An Automata-Theoretic Dynamic Completeness Criterion for Bounded Model-Checking

Rotem Oshman

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Abstract. Bounded model-checking is a technique for finding bugs in very large designs. Bounded model-checking by itself is incomplete: it can find bugs, but it cannot prove that a system satisfies a specification. A *dynamic completeness criterion* can allow bounded model-checking to prove properties. A dynamic completeness criterion typically searches for a "beginning" of a bug or bad behavior; if no such "beginning" can be found, we can conclude that no bug exists, and bounded model-checking can terminate. Dynamic completeness criteria have been suggested for several temporal logics, but most are tied to a specific bounded model-checking encoding, and the ones that are not are based on nondeterministic Büchi automata. In this paper we develop a theoretic framework for dynamic completeness criteria based on alternating Büchi automata. Our criterion generalizes and explains several existing dynamic completeness criteria, and is suitable for both linear-time and universal branching-time logic. We show that using alternating automata rather than nondeterministic automata can lead to much smaller completeness thresholds.

1 Introduction

Bounded model-checking (BMC) is a model-checking method that has gained popularity due to its ability to handle large industrial designs [4],[5]. Bounded model-checking is an iterative process in which one searches for a bug of increasing bounded length. In each iteration, one searches for a bug of size k, by constructing a Boolean formula which is satisfiable iff such a bug exists. A SAT solver is then used to determine whether or not the formula is satisfiable. If it is, then a bug has been found; otherwise, one increases the bound k and searches for a bug of greater size.

There are many BMC encodings for various fragments of linear-time logic and automata on words; e.g., [4] for LTL, [10] for PLTL, [11] for weak alternating Büchi automata. Many are based, directly or indirectly, on the idea of constructing a product automaton $M \times A$ for the model M in question and an automaton A which describes all the undesirable behaviors. Any accepting run of the product automaton $M \times A$ corresponds to a bad behavior of the model; thus, to check if the model contains a bug, we can search for an accepting run of the product automaton. Using automata as specification mechanisms can lead to simple and generic encodings. Even encodings based on temporal logic (e.g., [10] and [12]) can often be viewed as simulating the run of the product automaton, although they do not construct it directly.

Bounded model-checking is typically a semi-decision procedure: it is able to find bugs, but not to prove the correctness of properties. A *completeness threshold* is an upper bound on the size of k, such that if no bug has been found when the bound reaches k, then no bug exists. Thus, if a completeness threshold for the property and model in question is known, bounded model-checking can halt if the completeness threshold is reached and no bug was found, and conclude that the model satisfies the property.

Completeness thresholds can be broadly divided into two classes, although the division is not clear-cut. *Static* completeness thresholds ([5], [6]) attempt to over-approximate the size of the "longest shortest bug" the system can contain. For example, if a model does not satisfy an invariant p, then there exists a shortest path from an initial state of the model to a state that does not satisfy p. A static completeness threshold for invariant properties is therefore given by the length of the longest shortest path in the model (the *diameter*).

In contrast, *dynamic* completeness thresholds are based on a *dynamic completeness criterion*, which attempts to determine whether the current bound is already large enough to allow full exploration of the relevant part of the model. Dynamic completeness criteria typically check for the existence of a "beginning" of a counter-example (or bug). If such a beginning of size k cannot be found, then there cannot exist a counter-example of size greater than k, and there is no need to increase the bound. For example, the LTL property $\varphi = pUq$ describes a path in which q holds at some point, and until that point, p holds. Suppose φ describes the bad behaviors of a system. A dynamic completeness criterion for φ might check if there exists a simple (loop-free) path of length k, such that all states along the path are labeled with p. Such a path represents a "beginning" of a witness for φ. If we cannot find a witness of length k for φ, and we cannot find a simple path of length k as described above, then there cannot exist a witness of length greater than k for φ. Therefore, in this case bounded model-checking can terminate and conclude that the system contains no path that satisfies φ.

The effectiveness of dynamic completeness criteria has been shown in experimental results ([10], [22], [24]). However, designing completeness criteria that are both sound and effective can be challenging. For instance, the completeness criterion in [22] contains a subtle flaw: a constraint introduced to cause earlier termination and increase the effectiveness of the criterion causes the criterion to be unsound. For details, see [17]. In addition, existing completeness criteria are often custom-designed to fit one particular encoding. For example, the dynamic completeness criteria of [22], [18] and [24] are all based on ideas similar to the ones on which the current paper is based, but they each develop the completeness criterion anew to fit the particular encoding.

In this work we present an automata-theoretic dynamic completeness criterion for alternating Büchi automata. Our criterion generalizes several existing completeness criteria by formalizing the notion of a "beginning" of an accepting computation. The criterion we suggest is independent of a particular encoding; in addition to serving as a theoretical framework for existing completeness criteria, it can be instantiated to fit automata-based BMC encodings for which there is currently no dynamic completeness criterion, such as [11].

