On the Construction of Fine Automata for Safety Properties

Orna Kupferman and Robby Lampert

Hebrew University, School of Engineering and Computer Science, Jerusalem 91904, Israel
Email: {orna,robil}@cs.huji.ac.il

Abstract. Of special interest in formal verification are *safety* properties, which assert that the system always stays within some allowed region. Each safety property ψ can be associated with a set of *bad prefixes*: a set of finite computations such that an infinite computation violates ψ iff it has a prefix in the set. By translating a safety property to an automaton for its set of bad prefixes, verification can be reduced to reasoning about finite words: a system is correct if none of its computations has a bad prefix. Checking the latter circumvents the need to reason about cycles and simplifies significantly methods like symbolic fixed-point based verification, bounded model checking, and more.

A drawback of the translation lies in the size of the automata: while the translation of a safety LTL formula ψ to a nondeterministic Büchi automaton is exponential, its translation to a tight bad-prefix automaton — one that accepts all the bad prefixes of ψ, is doubly exponential. Kupferman and Vardi showed that for the purpose of verification, one can replace the tight automaton by a fine automaton — one that accepts at least one bad prefix of each infinite computation that violates ψ. They also showed that for many safety LTL formulas, a fine automaton has the same structure as the Büchi automaton for the formula. The problem of constructing fine automata for general safety LTL formulas was left open. In this paper we solve this problem and show that while a fine automaton cannot, in general, have the same structure as the Büchi automaton for the formula, the size of a fine automaton is still only exponential in the length of the formula.

1 Introduction

Today's rapid development of complex and safety-critical systems requires reliable verification methods. In formal verification, we verify that a system meets a desired property by checking that a mathematical model of the system meets a formal specification that describes the property. Of special interest are properties asserting that observed behavior of the system always stays within some allowed region, in which nothing "bad" happens. For example, we may want to assert that every message received was previously sent. Such properties of systems are called *safety properties*. Intuitively, a property ψ is a safety property if every violation of ψ occurs after a finite execution of the system. In our example, if in a computation of the system a message is received without previously being sent, this occurs after some finite execution of the system.

In order to formally define what safety properties are, we refer to computations of a nonterminating system as infinite words over an alphabet Σ. Typically, $\Sigma = 2^{AP}$, where

AP is the set of the system's atomic propositions. Consider a language L of infinite words over Σ. A finite word x over Σ is a *bad prefix* for L iff for all infinite words y over Σ, the concatenation $x \cdot y$ of x and y is not in L. Thus, a bad prefix for L is a finite word that cannot be extended to an infinite word in L. A language L is a *safety language* if every word not in L has a finite bad prefix. Linear properties of nonterminating systems are often specified using nondeterministic Büchi automata on infinite words (NBW) or linear temporal logic (LTL) formulas. We say that an NBW is a safety NBW if it recognizes a safety language. Similarly, an LTL formula is a safety formula if the set of computations that satisfy it form a safety language.

In addition to proof-based methods for the verification of safety properties [MP92,MP95], there is extensive work on model checking of safety properties. General methods for model checking of linear properties are based on a construction of an NBW $\mathcal{A}_{\neg\psi}$ that accepts exactly all the infinite computations that violate the property ψ [LP85,VW94]. Verification of a system M with respect to ψ is reduced to checking the emptiness of the product of M and $\mathcal{A}_{\neg\psi}$ [VW86]. This check can be performed on-the-fly and symbolically [CVWY92,GPVW95]. When ψ is an LTL formula, the size of $\mathcal{A}_\psi$ is exponential in the length of ψ, and the complexity of verification that follows is PSPACE, with a matching lower bound [SC85].

When ψ is a safety property, the NBW $\mathcal{A}_{\neg\psi}$ can be replaced by *bad-pref*$(\mathcal{A}_\psi)$ – an automaton on finite words (NFW) that accepts exactly all the bad prefixes of ψ [KV01a]. This has several advantages, as reasoning about finite words is simpler than reasoning about infinite words: symbolic reasoning (in particular, bounded model checking procedures) need not look for loops (cf. [HKSV97]) and can, instead, apply backward or forward reachability analysis [BCM$^+$92,IN97,CBRZ01]. In fact, the construction of *bad-pref*$(\mathcal{A}_\psi)$ reduces the model-checking problem to the problem of invariance checking [MP92], which is amenable to both model-checking techniques [BCM$^+$92,IN97] and deductive verification techniques [BM83,OSR95,MAB$^+$94]. In addition, using *bad-pref*$(\mathcal{A}_\psi)$, we can return to the user a finite error trace, which is a bad prefix, and which is often more helpful than an infinite error trace.

The construction of *bad-pref*$(\mathcal{A}_\psi)$ is studied in [KV01a]. As shown there, while the translation of ψ to the NBW $\mathcal{A}_{\neg\psi}$ involves an exponential blow up, the NFW *bad-pref*$(\mathcal{A}_\psi)$ may be doubly-exponential in the length of ψ. This discouraging blow up is reflected in the fact that users have restricted attention to invariance checking [GW91,McM92,Val93,MR97], have assumed that a general safety property is given by the set of its bad prefixes [GW91], or have worked with variants of $\mathcal{A}_{\neg\psi}$ that approximate the set of bad prefixes [GH01].

Such an approximation is also studied in [KV01a], which release the requirement on *bad-pref*$(\mathcal{A}_\psi)$ and seek, instead, an NFW that need not accept all the bad prefixes, yet must accept at least one bad prefix of every infinite computation that does not satisfy ψ. Such an NFW is said to be *fine* for ψ. For example, an NFW that accepts all the finite words in $0^* \cdot 1 \cdot (0+1)$ does not accept all the bad prefixes of the safety language $\{0^\omega\}$; in particular, it does not accept the minimal bad prefixes in $0^* \cdot 1$. Yet, such an NFW is fine for $\{0^\omega\}$. Indeed, every infinite word that is different from 0^ω has a prefix in $0^* \cdot 1 \cdot (0+1)$. In practice, almost all the benefit that one obtains from *bad-pref*$(\mathcal{A}_\psi)$ can also be obtained from a fine automaton. It is shown in [KV01a] that for many safety formulas ψ, a fine

automaton has the same structure as the NBW $\mathcal{A}_{\neg\psi}$, and can be constructed by redefining its set of accepting states.

The construction in [KV01a] has been optimized and implemented in [Lat03], which also describes an implementation of the algorithm for checking whether an LTL formula ψ is such that a fine automaton for it can be easily constructed from $\mathcal{A}_{\neg\psi}$. The problem of constructing fine automata for general safety LTL formulas was left open in both papers. We note that the problem was left open not due to a lack of interest; to the opposite – in practice we do come across formulas for which the constructions in [KV01a,Lat03,GH01] do not work, and a fine automaton is desirable. In this paper we solve this problem and show how to construct, for every LTL formula, a fine automaton of size only exponential in the length of the formula. We also show that while a fine automaton for ψ cannot, in general, have the same structure as the NBW $\mathcal{A}_{\neg\psi}$, its construction is similar to that of $\mathcal{A}_{\neg\psi}$. The key idea behind our construction is that even though it is impossible to bound the length of a good prefix, it is possible to bound the number of visits that a good prefix has to the set of accepting states. In addition to the above positive result, we give negative results about fine automata constructed from $\mathcal{A}_{\psi}$ (rather than $\mathcal{A}_{\neg\psi}$), and about the possibility of using properties of the NBWs obtained from LTL formulas (c.f., reverse determinism, single accepting run, obtained by alternation removal of a very weak alternating automaton) in order to define a fine automaton with the same structure as $\mathcal{A}_{\psi}$.

