# SATISFACTION UP TO LIVENESS

## *Verification based on Exhaustive Testing*

Ulrich Ultes-Nitsche
*University of Southampton*
*Department of Electronics and Computer Science*
*Southampton, SO17 1BJ, United Kingdom*
uun@ecs.soton.ac.uk

**Abstract** An approximation to the usual linear satisfaction of temporal properties is discussed in this paper. It is called *satisfaction up to liveness* as it only differs from the linear satisfaction relation on liveness but not on safety properties. From the point of view of observation, satisfaction up to liveness and linear satisfaction are indistinguishable. Roughly speaking, by observing all *finite* behaviours (exhaustive testing), we do not know whether a system satisfies a property up to liveness or linearly. Being indistinguishable from linear satisfaction in terms of complete observations, satisfaction up to liveness offers an alternative approach to model-checking.

**Keywords:** Model-Checking, Exhaustive Testing, Fairness, Safety & Liveness.

## 1. INTRODUCTION

One checks the satisfaction of linear-time temporal properties by checking whether the set of behaviours of a system is a subset of the property [3]. Both, the behaviour and the property, are $\omega$-languages (languages of infinite word-length [15]). Practically, these $\omega$-languages are derived from descriptions written in a high-level specification language. To describe properties, for instance, linear-time temporal logics [7] are very frequently used. Set-inclusion is decidable for regular $\omega$-languages ($\omega$-languages representable by finite-automata [15]). However it is more complex than for regular languages (with finite word-length) [15], mainly because the class of regular $\omega$-languages accepted by deterministic Büchi-automata [5; 15] is only a proper subclass of the regular $\omega$-languages accepted by non-deterministic Büchi-automata.

---

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35533-7_26

Tommaso Bolognesi and Diego Latella (Eds.), *Formal Methods for Distributed System Development.*

Using the notion of *relative liveness* [9; 14] and viewing it as a satisfaction relation for temporal properties [13; 14] allows to reduce the satisfaction relation to deciding set-inclusion for finitary languages (i.e. languages of finite word-length). Obviously, by turning to considering finitary languages, we lose some information. It will be argued that the resulting *satisfaction up to liveness* relation, as it will be called, does not ignore any practically important information about the system. To underpin the arguments, different aspects of satisfaction up to liveness will be discussed in order to document the usefulness of the general concept. The main point in the argumentation will be that, by observing all finite behaviours of a system (i.e. by testing it exhaustively), linear satisfaction and satisfaction up to liveness cannot be distinguished from one another.

The paper is structured as follows: It starts with a first definition of satisfaction up to liveness in Section 2. Since throughout this paper only languages and $\omega$-languages appear, but no state-based representations like automata is used, Section 3 contains an outline of how the local language represents an automaton entirely. Section 4 reveals more about the structure of satisfaction up to liveness and how it relates to truth under fairness. Section 5 explains that satisfaction up to liveness is in fact an approximation to the linear property satisfaction. The formal apsects of satisfaction up to liveness and their practical consequences are summarised in Section 6. Section 7 concludes the paper, including an outlook on future work.

## 2. THE BASIC DEFINITION OF SATISFACTION UP TO LIVENESS

Before introducing the basic definition of satisfaction up to liveness, we review linear satisfaction of properties briefly. Let $\mathcal{A}$ be the finite set of actions that a system can perform. By $\mathcal{A}^\omega$ we denote the set of all infinitely long sequences of actions in $\mathcal{A}$. A behaviour $b$ of the system is an element of $\mathcal{A}^\omega$, i.e. $b = a_1a_2a_3\ldots$ such that all $a_i$ are in $\mathcal{A}$. The set $B$ of all behaviours of the system is therefore a subset of $\mathcal{A}^\omega$.[1] A property $P$, from an abstract point of view, is itself a subset of $\mathcal{A}^\omega$, representing all correct behaviours with respect to the intuitive meaning of the property. Hence the system, represented by its set of behaviours $B$, satisfies property $P$ (written: $B \models P$) if and only if all behaviours in $B$ are correct with respect to $P$ and thus are in $P$. This

[1] In terms of formal language theory, $b$ is an $\omega$-word and $B$ is an $\omega$-language on the alphabet $\mathcal{A}$.