To our knowledge, the criterion we suggest is the first completeness criterion that can handle alternating automata. The choice of alternating Büchi automata as a specification mechanism is motivated by two factors. First, alternating Büchi automata are powerful enough to express all ω-regular properties. [11] developed an encoding for weak alternating Büchi automata, and showed that the increased expression power did not carry significant performance penalties.

The second factor is the succintness of alternating automata. It is well-known that an alternating automaton can be exponentially smaller than any equivalent nondeterministic automaton [20]. In this paper we show another compelling reason to use alternating automata as a specification mechanism: they can have much smaller completeness thresholds than the corresponding nondeterministic automata. This result is related to, but does not follow directly from, the exponential gap in the number of states.

In addition to linear-time logic, several BMC encodings and accompanying completeness criteria have been suggested for universal branching-time logic ([19], [23], [22], [18]). Our completeness criterion is based on automata on infinite words, which express linear-time properties. However, the criterion is also applicable to universal automata on infinite trees, which express universal branching-time properties. This is because our criterion is based on the product of the model and the automaton. The product of a model and a alternating automaton on infinite trees is an alternating automaton on infinite words [14]; thus, our dynamic completeness criterion, which is based on alternating automata on infinite words, is applicable to branching-time logic as well. Note, however, that in a branching-time setting, a Büchi acceptance condition is not expressive enough to express all ω-regular tree properties. Our criterion can therefore only handle the alternation-free fragment of universal μ-calculus.

The rest of the paper is organized as follows. In Section 3 we review the automata-theoretic approach to linear-time logic and define notation and terminology. In Section 4 we present the dynamic completeness criterion and the resulting completeness threshold. We show that the criterion is sound, and characterize its completeness. In Section 5 we show that there is an exponential ratio between the completeness thresholds of alternating and nondeterministic automata. We conclude in Section 6.

2 Related Work

In the original work on BMC ([4], [5]), the diameter of the model is suggested as a completeness threshold for formulas of the form EFp ("p is reachable"). [5] also shows a pessimistic completeness threshold of $|M| \times 2^{|\varphi|}$ for general LTL formulas φ. In [6], tighter completeness thresholds are shown for various classes of temporal properties, among them the class of all ω-regular properties, based on automata-theoretic methods. The completeness threshold suggested in [6] for general ω-regular properties is an over-approximation of the length of the shortest lasso-shaped accepting run of the product automaton. Our own work is based on similar ideas; however, the automata we consider are alternating, while [6] bases its threshold on nondeterministic automata. As we show in Section 5, using nondeterministic automata as a specification mechanism can increase the completeness threshold significantly. In addition, the completeness threshold of [6] does not take the form of a *dynamic completeness criterion* which is evaluated at different bounds to determine whether or not the completeness threshold has been reached. In [2], the authors apply similar ideas to [6], this time in a form closer to a dynamic completeness criterion: to check whether the completeness threshold has been reached, one can check the satisfiability of several Boolean formulas, which roughly speaking describe the existence of loop-free fragments of an accepting run in the product automaton. Unlike our own work, the completeness criteria of [2] check for the existence of both a "beginning" and an "ending" of an accepting run (forward and backward traversal). However, [2] is still restricted to nondeterministic automata. In [3], the authors of [2] extend their termination criterion to *generalized* nondeterministic Büchi automata, in which the acceptance criterion can consist of several accepting sets, and show that using generalized automata can lead to smaller completeness thresholds. The completeness threshold we suggest in this paper is easily extended to generalized Büchi automata.

In [10], an incremental encoding is presented for PLTL, along with a dynamic completeness criterion based on the idea of searching for a "beginning" of a witness. In an incremental scheme, the encoding is composed of two parts – a k-invariant part, containing constraints that are retained when the bound is increased, and a k-dependent part containing constraints that are discarded when the bound is increased. The formula used in [10] to determine whether the completeness threshold has been reached is obtained from the formula used to search for a witness by removing the k-dependent constraints and adding a simple-path constraint. Removing the k-dependent constraints has the effect of dropping eventuality requirements (e.g., when searching for a witness for Fp, the requirement that p be satisfied at some point along the path is a k-dependent constraint). The completeness formula of [10] is highly specific to the incremental scheme and the particular encoding used in [10]. Our completeness criterion can be extended to handle temporal logic with past operators by extending it to two-way automata on words [21].

Several bounded model-checking encodings have been suggested for universal branching-time temporal logic [18], [19], [22], [23]. [22] and [18] show accompanying dynamic completeness criteria for their respective encodings, and a dynamic completeness criterion for the encoding of [19] is presented in [24]. The criteria of [22], [18] and [24] are again highly encoding-specific, and all use a similar idea of searching for a "beginning" of a witness.