From a theoretical point of view, we find our result very interesting: the "fine-automaton" problem belongs to a class of long-standing open problems that refer to the power of nondeterminism, and this work is the first to solve a problem from this class. To get the flavor of this class, consider an NBW $\mathcal{A}$, and assume we want to translate it to an equivalent nondeterministic co-Büchi automaton[1] (NCW). The best known procedure for doing it is to determinize $\mathcal{A}$ to a Streett automaton and then define the co-Büchi condition on top of the deterministic automaton. This involves an exponential blow-up, with no matching lower bound, and the question about the existence of an efficient NBW-to-NCW translation that avoids determinization and goes directly to an NCW is open. More problems in this class include a translation of an NBW to an automaton on finite words (accepting a language whose limit is the language of the original automaton, see [Lan69]), a translation of nondeterministic tree automata for a derivable language to a word automaton for the language that derives it (see [KSV96]), and more. The "fine-automaton" problem is the first problem in this class to which we are able to avoid determinization. Indeed, the existing solution, from [KV01a], applies the subset construction. We hope the solution would shed light on the other problems in this class. In particular, we believe that the idea used here, of using the complementary nondeterministic automaton instead of a deterministic automaton would be useful for the other problems: both determinization and complementation involves an exponential blow-up. Nevertheless, when the language is given in terms of an LTL formula, complementation is for free. Thus, as is the case with fine-automata, a construction that uses the complementing automaton rather than the deterministic one is exponentially better.

[1] Such a translation is not always possible, and attention is restricted to languages for which it is possible. The need for such a translation arises from the fact that the transition from NCW to the alternation-free μ-calculus is linear, whereas the transition from NBW to the alternation-free μ-calculus is exponential. For details, see [KV05a].

From a practical point of view, our solution shows that reasoning about all safety formulas can be done symbolically with a single exponential blow-up and a single nested fixed-point. In Section 5, we discuss the application of our result in run-time verification and bounded model checking.

2 Preliminaries

Safety and co-safety languages. Consider a language $L \subseteq \Sigma^\omega$ of infinite words over the alphabet Σ. A finite word $x \in \Sigma^*$ is a *bad prefix* for L iff for all $y \in \Sigma^\omega$, we have $x \cdot y \notin L$. Thus, a bad prefix is a finite word that cannot be extended to an infinite word in L. Note that if x is a bad prefix, then all the finite extensions of x are also bad prefixes. A language L is a *safety* language iff every infinite word $w \notin L$ has a finite bad prefix. For a safety language L, we denote by *bad-pref*(L) the set of all bad prefixes for L.

For a language $L \subseteq \Sigma^\omega$ (Σ^*), we use *comp*(L) to denote the complement of L; i.e., *comp*$(L) = \Sigma^\omega \setminus L$ ($\Sigma^* \setminus L$, respectively). We say that a language $L \subseteq \Sigma^\omega$ is a *co-safety* language iff *comp*(L) is a safety language. (The term used in [MP92] is *guarantee* language.) Equivalently, L is co-safety iff every infinite word $w \in L$ has a *good prefix* $x \in \Sigma^*$: for all $y \in \Sigma^\omega$, we have $x \cdot y \in L$. For a co-safety language L, we denote by *good-pref*(L) the set of good prefixes for L. Note that for a safety language L, we have that *good-pref*(*comp*(L)) = *bad-pref*(L). Thus, in order to construct the set of bad prefixes for a safety property, one can construct the set of good prefixes for its complementary language.

Word automata. Given an alphabet Σ, an *infinite word over* Σ is an infinite sequence $w = \sigma_1 \cdot \sigma_2 \cdots$ of letters in Σ. We denote by w^l the suffix $\sigma_l \cdot \sigma_{l+1} \cdot \sigma_{l+2} \cdots$ of w. A *nondeterministic Büchi word automaton* (NBW, for short) is $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, F \rangle$, where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, $Q_0 \subseteq Q$ is a set of initial states, and $F \subseteq Q$ is a set of accepting states. If $|Q_0| = 1$ and δ is such that for every $q \in Q$ and $\sigma \in \Sigma$, we have that $|\delta(q, \sigma)| = 1$, then $\mathcal{A}$ is a *deterministic* Büchi word automaton (DBW, for short).

Given an input word $w = \sigma_0 \cdot \sigma_1 \cdots$ in Σ^ω, a *run* of $\mathcal{A}$ on σ is a sequence $r_0, r_1, \ldots$ of states in Q, such that $r_0 \in Q_0$ and for every $i \geq 0$, we have $r_{i+1} \in \delta(r_i, \sigma_i)$; i.e., the run starts in one of the initial states and obeys the transition function. Note that a nondeterministic automaton can have many runs on w. In contrast, a deterministic automaton has a single run on w. For a run r, let $inf(r)$ denote the set of states that r visits infinitely often. That is,

$$inf(r) = \{q \in Q \ : r_i = q \text{ for infinitely many } i \geq 0\}.$$

As Q is finite, it is guaranteed that $inf(r) \neq \emptyset$. The run r is *accepting* iff $inf(r) \cap F \neq \emptyset$. That is, iff there exists a state in F that r visits infinitely often. A run that is not accepting is *rejecting*. An NBW $\mathcal{A}$ accepts an input word w iff there exists an accepting run of $\mathcal{A}$ on w. The *language* of an NBW $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. We assume that a given NBW $\mathcal{A}$ has no empty states (that is, at least one word is accepted from each state – otherwise we can remove the state).

We say that the automaton $\mathcal{A}$ over infinite words is a safety (co-safety) automaton iff $\mathcal{L}(\mathcal{A})$ is a safety (co-safety) language. We use *bad-pref*$(\mathcal{A})$, *good-pref*$(\mathcal{A})$, and *comp*$(\mathcal{A})$ to abbreviate *bad-pref*$(\mathcal{L}(\mathcal{A}))$, *good-pref*$(\mathcal{L}(\mathcal{A}))$, and *comp*$(\mathcal{L}(\mathcal{A}))$, respectively.

Linear Temporal Logic. The logic *LTL* is a linear temporal logic. Formulas of LTL are constructed from a set AP of atomic propositions using the usual Boolean operators and the temporal operators G ("always"), F ("eventually"), X ("next time"), and U ("until"). Formulas of LTL describe computations of systems over AP. For example, the LTL formula $G(req \rightarrow F\, ack)$ describes computations in which every position in which req holds is eventually followed by a position in which ack holds. For the detailed syntax and semantics of LTL, see [Pnu81]. The *model-checking problem* for LTL is to determine, given an LTL formula ψ and a system M, whether all the computations of M satisfy ψ.