*linear* satisfaction of properties is defined by

$$B \models P \iff B \subseteq P.$$

Since $B$ can only be a subset of $P$ if $B \cap P = B$, we can rewrite the above definition as

$$B \models P \iff B = B \cap P.$$

Let $\mathcal{A}^*$ denote the set of all finitely long sequences of actions in $\mathcal{A}$. For a subset $S$ of $\mathcal{A}^\omega$ let $pre(S)$ denote the set $\{w \in \mathcal{A}^* \mid \exists x \in \mathcal{A}^\omega : wx \in S\}$ of finite prefixes of behaviours in $S$. For an element $s$ of $\mathcal{A}^\omega$, let $pre(s) = pre(\{s\})$. Instead of saying that $B$ satisfies $P$ if and only if $B$ is equal to $B \cap P$, we relax this definition to requiring only that the finite prefixes of $B$ have to equal the finite prefixes of $B \cap P$. This relaxation leads us to the definition of satisfaction up to liveness. We write $B \Vdash P$ for saying $B$ satisfies $P$ up to liveness. We define:

$$B \Vdash P \iff pre(B) = pre(B \cap P).$$

What can we see by looking at this definition? Firstly, $B \models P$ always implies $B \Vdash P$ but not vice versa. An example for $B \Vdash P$ not implying $B \models P$ is $B = \{a, b\}^\omega$ and $P = \{a, b\}^* \cdot \{b\} \cdot \{a, b\}^\omega$.[2] Secondly, by observing systems, we cannot distinguish between a system satisfying a property linearly and a system satisfying the same property up to liveness: If we monitor all actions a system performs for an arbitrary period of time, we always see a (prefix-closed) set of prefixes of $B$. Since all these prefixes are prefixes of $B \cap P$, both, when $P$ is satisfied by $B$ up to liveness or linearly, linear satisfaction and satisfaction up to liveness give us the same view on the observable behaviour of a system.

In other words: From a set of behaviours $B$ satisfying a property $P$ up to liveness but not linearly, we can derive a set of behaviours $B'$ by removing all behaviours from $B$ that do not satisfy $P$ linearly. $B'$ satisfies $P$ linearly. But by observing their finite behaviours, $B$ cannot be distinguished from $B'$, i.e. $pre(B) = pre(B')$. Their differences could only be detected by observing systems for an infinite amount of time, being by that practically undetectable. An observer would consider a system with behaviour $B$ to be equivalent to a system with behaviour $B'$ even though they are distinguishable by the linear satisfaction relation. Since we would have to observe a system's finite behaviour completely to determine whether it satisfies a property up to liveness, we can say that satisfaction up to liveness is based on exhaustive testing.

[2] $P$ is in this example the temporal property "eventually $b$"; in temporal logic $P \equiv \Diamond b$.

Because the test whether $pre(B) = pre(B \cap P)$ can be reduced to checking whether $pre(B) \cap (\mathcal{A}^* \setminus pre(B \cap P))$ is empty, and the emptiness test as well as intersection and complementation is decidable for regular languages/$\omega$-languages [6; 15], this give us the algorithm to check satisfaction up to liveness automatically (model-checking) [7].

## 3. HOW THE LOCAL LANGUAGE ENCODES AUTOMATA

In this paper, we always consider languages and never talk about automata (with this section being an exception from the rule). That restriction seems to be relatively far away from practical systems where one would talk of events, states, and so on. The definitions of this paper, however, can be formulated more concisely on languages. A very common view is that by considering languages one loses (state-) information compared to the automaton level. In fact, this view is not correct as an automaton can be characterised completely by its *local language* [6]. We explain this argument briefly in this section.