A related SAT-based technique which can prove properties is temporal induction [7], which can prove invariants. General safety and liveness properties can be transformed into invariants, but such translations increase the size of the model and may increase the depth necessary for bounded model-checking.

Our work is also closely related to [8], which discusses extensions of LTL that can be used to reason about *truncated* paths. Our notion of a partial run corresponds to the weak semantics of LTL for truncated paths described in [8], and can be taken as an automata-theoretic formulation of the weak semantics. We are interested in investigating this connection.

3 Preliminaries

Given a set X, we denote by $\mathcal{B}^+(X)$ the set of positive Boolean formulas obtained by applying the connectives $\wedge$ (conjunction) and $\vee$ (disjunction) to elements of X, as well as the formulas **true** and **false**. We say that a subset $Y \subseteq X$ *satisfies* a formula $\alpha \in \mathcal{B}^+(X)$, and denote $Y \models \alpha$, if the assignment v_Y, defined by $v_Y(x) = 1$ if $x \in Y$ and $v_Y(x) = 0$ if $x \notin Y$, satisfies the formula α.

An *alternating Büchi automaton on infinite words* is a tuple $A = (\Sigma, Q, q_0, \delta, F)$, where Σ is the automaton's alphabet, Q is the set of automaton states, $q_0 \in Q$ is the initial state of the automaton, $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a transition relation, and $F \subseteq Q$ is the set of *accepting* (or *fair*) states. A *nondeterministic* automaton is an alternating automaton which has only disjunctions in all its transitions. We use Σ^ω to denote the set of infinite words over the alphabet Σ, and we use x^ω to denote the infinite word obtained by iterating the finite word x infinitely often.

To model the runs of A we use Q-trees. A *Q-tree* is a pair $t = (N, \ell)$, where $N \subseteq \mathbb{N}^*$ is a prefix-closed set of tree nodes, and $\ell : N \to Q$ labels each node of the tree with an automaton state. The root of the tree is the empty word ε, and given a node $n \in N$, the set of children of n in the tree is given by $\mathit{children}(n) = \{n' \mid n' = n \cdot i \text{ for some } i \in \mathbb{N}\}$. We denote by $|n|$ the length of the finite word n, and for an infinite word n we denote $|n| = \omega$. For a tree node n, the length $|n|$ is also the distance of n from the root of the tree (ε). A *branch* of the tree is a maximal sequence $n_0 n_1 n_2 \ldots$ (which can be either finite or infinite), such that $n_0 = \varepsilon$, and for all $i \geq 0$, $n_{i+1} \in \mathit{children}(n_i)$. If $t = (N, \ell)$ is a finite tree, the *front* of t is defined by $\mathrm{front}(t) = \{n \in N \mid \mathit{children}(n) = \emptyset\}$, and the *height* of t is the length of the longest branch in t. Note that we measure height by the number of edges, not the number of nodes.

A *run* (or *run-tree*) of an automaton A on an infinite word $w = w_0 w_1 w_2 \ldots \in \Sigma^\omega$ is a Q-tree $r = (N, \ell)$, such that for all $n \in N$, $\mathit{children}(n) \models \delta(\ell(n), w_{|n|})$. We say that a run r is *accepting* if for every branch $n_0 n_1 n_2 \ldots$ of r, some accepting state $q \in F$ appears infinitely often on the branch (that is, $\ell(n_i) = q$ for infinitely many values of i). If A has some accepting run on a word w, we say that A *accepts* w. The language of A, denoted $\mathcal{L}(A)$, is the set of words $w \in \Sigma^\omega$ such that A accepts w. Note that runs can be finite or infinite trees, and even in an infinite run there can be finite branches. However, finite branches must always end in a node n such that $\delta(n, w_{|n|}) = \textbf{true}$.

To model programs, we use Kripke structures. Given a set AP of atomic propositions, a *Kripke structure* (or *model*) over AP is a tuple $M = (S, s_0, R, L)$, where S is the state-space of the model, $s_0 \in S$ is the initial state, $R \subseteq S \times S$ is a transition relation, and $L : S \to 2^{AP}$ is a labeling function which assigns to each model state a set of atomic propositions from AP. A *path* of M is a maximal sequence $\pi = s_0 s_1 s_2 \ldots$ starting at s_0, such that for all $i \geq 0$, $(s_i, s_{i+1}) \in R$. The *labeling* of a path $\pi = s_0 s_1 s_2 \ldots$ is the word $L(\pi) = L(s_0) L(s_1) L(s_2) \ldots$.

