{A}}$. The *trace* resulting in the execution of $\sigma$ on $\mathcal{A}$ is the unique sequence (finite or not) of tuples $(q_0, \sigma_0, q_1) \cdot (q_1, \sigma_1, q_2) \cdots$ where $run(\sigma, \mathcal{A}) = q_0 \cdot q_1 \cdots$.

Also we consider the notion of infinite visitation of an execution sequence $\sigma \in \Sigma^{\omega}$ on a Streett automaton $\mathcal{A}$, denoted $vinf(\sigma, \mathcal{A})$, as the set of states appearing infinitely often in $run(\sigma, \mathcal{A})$. It is formally defined as follows: $vinf(\sigma, \mathcal{A}) = \{q \in Q^{\mathcal{A}} \mid \forall n \in \mathbb{N}, \exists m \in \mathbb{N} \cdot m > n \wedge q = q_m$ with $run(\sigma, \mathcal{A}) = q_0 \cdot q_1 \cdots\}$.

For a Streett automaton, the notion of acceptance condition is defined using the accepting pairs.

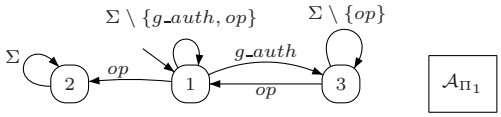


Figure 1: Recognizing automaton for the safety *r*-property $\Pi_1$

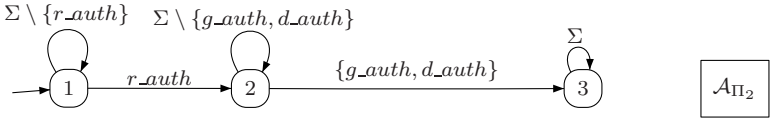


Figure 2: A guarantee-automaton for the guarantee *r*-property $\Pi_2$

DEFINITION 5.2 (ACCEPTANCE CONDITION (INFINITE SEQUENCES)) *For $\sigma \in \Sigma^\omega$, we say that $\mathcal{A}$ accepts $\sigma$ if $\forall i \in \{1,\ldots,m\} \cdot inf(\sigma,\mathcal{A}) \cap R_i \neq \emptyset \vee inf(\sigma,\mathcal{A}) \subseteq P_i$.*

To deal with *r*-properties we need to define also an acceptance criterion for *finite* sequences.

DEFINITION 5.3 (ACCEPTANCE CONDITION (FINITE SEQUENCES)) *For a finite-length execution sequence $\sigma \in \Sigma^*$ such that $|\sigma| = n$, we say that the $m$-automaton $\mathcal{A}$ accepts $\sigma$ if $(\exists q_0,\ldots,q_n \in Q^{\mathcal{A}} \cdot run(\sigma,\mathcal{A}) = q_0 \cdots q_n \wedge q_0 = {q_{\text{init}}}^{\mathcal{A}}$ and $\forall i \in \{1,\ldots,m\} \cdot q_n \in P_i \cup R_i)$.*

**The hierarchy of automata.** By setting syntactic restrictions on a Streett automaton, we modify the kind of properties recognized by such an automaton.

- A *safety automaton* is a plain automaton such that $R = \emptyset$ and there is no transition from a state $q \in \overline{P}$ to a state $q' \in P$.
- A *guarantee automaton* is a plain automaton such that $P = \emptyset$ and there is no transition from a state $q \in R$ to a state $q' \in \overline{R}$.
- An $m$-*obligation automaton* is an $m$-automaton such that for each $i$ in $\{1,\ldots,m\}$:
  - there is no transition from $q \in \overline{P_i}$ to $q' \in P_i$,
  - there is no transition from $q \in R_i$ to $q' \in \overline{R_i}$,
- A *response automaton* is a plain automaton such that $P = \emptyset$,
- A *persistence automaton* is a plain automaton such that $R = \emptyset$,
- A *reactivity automaton* is any unrestricted automaton.

**Automata and properties.** We say that a Streett automaton $\mathcal{A}_\Pi$ *defines* a *r*-property $(\phi,\varphi) \in \Sigma^* \times \Sigma^\omega$ if and only if the set of finite (resp. infinite) execution sequences accepted by $\mathcal{A}_\Pi$ is equal to $\phi$ (resp. $\varphi$). Conversely, a property $(\phi,\varphi) \in \Sigma^* \times \Sigma^\omega$ is said to be *specifiable* by an automaton $\mathcal{A}_\Pi$ if the set of finite (resp. infinite) execution sequences accepted by the automaton $\mathcal{A}_\Pi$ is $\phi$ (resp. $\varphi$).