Consider a finite automaton with $Q$ being its set of states, $\Sigma$ being its alphabet, $\delta \subseteq Q \times \Sigma \times Q$ being its transition relation, $q_0 \in Q$ being its initial state and $F \subseteq Q$ being its set of final states. A finite automaton accepts sequences of letters in $\Sigma$ that lead it from its initial state to some final state only taking transitions in $\delta$. For a sequence of letters, we usually do not know in which state we may end up (due to nondeterminism). So, by looking at the language, we lose state information. If, instead of keeping track of the letters that are the transition labels, we keep track of the triples *(state, letter, successorstate)*, we do not lose state information. The language consisting of transition triples is called the local language of the automaton. It can be represented by the same automaton except that the alphabet becomes $\Sigma' = \delta$ and the transition relation becomes $\delta' \subseteq Q \times (Q \times \Sigma \times Q) \times Q$ such that $(q, (q, a, p), p)$ is in $\delta'$ if and only if $(q, a, p)$ was in $\delta$. The so constructed automaton accepts exactly the local language of the former one.

From its local language a (reduced[3] non-empty) automaton can be reconstructed entirely. From the transition triples that occur as letters in the local language we obtain $\delta$, $Q$, and $\Sigma$. The initial state $q_0$ is the first component of the leftmost transition triple of any word in the local language. The set of final states contains all right components of rightmost transition triples of words in the local language. For instance, from

[3]reduced = all states are reachable from the initial state and from all states a final state is reachable.

knowing that $(q,a,p)(p,b,p)(p,a,r)$ is a word in the local language of an automaton, we learn that $q$, $p$, and $r$ are some of its states, $a$ and $b$ are letters in its alphabet, $(q,a,p)$, $(p,b,p)$, and $(p,a,r)$ are transitions of the automaton, $q$ is its initial state, and $r$ is one of its final states. Applying the projection homomorphism, that projects on the middle component of transition triples, to the local language gives us the language accepted by an automaton.

So, in fact, considering only languages in this paper is not a restriction. By considering the local language, we can take the concepts that we develop in this paper to the automaton level, without having to do any proofs on automata, but only more concisely on languages.

# 4. SATISFACTION OF THE PROPERTY IS ALWAYS POSSIBLE

In this section, we review the definition of satisfaction up to liveness. By giving an alternative, equivalent definition to the one in Section 2, we explore some more aspects of properties satisfied up to liveness. This second definition occurs under the name *relative liveness* in different contexts in [4; 9; 14], where it is mainly a derivate of *machine-closed structures* [1; 2] to classify properties with respect to other properties. A dual type of properties, *relative safety* properties can also be defined [9; 14]. In [14] relative liveness is used as a satisfaction relation as well as in [13], where it is named approximate satisfaction.

Considering our first definition of satisfaction up to liveness,

$$B \Vdash P \iff pre(B) = pre(B \cap P),$$

we recognise that $pre(B \cap P)$ is always a subset of $pre(B)$, because $B \cap P \subseteq B$. Therefore

$$B \Vdash P \iff pre(B) \subseteq pre(B \cap P),$$

emphasising that $B \Vdash P$ states something about the finite prefixes of behaviours in $B$ and their possible infinite continuation. Before being able to define what that "something" is that $B \Vdash P$ says about finite prefixes of $B$, we have to formalise the continuation of finite prefixes.[4] Let $b \in \mathcal{A}^*$ (remember that $\mathcal{A}$ is the set of actions occuring in the behaviours in $B$, i.e. $B \subseteq \mathcal{A}^\omega$). Then

$$cont(b,B) = \{x \in \mathcal{A}^\omega \mid bx \in B\}.$$

[4]To define the contiunation of finite prefixes, we borrow the notion *left-quotient* of a language by a word from formal language theory [6]. We use the different notation $cont(w,L)$ rather than $w^{-1}(L)$ because it appears to be more intuitive in our context.

So $x$ is in $cont(b, B)$ if an only if $b$ can be continued with $x$ resulting in a behaviour $bx$ that is in $B$. Using this notation, we re-define satisfaction up to liveness and prove the new definition to be equivalent to the first one.