Two parameters are often used to measure the complexity of a Kripke structure. The *diameter* d_M of a structure M is the length of the longest shortest path in M. The *recurrence diameter* r_M is the length of the longest loop-free path in M. The diameter of a model is no greater than its recurrence diameter, since a shortest path is always loop-free, but the recurrence diameter is easier to compute than the diameter.

We say that a model $M = (S, s_0, R, L)$ *satisfies* an (existentially-interpreted) automaton $A = (2^{AP}, Q, q_0, \delta, F)$, and denote $M \models A$, if there is a path π of M such that $L(\pi) \in \mathcal{L}(A)$. The path π is then called a *witness*. In bounded model-checking (and linear-time model-checking in general), the automaton A describes the bad behaviors of the system, and a witness, if one exists, represents a bug in the system.

One way to check if $M \models A$ is to construct the *product* of M and A [20], an alternating automaton which, informally, describes all the runs that A can have on paths of M. The product automaton is defined by $M \times A = (\{a\}, S \times Q, (s_0, q_0), \delta_M, S \times F)$. The transition relation δ_M of the product automaton is defined by $\delta_M((s,q),a) = \bigvee_{s' : (s,s') \in R} \alpha_{q,s,s'}$, where $\alpha_{q,s,s'}$ is the formula obtained from $\delta(q, L(s))$ by replacing every atom $q' \in Q$ with (s', q'). (For example, if $\delta(q, \{b\}) = q_1 \wedge q_2$, $L(s) = \{b\}$, and the only transitions from s are to s_1 and to s_2, then $\delta_M((s,q),a) = ((s_1,q_1) \wedge (s_1,q_2)) \vee ((s_2,q_1) \wedge (s_2,q_2))$.)

It can be shown that $M \models A$ iff $\mathcal{L}(M \times A) \neq \emptyset$. Therefore, to check if $M \models A$, we can construct $M \times A$ and check whether or not its language is empty [20]. Note that the product automaton $M \times A$ is over a *unary* alphabet $\{a\}$, and thus, if $\mathcal{L}(M \times A) \neq \emptyset$, then $M \times A$ must accept the word a^ω.

Throughout the paper we will be interested in prefixes of words and trees. We denote $x \prec y$ if x is a prefix of y. Given a finite or infinite word $x = x_0 x_1 \ldots$ and a number $h \leq |x|$, we denote $\mathrm{pref}_h(x) = x_0 \ldots x_h$. We use $\mathrm{pref}(x)$ (witout the subscript) to denote the set of prefixes of x, that is, $\mathrm{pref}(x) = \{\mathrm{pref}_h(x) \mid 0 \leq h \leq |x|\}$. Similarly, given a finite or infinite tree $t = (N, \ell)$, we use $\mathrm{pref}_h(t)$ to denote the tree $\mathrm{pref}_h(t) = (N_h, \ell_h)$ defined by $N_h = \{n \in N \mid |n| \leq h\}$ and $\ell_h(n) = \ell(n)$ for all $n \in N_h$. We also denote $\mathrm{pref}(t) = \{\mathrm{pref}_h(t) \mid h \in \mathbb{N}\}$. Finally, for an automaton A, we denote by $\mathrm{pref}_h(A) = \{\mathrm{pref}_h(r) | r \text{ is a run of } A\}$ the set of all prefixes of height h of runs of A. For a model M, we denote $\mathrm{pref}_h(M) = \{\mathrm{pref}_h(\pi) \mid \pi \text{ is a path of } M\}$.

4 A Dynamic Completeness Criterion for Alternating Automata

In this section we define a dynamic completeness criterion, which checks whether the automaton has a "beginning" of an accepting computation of length k. If there is no such "beginning", bounded model-checking can terminate and return $M \not\models A$ when the bound reaches k.

To formalize the notion of a "beginning" of an accepting computation, we define *canonical partial runs* of the automaton. Informally, a *partial run* is a truncated run-tree; it is a finite tree in which only the inner nodes are required to satisfy the transition relation. A *canonical* partial run is a partial run that contains no "useless" loops. We will later discuss another restriction on partial runs that may lead to smaller completeness thresholds.

Fix an alternating Büchi automaton on infinite words $A = (2^{AP}, Q, q_0, \delta, F)$.

Definition 1. *A* partial run of height h *of* A *on a word* $w = w_0 w_1 \ldots$ *is a* Q*-tree* $r = (N, \ell)$ *of height* h *satisfying the following conditions.*

1. *$\ell(\varepsilon) = q_0$.*
2. *For all $n \in N$ such that $|n| < h$, children$(n) \models \delta(\ell(n), w_{|n|})$.*

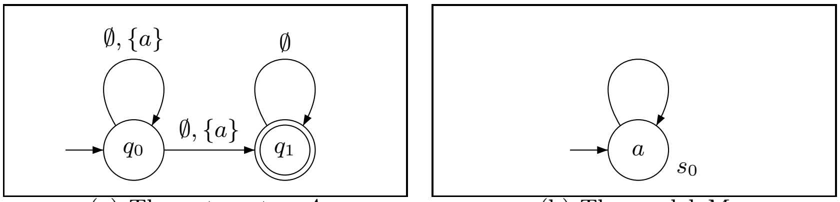
The definition of a partial run of height h is distinguished from the definition of an accepting run of the same height by the requirement on the leaves (nodes n with $|n| = h$): in an accepting run, if n is a leaf then $\delta(n, w_{|n|}) = \mathbf{true}$; in a partial run, there is no such requirement. Consequently, every accepting run is also a partial run, but the converse is not true.