General methods for LTL model checking are based on translation of LTL formulas to nondeterministic Büchi word automata:

Theorem 1. [VW94] *Given an LTL formula ψ, one can construct an NBW $\mathcal{A}_\psi$ that accepts exactly all the computations that satisfy ψ. The size of $\mathcal{A}_\psi$ is, in the worst case, exponential in the length of ψ.*

Given a system M and a property ψ, model checking of M with respect to ψ is reduced to checking the emptiness of the product of M and $\mathcal{A}_{\neg\psi}$ [VW94]. This check can be performed on-the-fly and symbolically [CVWY92,GPVW95,TBK95], and the complexity of model checking that follows is PSPACE, with a matching lower bound [SC85].

3 Detecting Bad and Good Prefixes

Recall that the model-checking problem for an LTL formula ψ involves the construction of an NBW that accepts infinite computations that violate ψ. As discussed in Section 1, when ψ is a safety property, it is desirable to construct, instead, a *nondeterministic automaton on finite words* (NFW, for short) for the bad prefixes of ψ. In this section we recall the relevant results from [KV01a], and the problem that has been left open in [KV01a].

Consider a safety NBW $\mathcal{A}$. If $\mathcal{A}$ is deterministic, we can construct a *deterministic automaton on finite words* (DFW, for short) for *bad-pref*$(\mathcal{A})$ by defining the set of accepting states to be the set of states s for which $\mathcal{A}$ with initial state s is empty. Likewise, if $\mathcal{A}$ is a co-safety automaton, we can construct a DFW for *good-pref*$(\mathcal{A})$ by defining the set of accepting states to be the set of states s for which $\mathcal{A}$ with initial state s is universal.

When $\mathcal{A}$ is nondeterministic, the story is more complicated. Even if we are after a nondeterministic, rather than a deterministic, automaton for the bad or good prefixes, the transition from infinite words to finite words involves an exponential blow-up. Formally, we have the following.

Theorem 2. [KV01a] *Consider an NBW $\mathcal{A}$ of size n.*

1. *If $\mathcal{A}$ is a safety automaton, the size of an NFW for bad-pref$(\mathcal{A})$ is $2^{\Theta(n)}$.*
2. *If $\mathcal{A}$ is a co-safety automaton, the size of an NFW for good-pref$(\mathcal{A})$ is $2^{\Theta(n)}$.*

The lower bound in Theorem 2 for the case $\mathcal{A}$ is a safety automaton is not surprising. Essentially, it follows from the fact that *bad-pref*$(\mathcal{A})$ refers to words that are not accepted by $\mathcal{A}$. Hence, it has the flavor of complementation, and complementation of nondeterministic automata involves an exponential blow-up [MF71]. The second blow up, however, in going from a co-safety automaton to a nondeterministic automaton for its good prefixes

is surprising. Since its proof in [KV01a] highlights our contribution here, we describe it below.

For $n \geq 1$, let $\Sigma_n = \{1, \ldots, n, \&\}$. We define L_n as the language of all words $w \in \Sigma_n^\omega$ such that w contains at least one $\&$ and the letter after the first $\&$ is either $\&$ or it has already appeared somewhere before the first $\&$. The language L_n is a co-safety language. Indeed, each word in L_n has a good prefix (e.g., the one that contains the first $\&$ and its successor). We can recognize L_n with an NBW with $O(n)$ states (the NBW guesses the letter that appears after the first $\&$). Obvious good prefixes for L_n are $12\&\&$, $123\&2$, etc. That is, prefixes that end one letter after the first $\&$, and their last letter is either $\&$ or has already appeared somewhere before the $\&$. We can recognize these prefixes with an NFW with $O(n)$ states. But L_n also has some less obvious good prefixes, like $1234\cdots n\&$ (a permutation of $1 \ldots n$ followed by $\&$). These prefixes are indeed good, as every suffix we concatenate to them would start in either $\&$ or a letter in $\{1, \ldots, n\}$, which has appeared before the $\&$. To recognize these prefixes, an NFW needs to keep track of subsets of $\{1, \ldots, n\}$, for which it needs 2^n states. Consequently, an NFW for *good-pref*(L_n) must have at least 2^n states.

As described in the proof, some good prefixes for L_n (the "obvious prefixes") can be recognized by a small NFW. What if we give up the non-obvious prefixes and construct an NFW $\mathcal{A}'$ that accepts only the "obvious subset" of L_n? It is not hard to see that each word in L_n has an obvious prefix. Thus, while $\mathcal{A}'$ does not accept all the good prefixes, it accepts at least one prefix of every word in L. This useful property of $\mathcal{A}'$ is formalized below.

Consider a safety language L. We say that a set $X \subseteq$ *bad-pref*(L) is a *trap* for L iff every word $w \notin L$ has at least one bad prefix in X. Thus, while X need not contain all the bad prefixes for L, it must contain sufficiently many prefixes to "trap" all the words not in L. Dually, a trap for a co-safety language L is a set $X \subseteq$ *good-pref*(L) such that every word $w \in L$ has at least one good prefix in X. We denote the set of all the traps, for an either safety or co-safety language L, by *trap*(L).

An NFW $\mathcal{A}$ is *fine* for a safety or a co-safety language L iff $\mathcal{A}$ accepts a trap for L. For example, an NFW that accepts $0^* \cdot 1 \cdot (0 + 1)$ does not accept all the bad prefixes of the safety language $\{0^\omega\}$; in particular, it does not accept the minimal bad prefixes in $0^* \cdot 1$. Yet, such an NFW is fine for $\{0^\omega\}$. Indeed, every infinite word that is different from 0^ω has a prefix in $0^* \cdot 1 \cdot (0 + 1)$. Likewise, the NFW is fine for the co-safety language $0^* \cdot 1 \cdot (0 + 1)^\omega$. In practice, almost all the benefit that one obtains from an NFW that accepts all the bad/good prefixes can also be obtained from a fine automaton. It is shown in [KV01a] that for natural safety formulas ψ, the construction of an NFW fine for ψ is as easy as the construction of $\mathcal{A}_{\neg\psi}$. In more details, if we regard $\mathcal{A}_{\neg\psi}$ as an NFW, with an appropriate definition of the set of accepting states, we get an automaton fine for ψ. For general safety formulas, the problem of constructing small fine automata was left open:

Open question [KV01a]: *Are there feasible constructions of fine automata for general safety and co-safety formulas?*

In the rest of the paper, we solve the question and discuss our solution.

4 Fine Automata for Safety and co-Safety Properties

In this section we study the size of fine automata for safety and co-safety properties. We start with the case the property is given by means of an NBW and show that then, the size of a fine automaton is polynomial in the sizes of the NBWs for the property and its negation. Since for LTL, the sizes of the NBWs for a formula and its negation are both exponential in the length of the formula, we conclude that LTL formulas have exponential fine automata.