**Lemma 1**

$$B \Vdash P \iff \forall b \in pre(B) : \exists x \in cont(b, B) : bx \in P.$$

Informally this lemma states that a behaviour satisfies a property up to liveness if and only if "all finite behaviours can be continued to an infinite behaviour satisfying the property".

**Proof** *If $pre(B) \subseteq pre(B \cap P)$, then all $b$ in $pre(B)$ are also in the set $pre(B \cap P)$. Therefore, for all $b \in pre(B)$, there exists an $x \in cont(b, B)$ such that $bx \in P$. If for all $b \in pre(B)$ there exists $x \in cont(b, B)$ such that $bx \in P$, then $b \in pre(B \cap P)$ for all $b \in pre(B)$ and thus $pre(B) \subseteq pre(B \cap P)$.* □

To be able to explain why the discussed satisfaction relation is called satisfaction *up to liveness*, we have to review the definition of safety and liveness properties [3], adapting it to the notation used in this paper. Note that any property is the intersection of a safety and a liveness property [3].

A property $P \subseteq \mathcal{A}^\omega$ is a safety property if and only if

$$\forall b \in \mathcal{A}^\omega : b \not\models P \Rightarrow \exists v \in pre(b) : \forall c \in \mathcal{A}^\omega : vc \not\models P.$$

A property $P \subseteq \mathcal{A}^\omega$ is a liveness property if and only if

$$\forall v \in \mathcal{A}^* : \exists c \in \mathcal{A}^\omega : vc \models P.$$

We show that for safety properties satisfaction up to liveness and linear satisfaction coincide:

**Lemma 2** *If $P \subseteq \mathcal{A}^\omega$ is a safety property, then for all $B \subseteq \mathcal{A}^\omega$:*

$$B \Vdash P \iff B \models P.$$

**Proof** *Because $B \models P$ always implies $B \Vdash P$, we only have to prove that $B \Vdash P$ implies $B \models P$. Recall that, if $B \Vdash P$, then $\forall b \in pre(B) : \exists x \in cont(b, B) : bx \in P$. Assuming $B \models P$ does not hold, we show a contradiction to $B \Vdash P$.*

*Assume $B \not\models P$. Then, because $P$ is a safety property, there exists $v \in pre(B)$ such that $vc \not\models P$, for all $c \in \mathcal{A}^\omega$. So, in particular, we have $vc \not\models P$, for all $c \in cont(v, B)$. Since $v$ is in $pre(B)$, this implies that*

$\forall b \in pre(B) : \exists x \in cont(b, B) : bx \in P$ *does not hold, contradicting that* $B \Vdash P$. □

So, for safety properties, there is no difference between satisfaction up to liveness and linear satisfaction. Only when checking liveness properties has one to distinguish between the two concepts. This fact led to the name *up to liveness*, stating that for checking whether a liveness property is satisfied linearly, knowing that it is satisfied up to liveness, we would have to perform additional checks.

## 5. SATISFACTION UP TO LIVENESS APPROXIMATES LINEAR SATISFACTION

To make the point again that the only difference between the usual linear satisfaction relation and satisfaction up to liveness can be revealed if one observes a system for an infinite amount of time, it is not surprising that satisfaction up to liveness can be regarded as an approximation to linear satisfaction. To make this precise, we present the characterisation of satisfaction up to liveness within the Cantor topology, i.e. the topological space over $\Sigma^\omega$ compatible with the following metric [6]. (For topological notions see [10].)