Lemma 1. *If r is an accepting run on w, then for all $h \in \mathbb{N}$, $\mathrm{pref}_h(r)$ is a partial run of height h on w.*

Clearly, if A has an accepting run of height h or greater on a word w, then A has a partial run of height h' on w for all $h' \le h$. However, there may exist partial runs of height h for all $h \in \mathbb{N}$ even if there is no accepting run, as in the following example.

Example 1. Consider the nondeterministic automaton A shown in Fig. 1(a). A accepts the language of words over the alphabet $\Sigma = \{\emptyset, \{a\}\}$ which contain a finite number of occurrences of the letter $\{a\}$. A run of A stays in q_0 until A "guesses" it has seen the last $\{a\}$, and then moves to q_1, which is an accepting state; if an $\{a\}$ is read from state q_1, the run gets stuck (that is, the transition is **false**).

The model M shown in Fig. 1(b) is not accepted by the automaton: the only path in M is $\pi = (s_0)^\omega$, whose labeling, $L(\pi) = (\{a\})^\omega$, contains infinitely many occurrences of $\{a\}$. However, for all $h \in \mathbb{N}$, the sequence $r_h = q_0^h$ represents a partial run of height h of A on π.



(a) The automaton A (b) The model M

Fig. 1. The automaton and model from Example 1

Notice that for $h > 2$, r_h is not a "good" partial run on π: after one step we enter a loop in both r and π in which no accepting state of A appears. Thus, r_h for $h > 1$ contains unnecessary padding which increases its height. Intuitively, a "good" partial run will contain no unnecessary loops. We now formalize this notion.

Definition 2. *Given a partial run $r = (N, \ell)$ of height h,*

1. *A* loop *in r is a pair $(n_1, n_2) \in N^2$ such that $n_1 \prec n_2$ and $\ell(n_1) = \ell(n_2)$.*
2. *We say that an automaton state $q \in Q$* occurs in (n_1, n_2) *if there is a node $n \in N$ such that $n_1 \prec n \prec n_2$ and $\ell(n) = q$.*

3. *A loop (n_1, n_2) is said to be* useless *if no state $q \in F$ occurs in (n_1, n_2).*
4. *We say that r is* canonical *if there are no useless loops in r.*

Lemma 2. *If an automaton A over a unary alphabet $\{a\}$ accepts the word a^ω, then A has a canonical accepting run on a^ω.*

Proof sketch. The existence of a canonical accepting run follows from the existence of a memoryless winning strategy, shown in [9] for alternating parity automata, a more general class of automata than alternating Büchi automata. The accepting run that corresponds to a memoryless winning strategy for the word a^ω has no useless loops: if it had a useless loop, then the run-tree would contain a branch along which the loop repeats forever, because each time the automaton would reach each state in the loop it would be obliged by the memoryless strategy to make the same move. Since the loop contains no accepting state, this branch would contain only finitely many accepting states, and the run would not be accepting. □

The definition of a canonical partial run captures the notion of a "beginning" of a counter-example used in the dynamic completeness criteria of, e.g., [2], [22], [24]. The threshold we suggest is as follows.

Definition 3. *Given an automaton A and a model M, the* dynamic completeness threshold $\mathrm{CT}(M, A)$ *is the minimal number h such that $M \times A$ does not have a canonical partial run of height h (on a^ω), or ∞ if there is no such number.*

Next we show that the dynamic completeness threshold is sound — that is, it does not cause termination too early.

Theorem 1. *If $M \models A$, but for all $h' < h$ the product automaton $M \times A$ has no accepting run of height h', then* $\mathrm{CT}(M, A) \geq h$.

Proof. From Lemma 2, since $M \models A$, the product automaton $M \times A$ has a canonical accepting run r on a^ω. The run r is of height at least h, because $M \times A$ has no accepting runs of height smaller than h. Let $r' = \mathrm{pref}_h(r)$. From Lemma 1, r' is a partial run, and since r has no useless loops, neither does r'. Thus, r' is a canonical partial run of height h of $M \times A$, and $\mathrm{CT}(M, A) > h$.