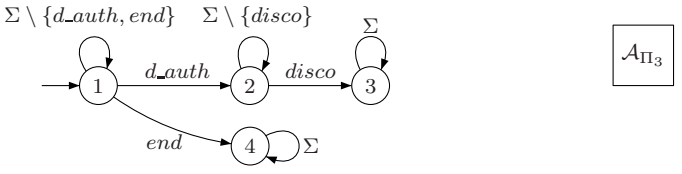


Figure 3: A 1-obligation-automaton for the obligation *r*-property $\Pi_3$

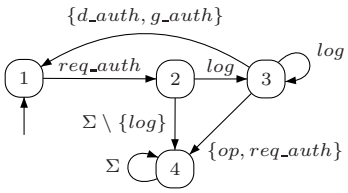


Figure 4: A response-automaton for the response *r*-property $\Pi_4$

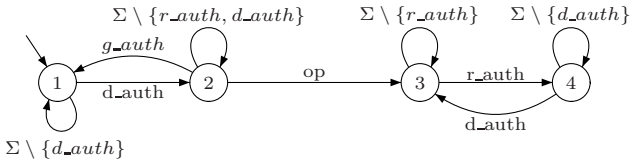


Figure 5: A persistence-automaton for the persistence *r*-property $\Pi_5$

EXAMPLE 5.1 (SPECIFYING *r*-PROPERTIES BY STREETT AUTOMATA) *The r-properties previously introduced in example 3.1 can be specified by Streett automata.*

- *Property* $\Pi_1$ *is specified by automaton* $\mathcal{A}_{\Pi_1}$ *depicted on Fig. 1. Its set of states is* $\{1,2,3\}$*, the initial state is* $1$*, and we have* $R = \emptyset$ *and* $P = \{1,3\}$.
- *Property* $\Pi_2$ *is specified by automaton* $\mathcal{A}_{\Pi_2}$ *depicted on Fig. 2 (up side). Its set of states is* $\{1,2,3\}$*, the initial state is* $1$*, and we have* $P = \emptyset$ *and* $R = \{3\}$.
- *Property* $\Pi_3$ *is specified by automaton* $\mathcal{A}_{\Pi_3}$ *depicted on Fig. 3. Its set of states is* $\{1,2,3,4\}$*, the initial state is* $1$*, and we have* $P = \{1\}$ *and* $R = \{3\}$.
- *Property* $\Pi_4$ *is specified by automaton* $\mathcal{A}_{\Pi_4}$ *depicted on Fig. 4. Its set of states is* $\{1,2,3,4\}$*, the initial state is* $1$*, and we have* $P = \emptyset$ *and* $R = \{1\}$.
- *Property* $\Pi_5$ *is specified by automaton* $\mathcal{A}_{\Pi_5}$ *depicted on Fig. 5. Its set of states is* $\{1,2,3,4\}$*, the initial state is* $1$*, and we have* $P = \{3\}$ *and* $R = \emptyset$.

**Properties of automata.** Now we give a property of Streett automata related to their accepting pairs. Indeed given a Streett $m$-obligation automaton (with $m$ accepting pairs), it is possible to characterize the language accepted by the automaton resulting in "forgetting" some accepting pairs of the initial automaton. This is formalized as follows.

LEMMA 5.1 (FORGETTING ACCEPTING PAIRS FOR OBLIGATION PROPERTIES) *Given a* $m$*-automaton* $\mathcal{A}_\Pi = (Q, q_{\text{init}}, \Sigma, \longrightarrow, \{(R_1, P_1), \ldots, (R_m, P_m)\})$ *recognizing a r-property* $\Pi$*. Following [1],* $\Pi$ *can be expressed as* $\bigcap_{i=1}^{m} \Pi_i$ *where the* $\Pi_i$ *are obligation r-properties.*
*Given a subset* $X \subseteq \{1, \ldots, m\}$*, the automaton* $\mathcal{A}_{\Pi/X} = (Q, q_{\text{init}}, \Sigma, \longrightarrow, \{(R_i, P_i) \mid i \in X\})$ *recognizes the r-property* $\bigcap_{i \in X} \Pi_i$.

**Proof.** For infinite execution sequences, this proof has been done in [1]. For finite execution sequences, the proof is a straightforward adaptation. ■

## 5.1 Synthesis of Streett automata from DFAs

A Deterministic Finite-state Automaton (DFA) [7], is defined relatively to an alphabet $\Sigma$, and is formally defined as a tuple $(Q, q_{\text{init}}, \longrightarrow, F)$ where $Q$ is a finite set of states, $q_{\text{init}} \in Q$ is the initial state, $\rightarrow$: $Q \times \Sigma \rightarrow Q$ is the transition function, and $F \subseteq Q$ is the set of accepting states.