**Definition 3** *Let* $common(a, b)$ *designate the longest common prefix of two behaviours* $a$ *and* $b$ *in* $\mathcal{A}^\omega$. *We define the metric* $d(a, b)$ *by*

$$\forall a, b \in \mathcal{A}^\omega, a \neq b : d(a, b) = \frac{1}{|common(a, b)| + 1}$$

$$\forall b \in \mathcal{A}^\omega : d(b, b) = 0.$$

What does it mean in the given context that a satisfaction relation approximates another one? It means that the correctness criterion of the first satisfaction relation must imply that all behaviours of the system are arbitrarily close to meeting the second satisfaction relation's correctness criterion. Being arbitrarily close to meeting the correctness requirement must be translated into, in our context, that the correct behaviours of the system must be dense within the set of all behaviours. We can indeed derive this fact from knowing that a property is satisfied up to liveness, as stated by the following lemma [14]:

**Lemma 4** *Let* $B \subseteq \mathcal{A}^\omega$ *and* $P \subseteq \mathcal{A}^\omega$ *be the set of behaviours of a system and the property it should meet, respectively. Then* $B \Vdash P$ *if and only if* $B \cap P$ *is a dense set in* $B$.

**Proof** *Let $B \Vdash P$ and let $b \in B$. Then $pre(B) = pre(B \cap P)$. Thus, $pre(b) \subseteq pre(B \cap P)$, implying $\forall w \in pre(b) : \exists a \in B \cap P : w \in pre(a)$. So, for all $b \in B$ and all $\varepsilon > 0$ ($\varepsilon$ is related to $\frac{1}{|w|+1}$), there is $a \in B \cap P$ such that $d(a,b) < \varepsilon$. So $B \cap P$ is a dense set in $B$.*

*Let $B \cap P$ be a dense set in $B$. Then, for all $b \in B$ and all $\varepsilon > 0$, there exists $a \in B \cap P$ such that $d(a,b) < \varepsilon$. Let $w \in pre(b)$ and $\varepsilon = \frac{1}{|w|+1}$. Let $a \in B \cap P$ such that $d(a,b) < \varepsilon$. Then $w \in pre(a)$. Because $w$ was chosen arbitrarily, $pre(B) \subseteq pre(B \cap P)$, implying $pre(B) = pre(B \cap P)$. So $B \Vdash P$.* □

If we agree that a coarser satisfaction relation is, from a practical point of view, as good as a more exact one, if we could only distinguish the two relations by observing a system for an infinitely long time, then we can define:

**Definition 5** *A satisfaction relation is a reasonable satisfaction relation for linear temporal properties whenever the satisfaction of property $P$ by behaviour $B$ implies that $B \cap P$ is dense in $B$.*

The condition "$B \cap P$ is dense in $B$" just states that, if $B \cap P$ can be distinguished from $B$, then only by observing $B$ until infinity. Obviously, linear satisfaction itself is *reasonable* with respect to the definition above: $B \models P$ is equivalent to $B \cap P = B$, which implies that $B \cap P$ is dense in $B$.

According to Lemma 4, satisfaction up to liveness is the weakest possible *reasonable* satisfaction relation for linear temporal properties: $B \vdash P$, for any *reasonable* satisfaction relation "$\vdash$", always implies $B \Vdash P$. So, alternatively, we could define a *reasonable* satisfaction relation to be one that implies $B \Vdash P$.

## 6. DISCUSSION

We discuss in this section whether satisfaction up to liveness offers a reasonable alternative to the usual satisfaction relation of properties that is called linear satisfaction in this paper. Not surprisingly is the author in favour of satisfaction up to liveness. However, we have to weigh the advantages and disadvantages at this point.

Obviously, satisfaction up to liveness and linear satisfaction are different satisfaction relations. But to distinguish a system that satisfies a property linearly from one that does only satisfy it up to liveness we may have to observe the two systems for an infinite amount of time. So even though there are differences these differences do not have an impact on the system behaviour as we will be able to observe it. If we

cannot observe the difference, it appears that the difference between the two concepts does not matter in practical terms. By arguments given in Section 5 we may regard linear satisfaction as the strongest and satisfaction up to liveness as the weakest *reasonable* satisfaction relation for linear-time temporal properties.