Remark 1. Consider the case of a nondeterministic automaton A. A canonical partial run of A is a sequence of states — that is, a tree with a single branch. If the run contains a loop, then an accepting state must appear in the loop, implying the existence of an infinite accepting run of A.

Thus, for a nondeterministic automaton, the existence of a canonical partial run of height k indicates the existence of either an accepting run or a loop-free run of height k. However, the dynamic completeness criterion is only applied after we *fail* to find an accepting run at the current bound k; thus, we can rule out the first case, and simply search for a loop-free run of length k. This yields the length of the longest loop-free path in the product automaton as an upper bound on the completeness threshold of nondeterministic automata, as already observed in [6]. This is also the basis for the forward-traversal termination criteria of [2].

Table 1. The transition of the automaton from Example 2

q	$\delta(q,a)$	$\delta(q,b)$	$\delta(q,c)$
q_0	$q_0 \wedge q_a$	$q_0 \wedge q_b$	q_0
q_a	q_a	q_a	**true**
q_b	q_b	q_b	**false**

4.1 Eliminating Bad Prefixes

As a consequence of Theorem 1, if no accepting run of $M \times A$ has been found when the bound reaches $\mathrm{CT}(M, A)$, then bounded model-checking can halt and return $M \not\models A$. Conversely, we would expect that while $k < \mathrm{CT}(M, A)$ — that is, if there exists a canonical partial run of height k — then there should be "a reason" to increase the bound and search for an accepting run of height greater than k. However, the following example shows that this is not always the case.

Example 2. Consider the automaton $A = (\{a, b, c\}, \{q_0, q_a, q_b\}, q_0, \delta, \{q_0, q_b\})$, where the transition relation δ is given in Table 1. A accepts only words that do not contain both an a and a b: when an a is read, A moves to state q_a, where it waits to read a c; and when a b is read, A moves to state q_b, in which it must *not* read a c. Thus, any word in which both an a and a b appear will not be accepted. However, A still has a canonical partial run r of height 1 on any word w which has $\mathrm{pref}_1(w) = ab$. The partial run r cannot be extended into an accepting run regardless of the rest of w, but its existence may cause BMC not to terminate with a bound of 1 if the termination criterion from Definition 3 is used.

A prefix which cannot be extended into a word in the language, like ab in Example 2, is called a *bad prefix* [8]. One way to avoid bad prefixes is to construct a *prefix automaton* A_{pref} for A — an automaton which accepts a finite word iff it can be extended into an infinite word in the language of A — and to use it in the completeness criterion. This option is suitable for criteria based on nondeterministic automata (e.g., [2]), where a prefix automaton is constructed by simply removing any states from which there is no accepting run on any word and making all remaining states accepting. For alternating automata, however, this option is not suitable: [1] shows that for alternating automata the size of a prefix automaton can be exponential in the size of the original automaton. Next we present an alternative, which does not require the use of the prefix automaton in the dynamic criterion.

Definition 4. *Given an automaton $A = (\Sigma, Q, q_0, \delta, F)$, a set of automaton states $Q' \subseteq Q$ is said to be A-*consistent *if $\bigcap_{q \in Q'} \mathcal{L}(A_q) \neq \emptyset$, where the automaton A_q is defined by $A_q = (\Sigma, Q, q, \delta, F)$ (with q replacing q_0 as initial state).*

Now we augment the definition of canonical partial runs as follows.

Definition 5. *A partial run r of $M \times A$ is said to be* prefix-canonical *if r is canonical and the set $R = \{q \in Q \mid$ there exists $s \in S$ such that $(s, q) \in \mathrm{front}(r)\}$ is A-consistent.*

To compute the sets of A-consistent (or A-inconsistent) sets in an automaton A, we can build an equivalent nondeterministic automaton A' using the Miyano-Hayashi construction [16], which is a modified subset construction, and check which states of A' have some accepting computation. We do not go into the details of the construction for lack of space. The construction is exponential in the size of A, but it involves only the automaton and not the model. In addition, identifying the A-consistent sets is a preprocessing step which only needs to be performed once per automaton, and does not need to be repeated in every iteration of BMC.

The completeness threshold can be strengthened by adding the new requirement, and defining the *prefix completeness threshold* $\mathrm{CT}_P(M, A)$ to be the smallest number h such that $M \times A$ does not have a prefix-canonical partial run of height h. It is easy to show the equivalent of Theorem 1 for $\mathrm{CT}_P(M, A)$. In addition, we can now also show the following lemma and corollary, which give us a a reason why bounded model-checking should continue if the threshold has not been reached.