In the following $\psi = (Q^\psi, {q_{\text{init}}}^\psi, \longrightarrow_\psi, F^\psi)$ designates a DFA recognizing a finitary property $\psi$.

DEFINITION 5.4 (DFA TO STREETT SAFETY AUTOMATA) *Given a DFA $\mathcal{A}_\psi$ recognizing a finitary property $\psi \subseteq \Sigma^*$. We define the transformation* $\mathrm{DFA2StreettSafety}(\mathcal{A}_\psi) = \mathcal{A}_\Pi = (Q^{\mathcal{A}_\Pi}, {q_{\text{init}}}^{\mathcal{A}_\Pi}, \longrightarrow_{\mathcal{A}_\Pi}, \{(\emptyset, P)\})$ *such that:*

- $Q^\Pi = Q^{\mathcal{A}_\psi} \cap F \cup \{sink\}$, *where* $sink \notin Q^{\mathcal{A}_\psi}$,
- ${q_{\text{init}}}^{\mathcal{A}_\Pi} = {q_{\text{init}}}^{\mathcal{A}_\psi}$ *if* ${q_{\text{init}}}^{\mathcal{A}_\psi} \in F^{\mathcal{A}_\psi}$, *and sink else,*
- $\rightarrow_\Pi$ *is defined as the smallest relation verifying:*
  - $q \xrightarrow{a}_{\mathcal{A}_\Pi} q'$ *if* $q \in F \wedge q' \in F \wedge q \xrightarrow{a}_{\mathcal{A}_\psi} q'$ (TSAF1)
  - $q \xrightarrow{a}_{\mathcal{A}_\Pi} sink$ *if* $q' \notin F \wedge q \xrightarrow{a}_{\mathcal{A}_\psi} q'$ (TSAF2)
- $P = F$, $(m = 1)$

One can notice that the resulting automaton is indeed a Streett safety automaton as $R = \emptyset$ and there is no transition from $\overline{P}$-states to $P$-states.

DEFINITION 5.5 (DFA TO STREETT GUARANTEE AUTOMATA) *Given a DFA $\mathcal{A}_\psi$ recognizing a finitary property $\psi \subseteq \Sigma^*$. We define the transformation* $\mathrm{DFA2StreettGuarantee}(\mathcal{A}_\psi) = \mathcal{A}_\Pi = (Q^{\mathcal{A}_\Pi}, {q_{\text{init}}}^{\mathcal{A}_\Pi}, \longrightarrow_{\mathcal{A}_\Pi}, \{(R, \emptyset)\})$ *such that:*

- $Q^\Pi$ *is the minimal subset of* $Q^{\mathcal{A}_\psi}$ *of attainable states with* $\longrightarrow_{\mathcal{A}_\Pi}$ *from the initial state* ${q_{\text{init}}}^{\mathcal{A}_\Pi}$
- ${q_{\text{init}}}^{\mathcal{A}_\Pi} = {q_{\text{init}}}^{\mathcal{A}_\psi}$,
- $\rightarrow_\Pi$ *is defined as the smallest relation verifying:*
  - $q \xrightarrow{a}_{\mathcal{A}_\Pi} q$ *if* $\exists q' \in Q^{\mathcal{A}_\psi} \cdot q \xrightarrow{a}_{\mathcal{A}_\psi} q' \wedge q \in F$ (TGUAR1)
  - $q \xrightarrow{a}_{\mathcal{A}_\Pi} q'$ *if* $q \notin F \wedge q \xrightarrow{a}_{\mathcal{A}_\psi} q'$ (TGUAR2)
- $R = F$, $(m = 1)$

One can notice that the resulting automaton is indeed a Streett guarantee automaton as $P = \emptyset$ and there is no transition from $R$ to $\overline{R}$. This automaton is not minimal for $R$-states, these states can be merged.

DEFINITION 5.6 (DFA TO STREETT RESPONSE AUTOMATA) *Given a DFA $\mathcal{A}_\psi$ recognizing a finitary property $\psi \subseteq \Sigma^*$. We define the transformation* $\mathrm{DFA2StreettResponse}(\mathcal{A}_\psi) = \mathcal{A}_\Pi = (Q^{\mathcal{A}_\Pi}, {q_{\text{init}}}^{\mathcal{A}_\Pi}, \longrightarrow_{\mathcal{A}_\Pi}, \{(R, \emptyset)\})$ *such that:*

- $Q^\Pi = Q^{\mathcal{A}_\psi}$,
- ${q_{\text{init}}}^{\mathcal{A}_\Pi} = {q_{\text{init}}}^{\mathcal{A}_\psi}$,
- $\rightarrow_\Pi$ *is defined as* $\rightarrow_{\mathcal{A}_\psi}$,
- $R = \{q \in Q^\Pi \cap F \mid Reach_{\mathcal{A}_\Pi}(q) \cap F \neq \emptyset\}$, $(m = 1)$