The advantage of satisfaction up to liveness is that by its definition, language inclusion has only to be checked on the level of finite-word languages rather than on $\omega$-languages as in the case of linear satisfaction. Satisfaction up to liveness comes with an exact notion of abstraction, namely weakly continuation-closed homomorphisms [14]. Linear satisfaction does not have such a clear and simple notion of abstraction.

The major disadvantage of satisfaction up to liveness is that it is not closed under intersection. If a system satisfies properties $P$ and $P'$ up to liveness, it does not necessarily satisfy $P \cap P'$ up to liveness. This fact states a lack of compositionality. But if either $P$ or $P'$ is a safety property, then intersection is preserved. In contrast, linear satisfaction is preserved under intersection in any case.

A big bonus of satisfaction up to liveness is that it includes an inherent fairness condition [14]. If a property is satisfied up to liveness then it will be satisfied linearly under an abstract notion of fairness that can be made concrete to *strong fairness* [8] for particular implementations of the system [14]. Being able to ignore extreme execution scenarios without having to deal with fairness explicitly is a definite advantage of satisfaction up to liveness, having a strong impact on applying model-checking to practical problems [12]. We consider here a very simple example taken from [14]:

Consider a system that on receiving a *request* may produce either a *result* of some sort or *rejects* the request. The decision is taken internally (*yes* or *no*) before its outcome is reported to the requesting client. The decision is dependent on the availability of a resource that is handled by a resource manager and may be either *locked* or *free*. The system is described as a Petri net in Figure 1.

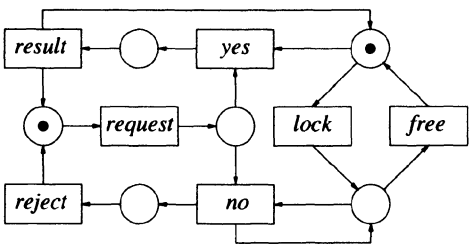


*Figure 1* A small system

The possible behaviours of the system are represented by the finite labelled transition system shown in Figure 2 (the reachability graph of the Petri net). The initial state is shaded grey.

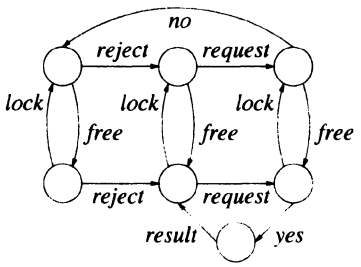


*Figure 2* The behaviours of the small system

From Figure 2, it is easy to see that our system does not satisfy linearly the propositional linear-time temporal logic [7] property $\Box\Diamond(result)$. This property requires that infinitely often a result will be produced by the system. Let $\mathcal{A} = \{request, reject, result, yes, no, lock, free\}$. Then the language representation of this property is $\mathcal{A}^* \cdot (result \cdot \mathcal{A}^*)^\omega$.

The property is violated because, for the example, whenever a request is made the resource could be unavailable. For instance, $lock \cdot (request \cdot no \cdot reject)^\omega$ is such a behaviour violating the linear satisfaction of the property. However, such a behaviour is highly unfair [8] and the specification in Figure 1 itself is very reasonable. One way of dealing with this situation would be to add additional fairness constraints to the specification which would make the design process of the system more complicated. A less complicated way of dealing with it would be using the inherent fairness condition of satisfaction up to liveness.

The property $\Box\Diamond(result)$ is satisfied by the system of Figure 1 up to liveness, because satisfaction up to liveness ignores extreme execution scenarios such as $lock \cdot (request \cdot no \cdot reject)^\omega$. Quite frequently, in communication protocols and Intelligent Network service specifications, such "extreme execution scenarios" are part of the specification ("whenever A calls B, B's line is busy") but should be ignored by the satisfaction relation [12]. Satisfaction up to liveness does exactly this job without additional effort, making it useful in practice [12].

To summarise, satisfaction up to liveness is at least an alternative, worth considering way of checking properties of reactive systems that is observationally indistinguishable from linear satisfaction.