4.1 Fine automata for safety NBW

We start with fine automata for safety NBWs. It is shown in [KV01a] that the transition from an NBW to a tight NFW for its bad prefixes is exponential, and that the exponential blow up follows from the fact that a complementing NBW can be construction from a tight NFW. When we consider fine automata, things are more complicated, as the fine NFW need not accept all bad prefixes. As we show below, however, a construction of fine automata still has the flavor of complementation, and must involve an exponential blow up.

Theorem 3. *Given a safety NBW $\mathcal{A}$ of size n, the size of an NFW fine for $\mathcal{A}$ is exponential in n.*

Proof: Since every tight NFW is fine, the upper bound follows from Theorem 2.

The lower bound follows from the exponential lower bound for NFW complementation [SS78]. As detailed below, given an NFW $\mathcal{U}$, one can construct an NBW $\mathcal{U}'$ of size linear in the size of $\mathcal{U}$, such that $\mathcal{L}(\mathcal{U}')$ is safety and an NFW fine for $\mathcal{U}'$ can be turned into an NFW for $\mathit{comp}(\mathcal{L}(\mathcal{U}))$ of the same size. It follows that a sub-exponential construction of fine automata would lead to a sub-exponential complementation construction, which is known to be impossible.

For an alphabet Σ with $\# \notin \Sigma$ and an NFW $\mathcal{U}$ for a language $L \subseteq \Sigma^*$, we define the language $L' = \{u\#^\omega : u \in L\} \cup \Sigma^\omega$ over the alphabet $\Sigma' = \Sigma \cup \{\#\}$. Note that every word $y \in (\Sigma')^\omega$ that is not in L' must contain at least one $\#$. Indeed, otherwise $y \in \Sigma^\omega$, which is contained in L'. Nevertheless, L' is a safety language, as every word $y \notin L'$ is of the form ht, where t is some string in $(\Sigma')^\omega$ and h is either $v\#$ for $v \in \Sigma^* \setminus L$, or $v\#^+a$ for $v \in (\Sigma')^\omega$ and $a \neq \#$. In both cases, h is a bad prefix.

Given $\mathcal{U}$, we construct the NBW $\mathcal{U}'$ for L' by adding to $\mathcal{U}$ two new states, which are going to be the only accepting states of $\mathcal{U}'$. The first state (which accepts Σ^ω) has a σ transition from the initial state and from itself, for every $\sigma \in \Sigma$. The second state (which accepts $\#^\omega$ and is reachable by traversing $v\#$ for $v \in L$) has a $\#$ transition from every accepting state of $\mathcal{U}$ and from itself. It is not hard to see that $\mathcal{L}(\mathcal{U}') = L'$.

Let $\mathcal{A}$ be an NFW fine for $\mathcal{U}'$. We now show how an NFW $\overline{\mathcal{U}}$ for $\mathit{comp}(L)$ can be obtained from $\mathcal{A}$. Given $\mathcal{A}$, we make the following two changes. First, we define the set of accepting states to be the set of states of $\mathcal{A}$ from which we can reach an accepting state by reading a string $t \in \#^+$. Then, we delete from $\mathcal{A}$ all $\#$ transitions.

We prove that $\overline{\mathcal{U}}$ indeed accepts $\mathit{comp}(L)$. Consider a word $w \in \Sigma^*$. Assume first that $w \in \mathit{comp}(L)$. Then, $w \notin L$ and $w\#^\omega \notin L'$. Therefore, as $\mathcal{A}$ is fine, a prefix of $w\#^\omega$ is

accepted by $\mathcal{A}$. Let h be the minimal prefix of $w\#^\omega$ accepted by $\mathcal{A}$, and let $|h| = l$. Since $\Sigma^\omega \subseteq L'$, the prefix h must contain at least one $\#$. Thus, $h = w\#^k$ for some $k \geq 1$. Let $r_0 r_1 \ldots r_l$ be an accepting run of $\mathcal{A}$ on h. Since we can reach the accepting state r_l from r_{l-k} by reading $\#^k$, then, by the definition of $\overline{\mathcal{U}}$, the state r_{l-k} is accepting in $\overline{\mathcal{U}}$, thus $r_0 r_1 \ldots r_{l-k}$ is an accepting run of $\overline{\mathcal{U}}$ on w, and $w \in \mathcal{L}(\overline{\mathcal{U}})$. Assume now that $w \in \mathcal{L}(\overline{\mathcal{U}})$. Let $|w| = m$ and let $r_0 r_1 \ldots r_m$ be an accepting run of $\overline{\mathcal{U}}$ on w. Then, by the definition of $\overline{\mathcal{U}}$, there is an accepting run $r_0 r_1 \ldots r_m r_{m+1} \ldots r_{m+k}$ of $\mathcal{A}$ on $h = w\#^k$ for some $k \geq 1$, which means that $h \in \textit{bad-pref}(L')$. As stated above, $h \in \textit{bad-pref}(L')$ if either $h = v\#t$ for $v \in \Sigma^* \setminus L$, or $h = v\#^+at$ for $v \in (\Sigma')^\omega$ and $a \neq \#$ (Note that since h need not be a minimal bad prefix, t may be a finite string over Σ'). Since h consists of w, which contains only letters from Σ, followed by $\#^k$, no letter $a \neq \#$ appears after the first $\#$ in h. Thus, h must have the form $v\#t$ for $v \notin L$. As the part of h preceding the first $\#$ is w, we have that $w \notin L$, and we are done. □

The proof of Theorem 3 shows that constructing a fine NFW for a safety NBW has the flavor of complementation. In Theorem 5, we show that the size of the complementary automaton is indeed the bottleneck in the construction of fine NFW for safety NBW, and that a fine automaton can be constructed with a blow up that depends on the size of the complementary automaton.

4.2 Fine automata for co-safety NBW

We now move on to consider co-safety NBWs. We start with bad news and show that a fine NFW for a co-safety NBW cannot, in general, have the same structure as the co-safety NBW. We then present our main result and show that a fine NFW for a co-safety property can be constructed from the NBWs for the property and its negation. Note that by dualizing this result, we get a similar bound also for safety NBWs.

NBWs are not fine-type The notion of *typeness* arises in the context of translations between different types of automata on infinite words [KPB94,KMM04]. For an acceptance condition γ (say, Büchi), an automaton $\mathcal{A}$ is said to be γ-type if whenever there is a γ-automaton equivalent to $\mathcal{A}$, there is also a γ-automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ with the same structure as $\mathcal{A}$. Thus, $\mathcal{A}'$ is obtained from $\mathcal{A}$ by redefining its acceptance condition. It is shown, for example, in [KPB94] that deterministic Rabin automata are Büchi type: if a deterministic Rabin automaton $\mathcal{A}$ recognizes a language that can be recognized by a deterministic Büchi automaton, then $\mathcal{A}$ has an equivalent deterministic Büchi automaton on the same structure. On the other hand, Streett automata are not Büchi type: there is a deterministic Streett automaton $\mathcal{A}$ that recognizes a language that can be recognized by a deterministic Büchi automaton, but all the possibilities of defining a Büchi acceptance condition on the structure of $\mathcal{A}$ result in an automaton recognizing a different language.