DEFINITION 5.7 (DFA TO STREETT PERSISTENCE AUTOMATA) *Given a DFA $\mathcal{A}_\psi$ recognizing a finitary property $\psi \subseteq \Sigma^*$. We define the transformation* $\mathrm{DFA2StreettPersistence}(\mathcal{A}_\psi) = \mathcal{A}_\Pi = (Q^{\mathcal{A}_\Pi}, {q_{\text{init}}}^{\mathcal{A}_\Pi}, \longrightarrow_{\mathcal{A}_\Pi}, \{(\emptyset, P)\})$ *such that:*

- $Q^\Pi = Q^{\mathcal{A}_\psi}$,
- ${q_{\text{init}}}^{\mathcal{A}_\Pi} = {q_{\text{init}}}^{\mathcal{A}_\psi}$,
- $\rightarrow_\Pi$ *is defined as* $\rightarrow_{\mathcal{A}_\psi}$,
- $P = \{q \in Q^\Pi \cap F \mid \exists q_0, \ldots, q_n \cdot \exists a_0, \ldots, a_{n-1} \cdot q_0 = q \wedge \exists i < n \cdot q_n = q_i \wedge \forall i \in \{1, \ldots, n\} \cdot q_i \in F \wedge q_i \xrightarrow{a_i}_{\mathcal{A}_\psi} q_{i+1}\}$, $(m = 1)$

**THEOREM 5.1** *Given a property $\psi$, defining a regular language over $\Sigma$ and recognized by a DFA $\mathcal{A}_\psi$, the safety (resp. guarantee, response, persistence) r-property $(X_f(\psi), X(\psi))$ where $X \in \{A, E, R, P\}$ is recognized by the Streett automaton obtained by the DFA to Streett transformation for safety (resp. guarantee, response, persistence) properties.*

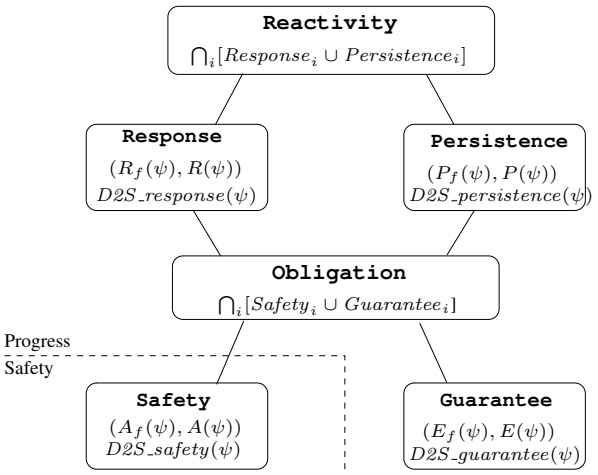


Figure 6: Hierarchal representation of the Safety-Progress classification of *r*-properties

# 6 Conclusion

We have extended the language-theoretic and automata views of the Safety-Progress classification in order to deal with finite-lenght sequences. These extension are consistent wrt. the considered views.

In Fig. 6 is depicted the hierarchal representation of the Safety-Progress classification of *r*-properties.

# References

[1] Chang, E., Manna, Z., Pnueli, A.: The safety-progress classification. Technical report, Stanford University, Dept. of Computer Science (1992) (document), 1, 1, 3, 4.2, 5, 5.1, 5

[2] Chang, E.Y., Manna, Z., Pnueli, A.: Characterization of temporal property classes. In: Automata, Languages and Programming. (1992) 474–486 1, 1

[3] Manna, Z., Pnueli, A.: A hierarchy of temporal properties (invited paper, 1989). In: PODC '90: Proceedings of the ninth annual ACM symposium on Principles of distributed computing, New York, NY, USA, ACM (1990) 377–410 1

[4] Lamport, L.: Proving the correctness of multiprocess programs. IEEE Trans. Softw. Eng. **3** (1977) 125–143 1

[5] Alpern, B., Schneider, F.B.: Defining liveness. Technical report, Cornell University, Ithaca, NY, USA (1984) 1

[6] Cerna, I., Pelanek, R.: Relating the hierarchy of temporal properties to model checking. In: Proceedings of Mathematical Foundations of Computer Science (MFCS 2003), Springer-Verlag (2003) 318–327 1

[7] Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, Massachusetts (1979) 1, 5.1

[8] Streett, R.S.: Propositional dynamic logic of looping and converse. In: STOC '81: Proceedings of the thirteenth annual ACM symposium on Theory of computing, New York, NY, USA, ACM (1981) 375–383 5