# 7. CONCLUSIONS AND FUTURE WORK

Different aspects of satisfaction up to liveness as an alternative satisfaction relation to the traditional linear satisfaction have been discussed in this paper. One of the major points is that systems satisfying a property up to liveness cannot be distinguished from systems satisfying the property linearly by observing (even all of) their finite behaviours. The major contribution of the paper is to give alternative presentations of satisfaction up to liveness in a unified framework and to interpret their mathematical foundation with respect to the observability of system behaviours. By this it is possible to compare satisfaction up to liveness with the usual concept of (linear) satisfaction, discussing their similarities and differences. In addition, a criterion for a relation to be a *reasonable* satisfaction relation for linear-time temporal properties has been given (again with respect to observability).

The concept of satisfaction up to liveness of non-safety properties is relatively close to linear satisfaction of safety properties (in the case of safety properties it is the identical concept). In both cases, one has to look at finitary languages (finitely long words). So in order to model-check satisfaction up to liveness, improvements might be gained by considering the particular aspects of model-checking safety properties [11]. Exploring possible improvements in that direction will be part of future work.

In this paper we were talking constantly about observability, knowing that the system satisfies a property (up to liveness). The notion of observing a system is understood by means of exhaustive testing. It will therefore also be part of the future work to explore in detail the link to testing and test case generation from specifications verified up to liveness.

Probabilistic satisfaction relations [17; 18] appear to lie somewhat in between linear satisfaction and satisfaction up to liveness. It will be part of future research to see how they could be embedded into the definition of a *reasonable* satisfaction relation as given in this paper.

## References

[1] M. Abadi and L. Lamport. The existence of refinement mappings. SRC Report 29, DEC System Research Center, July 1988.

[2] M. Abadi and L. Lamport. Composing specifications. SRC Report 66, DEC System Research Center, October 1990.

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

[4] R. Alur and T. A. Henzinger. Local liveness for compositional modeling of fair reactive systems. In P. Wolper, editor, *Computer Aided Verification (CAV) '95*,

volume 939 of *Lecture Notes in Computer Science*, pages 166–179. Springer, 1995.

[5] J. Büchi. On a decision method in restricted second order arithmetic. In E. Nagel et al., editors, *Proceedings of the International Congress on Logic, Methodology and Philosophy of Science 1960*, pages 1–11. Stanford University Press, 1962.

[6] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, New York, 1974.

[7] E. A. Emerson. Temporal and modal logic. In van Leeuwen [[16]], pages 995–1072.

[8] N. Francez. *Fairness*. Springer Verlag, New York, first edition, 1986.

[9] T. A. Henzinger. Sooner is safer than later. *Information Processing Letters*, 43:135–141, 1992.

[10] J. L. Kelley. *General Topology*. Van Nostrand, Princeton, 1955.

[11] O. Kupferman and M. Y. Vardi. Model-checking of safety properties. In N. Halbwachs and D. Peled, editors, *CAV'99*, volume 1633 of *Lecture Notes in Computer Science*, pages 172–183, Trento, Italy, 1999. Springer Verlag.

[12] U. Nitsche. Application of formal verification and behaviour abstraction to the service interaction problem in intelligent networks. *Journal of Systems and Software*, 40(3):227–248, March 1998. ISSN:0164-1212.

[13] U. Nitsche and P. Ochsenschläger. Approximately satisfied properties of systems and simple language homomorphisms. *Information Processing Letters*, 60:201–206, 1996. ISSN: 0020-0190.

[14] U. Nitsche and P. Wolper. Relative liveness and behavior abstraction (extended abstract). In *Proceedings of the 16th ACM Symposium on Principles of Distributed Computing (PODC'97)*, pages 45–52, Santa Barbara, CA, 1997.

[15] W. Thomas. Automata on infinite objects. In van Leeuwen [[16]], pages 133–191.

[16] J. van Leeuwen, editor. *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*. Elsevier, 1990.

[17] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 327–338, Portland, October 1985.

[18] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the 1st Symposium on Logic in Computer Science*, Cambridge, June 1986.