For a co-safety NBW $\mathcal{A}$, we say that $\mathcal{A}$ is *fine-type* if a fine automaton for $\mathcal{A}$ can be defined on the structure of $\mathcal{A}$. It is shown in [KV01a] that the NBWs for many co-safety LTL formulas are fine-type: by taking the NBW $\mathcal{A}_\psi$ for ψ and defining only accepting sinks to be accepting[2], one gets an NFW fine for ψ. Intuitively, each state of $\mathcal{A}_\psi$ is associated with

[2] The details in [KV01a] are for alternating automata, and the argument refers to the NBW obtained by translating these automata to nondeterministic ones via the construction of [MH84].

a set S of subformulas of ψ. A word w is accepted by $\mathcal{A}_\psi$ from state S iff w satisfies all the formulas in S. For natural LTL formulas and for constructions of $\mathcal{A}_\psi$ (c.f., [GPVW95]) that keep in S only formulas that are essential for the satisfaction, the set S would become empty after reading some prefix of a word that satisfies ψ. Unfortunately, this is not true for all formulas. The reason for this is the fact that known constructions for LTL proceed according to the syntax of the formulas, and the co-safetyness of a formula may hide. We demonstrate this in the examples below, which also show that NBWs are not fine type.

The NBW $\mathcal{A}_\varphi$ in Figure 1 is the union of two NBWs. The NBW on the left accepts all words over the alphabet $\{a, b\}$ that satisfy $Fa \wedge GFb$ ("eventually a and infinitely many b's"). The NBW on the right accepts all words satisfying $Fb \wedge GFa$. While each of these languages is neither safe nor co-safe, their union is the co-safety language of all words satisfying $Fa \wedge Fb$ ("eventually a and eventually b"). The formula $\varphi = (Fa \wedge GFb) \vee (Fb \wedge GFa)$ is *pathologically* co-safe [KV01a], which means intuitively that it is hard to tell that it is co-safe just from its syntax: a computation that satisfies φ has no *informative prefix* [KV01a] — a prefix in which all the syntactic eventualities in the formulas are satisfied. Indeed, only the combination of the two NBWs in the union reveals the co-safetyness of φ.

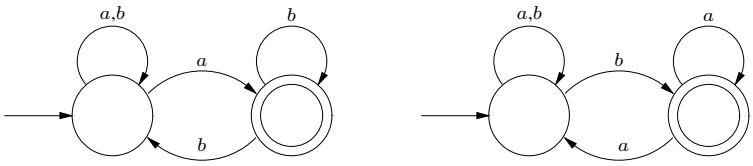

Fig. 1. An NBW for $\varphi = (Fa \wedge GFb) \vee (Fb \wedge GFa)$.

The above analysis is reflected in the fact that the NBW $\mathcal{A}_\varphi$ is not fine type; i.e., there is no way to define a fine NFW on its structure, by just redefining the accepting states. To see this, observe that every state in $\mathcal{A}_\varphi$ can be reached after reading a prefix of length at most 1. Since every such prefix can be extended to one of the infinite words a^ω or b^ω, which do not satisfy $Fa \wedge Fb$, an NFW with the structure of $\mathcal{A}_\varphi$ is either empty or accepts words that are not good prefixes for φ.

The above example refers to general co-safety NBWs. In Appendix A, we describe two stronger examples, in the sense that they show the non-fine-typeness of restricted classes of NBWs — classes that correspond to the NBWs obtained by translating LTL formulas to NBWs. The first class of NBWs we consider is *single run* NBWs; i.e., every word that is accepted by the NBW has a single accepting run. The NBWs whose construction is described in [VW94] are single run. The second class we consider is of NBWs obtained by applying the Miyano-Hayashi procedure for alternation removal [MH84] on top of the alternating Büchi automaton obtained from the LTL formula [KVW00].

A construction of fine NFWs While general co-safety NBWs are not fine-type, we can still construct an NFW fine for a co-safety NBW $\mathcal{A}$ and whose size depends on the sizes

of $\mathcal{A}$ and *comp*$(\mathcal{A})$. The idea is that it is possible to bound the number of times that a run of $\mathcal{A}$ visits the set of accepting states, when it runs on a word not in $\mathcal{L}(\mathcal{A})$. Formally, we have the following:

Lemma 1. *Consider a co-safety NBW $\mathcal{A}$. Let F be the set of accepting states of $\mathcal{A}$ and let $\overline{\mathcal{A}}$ be an NBW with $\overline{n}$ states such that $\mathcal{L}(\overline{\mathcal{A}}) =$ comp$(\mathcal{L}(\mathcal{A}))$. If a run of $\mathcal{A}$ on a finite word $h \in \Sigma^*$ visits F more than $|F| \cdot \overline{n}$ times, then h is a good prefix for $\mathcal{L}(\mathcal{A})$.*

Proof: Since $\mathcal{A}$ is a co-safety NBW, $\overline{\mathcal{A}}$ is a safety NBW. Recall that no state of $\overline{\mathcal{A}}$ is empty. Therefore, by [Sis94], every infinite run of $\overline{\mathcal{A}}$ is accepting[3]. Let $r = r_0 r_1 \ldots r_l$ be a run of $\mathcal{A}$ on h that visits F more than $|F| \cdot \overline{n}$ times. Assume, by way of contradiction, that h is not a good prefix. Then, h can be extended to a word accepted by $\overline{\mathcal{A}}$, and thus, there is a (finite) run $r' = r'_0 r'_1 \ldots r'_l$ of $\overline{\mathcal{A}}$ on h. Since r visits F more than $|F| \cdot \overline{n}$ times, there exist $0 \leq i < j \leq l$ such that $r_j = r_i \in F$ and $r'_j = r'_i$. Let $w = h_1 \ldots h_i (h_{i+1} \ldots h_j)^\omega$. Since $r_j = r_i \in F$, the run $r_0 r_1 \ldots r_i (r_{i+1} \ldots r_j)^\omega$ is an accepting run of $\mathcal{A}$ on w. On the other hand, $r'_0 r'_1 \ldots r'_i (r'_{i+1} \ldots r'_j)^\omega$ is an accepting run of $\overline{\mathcal{A}}$ on w. Hence, w is accepted by both $\mathcal{A}$ and $\overline{\mathcal{A}}$, and we have reached a contradiction. □

Theorem 4. *Consider a co-safety NBW with n states, m of them accepting. Let $\overline{\mathcal{A}}$ be an NBW with $\overline{n}$ states such that $\mathcal{L}(\overline{\mathcal{A}}) =$ comp$(\mathcal{L}(\mathcal{A}))$. There exists an NFW $\mathcal{A}'$ with $n \cdot (m \cdot \overline{n} + 1)$ states such that $\mathcal{A}'$ is fine for $\mathcal{L}(\mathcal{A})$.*

Proof: Let $t = m \cdot \overline{n}$. The NFW $\mathcal{A}'$ consists of $t+1$ copies of $\mathcal{A}$. The transition function is such that when a run of $\mathcal{A}'$ visits F in the j-th copy of $\mathcal{A}$, it moves to the $(j+1)$-th copy. The accepting states of $\mathcal{A}'$ are the states of F in the $(t+1)$-th copy. Thus, there is a run of $\mathcal{A}$ on an infinite word $w \in \Sigma^\omega$ that has $t+1$ visits in F iff there is a run of $\mathcal{A}'$ on w that reaches an accepting state in the $t+1$-th copy of $\mathcal{A}$.