Lemma 3. *If A has a prefix-canonical partial run on a finite word $w \in \Sigma^h$, then there exists an infinite word $w' \in \mathcal{L}(A)$ such that $\mathrm{pref}_h(w') = w$.*

Corollary 1. *If $M \not\models A$ and $\mathrm{CT}_P(M, A) \geq h$, then there exists a model M' such that $\mathrm{pref}_h(M \times A) \subseteq \mathrm{pref}_h(M' \times A)$ and $M' \models A$.*

Corollary 1 shows that as long as the threshold has not been reached, there is a model M' which does satisfy A, such that any computation of $M \times A$ is also a computation of $M' \times A$. Informally, the fragment of M that A "can see" to depth h also exists in M', but $M' \models A$, and so we cannot stop searching for a witness just yet.

4.2 The Limitations of Existential Dynamic Completeness Criteria

It might seem better to require, instead of Corollary 1, that if $\mathrm{CT}_P(M, A) \geq h$ then there should exist a model M' such that $\mathrm{pref}_h(M \times A) = \mathrm{pref}_h(M' \times A)$ (equality instead of containment) and $M' \models A$. This stronger requirement, if satisfied, would imply that the completeness threshold uses all the information that is available in $\mathrm{pref}_h(M)$: if the threshold has not been reached, then there is a model which is *indistinguishable* from M to a depth of h as far as A is concerned, which A accepts. However, the following example shows that no sound completeness criterion of the form "there exists a path $\pi \in \mathrm{pref}_h(M)$ which satisfies ψ", where ψ is some condition on paths of length h, can satisfy the stronger requirement.

Example 3. Consider the model M shown in Fig. 2(b) , and the nondeterministic automaton A shown in Fig. 2(a). A accepts paths in which there is exactly one state labeled with b, and all the states before that state are labeled with a. $M \not\models A$, since all infinite paths of M contain at least two states labeled with b.

Suppose $\mathcal{C}$ is a dynamic completeness threshold of the form: "$\mathcal{C}(M, A)$ is the smallest number h such that there does not exist a path $\pi \in \mathrm{pref}_h(M)$ satisfying the condition ψ", where ψ is some condition on paths of length h. Suppose by way of contradiction that $\mathcal{C}$ also satisfies the following two conditions:

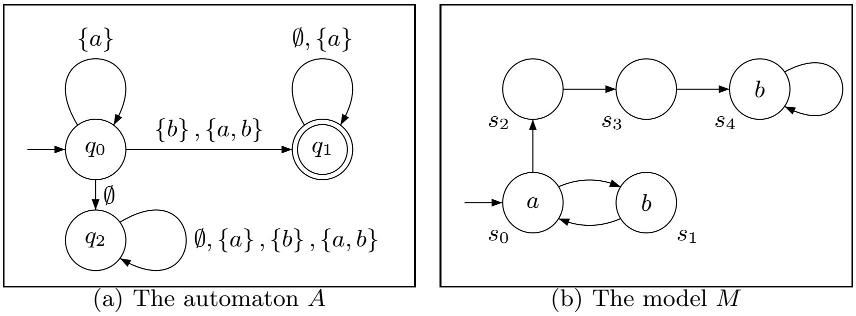


Fig. 2. The automaton and model from Example 3

1. (Soundness) If $M \models A$, but $M \times A$ has no accepting run of height $h' < h$, then $\mathcal{C}(M, A) \geq h$,
2. If $\mathcal{C}(M, A) \geq h$ then there exists a model M' such that $\text{pref}_h(M \times A) = \text{pref}_h(M' \times A)$ and $M' \models A$.

In our case, $M \not\models A$. However, for $h = 2$, there exists a model M', obtained from M by changing the labeling of s_4 to $L'(s_4) = \emptyset$, such that $\text{pref}_h(M) = \text{pref}_h(M')$ and $M' \models A$. Any accepting run of $M' \times A$ must be infinite (since A has no **true** transitions). Therefore, for M', A, and $h = 2$, the terms of condition 1 are satisfied, and hence $\mathcal{C}(M', A) \geq 2$. From our assumption about the structure of $\mathcal{C}$, there must exist a path $\pi \in \text{pref}_2(M') = \text{pref}_2(M) = \{s_0 s_1 s_0, s_0 s_2 s_3\}$ which satisfies the condition ψ. We show that this contradicts condition 2.

If $\pi_1 = s_0 s_1 s_0$ satisfies the condition ψ, then the model M_1 shown in Fig. 3(a) also has $\mathcal{C}(M_1, A) \geq 2$, since $\pi_1 \in \text{pref}_2(M_1)$. Similarly, if $\pi_2 = s_0 s_2 s_3$ satisfies ψ, then the model M_2 shown in Fig. 3(b) has $\mathcal{C}(M_2, A) \geq 2$. However, for both $i = 1, 2$, there does not exist a model M_i' such that $\text{pref}_2(M_i \times A) = \text{pref}_2(M_i' \times A)$ and $M_i' \models A$. This is because in both cases, any attempt to extend M_i into a model satisfying A must add a transition from either s_0 (for M_1 or M_2) or from s_1 (for M_1), which will create a new path of length 2 in M and a new partial run in $\text{pref}_2(M_i' \times A)$. We thus have that any model M_i' such that $M_i' \models A$ and $\text{pref}_2(M_i \times A) \subseteq \text{pref}_2(M_i' \times A)$ must also have $\text{pref}_2(M_i') \not\subseteq \text{pref}_2(M_i)$, contradicting condition 2.