Formally, given $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, F \rangle$, we define $\mathcal{A}' = \langle \Sigma, Q', \delta', Q'_0, F' \rangle$ as follows.

- $Q' = Q \times \{0, 1, \ldots, t\}$.
- For every $q \in Q$, $a \in \Sigma$, and $0 \leq i \leq t$ the transition function δ' is defined as follows.

$$\delta'(\langle q, i \rangle, a) = \begin{cases} \delta(q,a) \times \{i+1\} & \text{if } q \in F \text{ and } i < t, \\ \delta(q,a) \times \{i\} & \text{if } q \notin F. \end{cases}$$

- $Q'_0 = Q_0 \times \{0\}$.
- $F' = F \times \{t\}$. Note that there are no transitions from states in F'.

Clearly, the number of states in $\mathcal{A}'$ is $n \cdot (t+1)$.

For every run $r = r_0 r_1 \ldots$ of $\mathcal{A}$ on an infinite word $w \in \Sigma^\omega$ we define a single (possibly finite) corresponding run $s = s_0 s_1 \ldots$ of $\mathcal{A}'$ on w such that for all $i \geq 0$, we have that $s_i = \langle r_i, j_i \rangle$ for some $0 \leq j_i \leq t$, and in addition, the following hold.

1. $j_0 = 0$.
2. If $r_i \in F$, then $j_{i+1} = j_i + 1$; otherwise, $j_{i+1} = j_i$.

[3] Note that $\overline{\mathcal{A}}$ is equivalent to the nondeterministic word automaton obtained by making all its states accepting. Such automata are termed *looping*.

3. If $s_i = \langle r_i, t \rangle$ and $r_i \in F$, then the run s ends at s_i.

In order to prove that $\mathcal{A}'$ is fine for $\mathcal{L}(\mathcal{A})$, we first prove the following proposition, relating the runs r and s.

Proposition 1. *For every $i \geq 0$ and state $s_i = \langle r_i, j_i \rangle$ in s, the prefix $r_0 \ldots r_{i-1}$ of r has j_i visits in F.*

Proof: The proof proceeds by an induction on i. For $i = 0$, the prefix $r_0 \ldots r_{0-1}$ is empty, so it does not visit F at all, and indeed, by Condition 1 in the definition of s, we have that $j_0 = 0$. For $i > 0$, the induction hypothesis for $i-1$ implies that $r_0 \ldots r_{i-2}$ has j_{i-1} visits in F. If $r_{i-1} \in F$, then, by Condition 2 of the definition of s, we have that $j_i = j_{i-1} + 1$, and indeed $r_0 \ldots r_{i-1}$ has $j_{i-1} + 1$ visits in F. Otherwise, $r_{i-1} \notin F$, and, by Condition 2 of the definition of s, we have that $j_i = j_{i-1}$, and indeed $r_0 \ldots r_{i-1}$ has j_{i-1} visits in F. □

We can now prove that $\mathcal{A}'$ is fine for $\mathcal{L}(\mathcal{A})$. That is, we prove that for every $w \in \Sigma^\omega$, it holds that $w \in \mathcal{L}(\mathcal{A})$ if and only if $\mathcal{A}'$ accepts some prefix of w.

Assume first that $w \in \mathcal{L}(\mathcal{A})$. Then, there exists a run r of $\mathcal{A}$ on w that visits F infinitely many times. By Proposition 1, when r makes its $(t+1)$-th visit to F, the corresponding run of $\mathcal{A}'$ on w visits a state in F'. Thus, $\mathcal{A}'$ accepts a prefix of w, and we are done.

For the other direction, assume that there is an accepting run $s = s_0 s_1 \ldots s_k$ of $\mathcal{A}'$ on a prefix of w. The run s ends in some state in F'. Therefore, by Proposition 1, since $s_k \in F \times \{t\}$, the run s corresponds to a prefix $r_0 \ldots r_k$ of a run r of $\mathcal{A}$ on w such that $r_0 \ldots r_{k-1}$ has t visits in F. In addition, since $r_k \in F$, we have that $r_0 \ldots r_k$ has $t+1$ visits in F. Thus, by Lemma 1, $w \in \mathcal{L}(\mathcal{A})$, and we are done. □

Given a safety NBW, its complement NBW is co-safety. Thus, dualizing Theorem 4, we get the following.

Theorem 5. *Consider a safety NBW with n states. Let $\overline{\mathcal{A}}$ be an NBW with $\overline{n}$ states, $\bar{m}$ of them accepting, such that $\mathcal{L}(\overline{\mathcal{A}}) = comp(\mathcal{L}(\mathcal{A}))$. There exists an NFW $\mathcal{A}'$ with $\overline{n} \cdot (\bar{m} \cdot n + 1)$ states such that $\mathcal{A}'$ is fine for $\mathcal{L}(\mathcal{A})$.*

4.3 Fine automata for safety and co-safety LTL formulas

By Theorem 1, given an LTL formula ψ, we can construct NBWs $\mathcal{A}_\psi$ and $\mathcal{A}_{\neg\psi}$ for ψ and $\neg\psi$, respectively. The number of states in each of the NBWs is at most $2^{O(|\psi|)}$. Hence, by Theorem 4, we can conclude:

Theorem 6. *Consider a safety LTL formula φ of length n. There exists an NFW fine for φ with at most $2^{O(n)}$ states.*

5 Discussion

We have answered to the positive the question about the existence of exponential fine automata for general safety LTL formulas. This improves the doubly-exponential construction in [KV01a]. Essentially, our construction adds a counter on top of the NBW for the formula. The counter is increased whenever the NBW visits an accepting state, and a computation is accepted after the counter reaches a bound that depends on the size of the formula.

Our results give a better understanding of the relationship between safety and *bounded* properties. A property ψ is bounded if there is a bound $k \geq 0$ such that every word of length k is either a good or a bad prefix for ψ. Thus, satisfaction of ψ can be determined after reading a prefix of length k of the computation. It is known that a property ψ is bounded iff ψ is both safety and co-safety [KV01b]. For a bounded property with bound k, we know that if a word of length k is not a bad prefix, then it must be a good prefix. Accordingly, if the NBW $\mathcal{A}_\psi$ does not get stuck during its run on a prefix of a computation of length k, the computation satisfies ψ. Moreover, k depends on the size of the NBW for ψ [KV01b]. This enables simple application of bounded model-checking procedures [CBRZ01] for the verification of bounded properties. For a co-safety property, there is no such bound k: while we know that a computation that satisfies ψ has a good prefix, we cannot point to a k such that if the NBW $\mathcal{A}_\psi$ does not get stuck during its run on the prefix of a computation of length k, then the computation satisfies ψ. Our results here show that co-safety properties (and hence, also reasoning about safety properties) do have a bounded nature, only that the bound depends not only on the length of the prefix, but also on the number of visits to the set of accepting states that the NBW $\mathcal{A}_\psi$ makes on its run on the prefix. Indeed, there is a bound k such that if the NBW $\mathcal{A}_\psi$ has a run on a computation and the run visits the set of accepting states k times, then the computation satisfies ψ. Interestingly, the bound on k is similar to the one known for bounded properties [KV01b].

This result is helpful in the context of run-time verification and bounded model checking. Run-time verification does not store the entire state space of the system. Instead, it observes finite executions. In [GH01], the authors describe a semantics for LTL formula with respect to finite words. In this semantics, eventualities have to be satisfied within the finite prefix. Thus, as with the "informative prefixes" of [KV01a], a prefix never satisfies a pathologically safe formula ψ, even if it is a good prefix. By counting visits to the set F of accepting state of the NBW for $\neg\psi$, the semantics can be made tighter, and a prefix accepted by the fine automaton can be declared as violating ψ.

Counting visits to F can help also in SAT-based bounded model checking. Recall that SAT-based model checking of a safety property ψ tries to find a path that satisfies $\neg\psi$ and uses a bounded semantics for LTL: the formula $\neg\psi$ is checked with respect to prefixes of some bounded length k, possibly with a loop back. The method is complete in the sense that if some computation satisfies $\neg\psi$, then there is some prefix as above, where k depends on both the size of the checked system and ψ [CBRZ01]. The need to specify the fact that the prefix may have a loop back makes the formula whose satisfaction we check much more complex, and complicates the verification procedure. Our results imply an alternative approach, which prevents the need to consider loops and suggests, instead, to count sufficiently many visits to the set F of accepting state of the NBW for $\neg\psi$. We note that for many co-safety formulas, the states in F are accepting sinks, thus while k is

a bound for the number of transitions needed to reach F for the first time, we can expect successive visits to be made within a single transition.

Finally, while we solved the problem of constructing exponential fine automata for LTL formulas, the problem of constructing polynomial fine automata for co-safety NBW is still open. The challenge here is similar to other challenges in automata-theoretic constructions in which one needs both the NBW and its complementing NBW — something that is easy to have in the context of LTL, but difficult in the context of NBW. For a discussion of more problems in this status, see [KV05b]. From a practical point of view, however, the problem of going from a co-safety automaton to a fine NFW is of less interest, as users that use automata as their specification formalism are likely to start with an automaton for the bad or the good prefixes anyway. Thus, the problem about the size of fine automata is interesting mainly for the specification formalism of LTL, which we did solve.

Acknowledgment We thank Moshe Vardi for helpful discussions.

References

[BCM+92] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.

[BM83] R.S. Boyer and J.S. Moore. Proof-checking, theorem-proving and program verification. Technical Report 35, Institute for Computing Science and Computer Applications, University of Texas at Austin, January 1983.

[CBRZ01] E. M. Clarke, A. Bierea, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.

[CVWY92] C. Courcoubetis, M.Y. Vardi, P. Wolper, and M. Yannakakis. Memory efficient algorithms for the verification of temporal properties. *Formal Methods in System Design*, 1:275–288, 1992.

[GH01] D. Giannakopoulou and K. Havelund. Automata-based verification of temporal properties on running programs. In *Proc. 16th International Conference on Automated Software Engineering*, pages 412–416. IEEE Computer Society, 2001.

[GPVW95] R. Gerth, D. Peled, M.Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In P. Dembiski and M. Sredniawa, editors, *Protocol Specification, Testing, and Verification*, pages 3–18. Chapman & Hall, August 1995.

[GW91] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. In *Proc. 3rd Conference on Computer Aided Verification*, volume 575 of *Lecture Notes in Computer Science*, pages 332–342, Aalborg, July 1991. Springer-Verlag.

[HKSV97] R.H. Hardin, R.P. Kurshan, S.K. Shukla, and M.Y. Vardi. A new heuristic for bad cycle detection using BDDs. In *Computer Aided Verification, Proc. 9th International Conference*, volume 1254 of *Lecture Notes in Computer Science*, pages 268–278. Springer-Verlag, 1997.

[IN97] H. Iwashita and T. Nakata. Forward model checking techniques oriented to buggy designs. In *Proc. IEEE/ACM International Conference on Computer Aided Design*, pages 400–404, 1997.

[KMM04] O. Kupferman, G. Morgenstern, and A. Murano. Typeness for ω-regular automata. In *2nd International Symposium on Automated Technology for Verification and Analysis*, volume 3299 of *Lecture Notes in Computer Science*, pages 324–338. Springer-Verlag, 2004.

[KPB94] S.C. Krishnan, A. Puri, and R.K. Brayton. Deterministic ω-automata vis-a-vis deterministic Büchi automata. In *Algorithms and Computations*, volume 834 of *Lecture Notes in Computer Science*, pages 378–386. Springer-Verlag, 1994.

[KSV96] O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. In *Proc. 11th IEEE Symp. on Logic in Computer Science*, pages 322–333, DIMACS, June 1996.

[KV01a] O. Kupferman and M.Y. Vardi. Model checking of safety properties. *Formal methods in System Design*, 19(3):291–314, November 2001.

[KV01b] O. Kupferman and M.Y. Vardi. On bounded specifications. In *Proc. 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 2250 of *Lecture Notes in Computer Science*, pages 24–38. Springer-Verlag, 2001.

[KV05a] O. Kupferman and M.Y. Vardi. From linear time to branching time. *ACM Trans. on Computational Logic*, 6(2):273–294, April 2005.

[KV05b] O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, Pittsburgh, October 2005.

[KVW00] O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, March 2000.

[Lan69] L.H. Landweber. Decision problems for ω–automata. *Mathematical Systems Theory*, 3:376–384, 1969.

[Lat03] T. Latvala. Efficient model checking of safety properties. In *Proc. 10th SPIN Workshop on Model Checking of Software*, volume 2648 of *Lecture Notes in Computer Science*, pages 74–88, 2003.

[LP85] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc. 12th ACM Symp. on Principles of Programming Languages*, pages 97–107, New Orleans, January 1985.

[MAB+94] Z. Manna, A. Anuchitanukul, N. Bjorner, A. Browne, E. Chang, M. Colon, L. De Alfaro, H. Devarajan, H. Sipma, and T. Uribe. STeP: The Stanford Temporal Prover. Technical Report STAN-CS-TR-94-1518, Dept. of Computer Science, Stanford University, 1994.

[McM92] K.L. McMillan. Using unfolding to avoid the state explosion problem in the verification of asynchronous circuits. In *Proc. 4th Conference on Computer Aided Verification*, volume 663 of *Lecture Notes in Computer Science*, pages 164–174, Montreal, June 1992. Springer-Verlag.

[MF71] A.R. Meyer and M.J. Fischer. Economy of description by automata, grammars, and formal systems. In *Proc. 12th IEEE Symp. on Switching and Automata Theory*, pages 188–191, 1971.

[MH84] S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, Berlin, January 1992.

[MP95] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Safety*. Springer-Verlag, New York, 1995.

[MR97] S. Melzer and S. Roemer. Deadlock checking using net unfoldings. In *Computer Aided Verification, Proc. 9th International Conference*, volume 1254 of *Lecture Notes in Computer Science*, pages 364–375. Springer-Verlag, 1997.

[OSR95] S. Owre, R.E. Shankar, and J.M. Rushby. *User guide for the PVS specification and verification system*. CSL, 1995.

[Pnu81] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

[SC85] A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logic. *Journal ACM*, 32:733–749, 1985.

[Sis94] A.P. Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6:495–511, 1994.

[SS78] W. Sakoda and M. Sipser. Non-determinism and the size of two-way automata. In *Proc. 10th ACM Symp. on Theory of Computing*, pages 275–286, 1978.

[TBK95] H.J. Touati, R.K. Brayton, and R. Kurshan. Testing language containment for ω-automata using BDD's. *Information and Computation*, 118(1):101–109, April 1995.

[Val93] A. Valmari. On-the-fly verification with stubborn sets. In *Proc. 5nd Conference on Computer Aided Verification*, volume 697 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[VW86] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st Symp. on Logic in Computer Science*, pages 332–344, Cambridge, June 1986.

[VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.

A NBWs for LTL formulas are not fine-type

The NBW $\mathcal{A}_\theta$ in Figure 2 consists of two NBWs too. The left one accepts all words satisfying $Fa \wedge FGb$ and the right one accepts words satisfying $Fb \wedge GFa$. Thus, $\mathcal{A}_\theta$, their union, accepts all words satisfying $\theta = (Fa \wedge FGb) \vee (Fb \wedge GFa)$. It is not hard to see that $\mathcal{A}_\theta$ is a single-run automaton (in particular, it is the union of two disjoint languages) that accepts exactly the words satisfying the formula $Fa \wedge Fb$. Also, by the same considerations we had in Section 4.2 for $\mathcal{A}_\varphi$, for $\varphi = (Fa \wedge GFb) \vee (Fb \wedge GFa)$, it is not fine-type.

We note that the single-run NBW obtained for φ by following the translation procedure in [VW94] is not fine-type either.

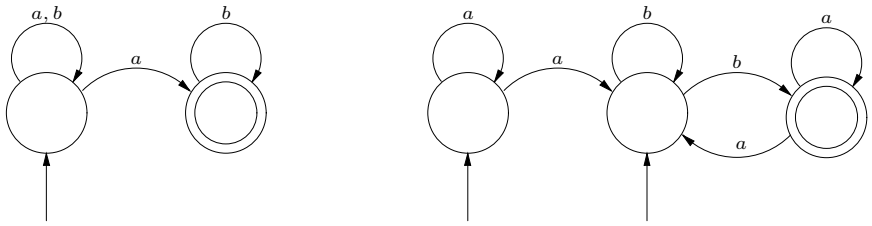

Fig. 2. A single run NBW for $\theta = (Fa \wedge FGb) \vee (Fb \wedge GFa)$.

In Figure 3, we describe the NBW $\mathcal{A}_\xi$ obtained for the formula $\xi = (GFa \wedge F(b \wedge XFb)) \vee (GFb \wedge F(a \wedge XFa))$. The formula ξ is equivalent to the formula $F(b \wedge XFb) \wedge F(a \wedge XFa)$. Thus, the language of $\mathcal{A}_\xi$ is the co-safety language of all infinite words that contain at least two a's and at least two b's. The NBW $\mathcal{A}_\xi$ is obtained from ξ by translating ξ to an alternating Büchi word automaton (ABW) and translating this automaton to a nondeterministic one [MH84]. As such, each state in $\mathcal{A}_\xi$ corresponds to a pair $\langle S, O \rangle$ of sets of subformulas of ξ. The set S is obtained by following the subset construction on the

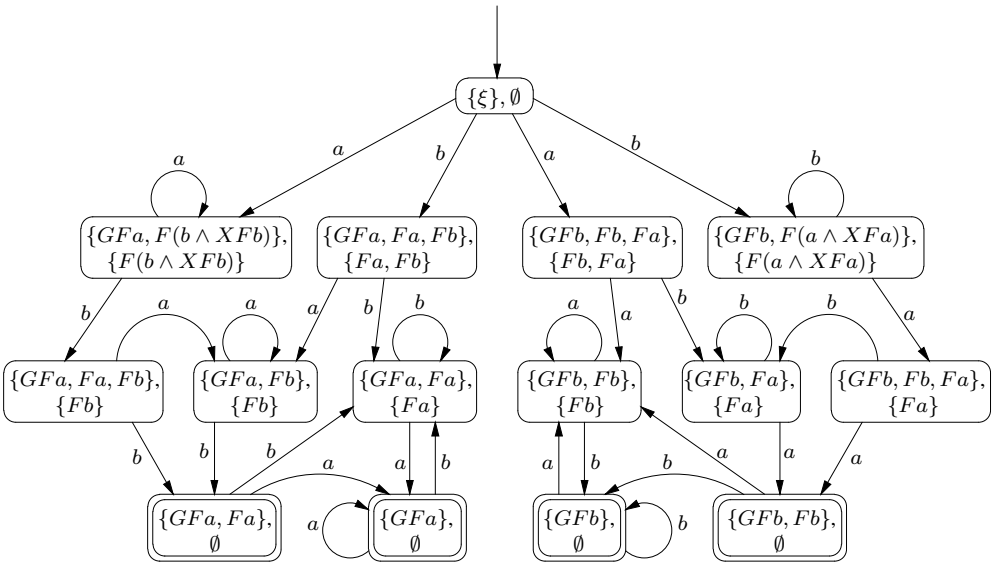

Fig. 3. An NBW for $\xi = (GFa \wedge F(b \wedge XFb)) \vee (GFb \wedge F(a \wedge XFa))$.

ABW, and the set O is a subset of S describing these formulas that have not yet fulfilled their eventualities (in Figure 3, we describe S at the top and O at the bottom of each state).

In order to see that the NBW $\mathcal{A}_\xi$ is not fine-type, observe that each one of its states can be reached after reading a prefix of length at most 3. Since every such prefix can be extended to an infinite word in which a or b appear at most once, and thus does not satisfy ξ, an NFW with the structure of $\mathcal{A}_\xi$ is either empty or accepts words that are not good prefixes for ξ.