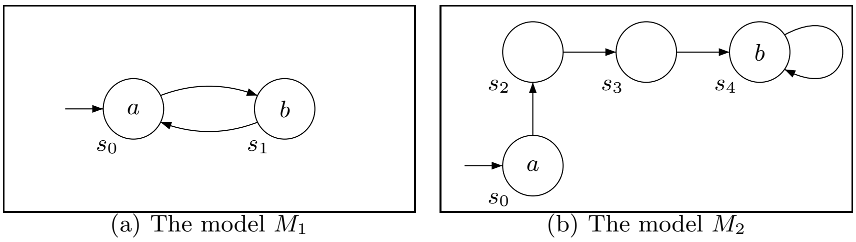


Fig. 3. The models M_1 and M_2 from Example 3

It is easy to extend Example 3 to conditions which talk about the existence of a path of arbitrary (but predetermined) length satisfying some condition ψ, not just paths of length h. The limitation we have shown is inherent to dynamic completeness thresholds based on existentially-interpreted automata.

5 The Gap between the Completeness Threshold for Alternating and Non-deterministic Automata

It is well known that although alternating automata are equivalent in expression power to nondeterministic automata, they are exponentially more succint: there exist languages recognized by an alternating automaton comprising n states, which cannot be recognized by a nondeterministic automaton that has less than 2^n states [20]. In this section we show that the ratio between the completeness thresholds for alternating and nondeterministic automata can also be exponential in the number of states of the alternating automaton.

In [16] it is shown that for any alternating automaton A with n states, there exists an equivalent nondeterministic automaton A' with $n' = 2^{2n}$ states, such that $\mathcal{L}(A) = \mathcal{L}(A')$. The following upper bound result follows in a straightforward manner, because an exponential blow-up in the number of states implies at worst an exponential blow-up in the size of loops in the run-tree.

Theorem 2. *For any alternating automaton A with n states and model M with diameter d, there is a nondeterministic automaton A' with $\mathcal{L}(A) = \mathcal{L}(A')$ such that $\mathrm{CT}(A') \leq 2^{2n} \cdot d \cdot \mathrm{CT}(A)$.*

In addition to the upper bound on the number of states, [20] shows a matching exponential lower-bound: there exists an alternating automaton with n states such that any equivalent nondeterministic automaton must have at least 2^n states. However, this does not immediately imply a corresponding lower bound on the completeness thresholds nondeterministic automata, because a large number of states does not necessarily translate to long loops in the runs of the automaton. For example, the family of languages used in [20] to show the exponential lower bound can be recognized by altnernating and nondeterministic automata that have the same completeness threshold on any given model (even though the nondeterministic automaton would require exponentially more states than the alternating automaton).

The lower bound we will show stems from the fact that in an alternating automaton we require that *each branch* of the run-tree not contain unnecessary loops. Thus, we can detect "hopeless situations" as soon as one branch cannot be extended without creating an unnecessary loop, even if other branches in the run-tree are loop-free. In contrast, the runs of a nondeterministic automaton only have one branch, and we show that this branch can be made to grow exponentially long without closing a loop. Our result is related to the ability of alternating automata to count to 2^n using only n states, while nondeterministic automata require 2^n states to accomplish the same task [13].

Theorem 3. *There is a model M over a single atomic proposition $AP = \{p\}$, and a family of languages $\{L_n\}_{n=1}^{\infty}$ over 2^{AP}, such that for all $n \geq 1$,*

1. *There is an alternating automaton A with $O(n)$ states such that $\mathcal{L}(A) = L_n$ and* $\mathrm{CT}(M, A) = 1$*, and*
2. *Any nondeterministic automaton A' such that $\mathcal{L}(A') = L_n$ must have* $\mathrm{CT}(M, A') \geq 2^n - 1$.

Proof. We will use the model $M = (\{s\}, s, \{(s,s)\}, L)$, where $L(s) = \emptyset$. For simplicity we will represent $2^{AP} = \{\emptyset, \{p\}\}$ by the binary alphabet $\Sigma = \{a, b\}$, where a stands for $\emptyset$ and b stands for $\{p\}$. We now construct the family $\{L_n\}_{n=1}^{\infty}$.

Let $n \geq 1$. Consider the finitary language $H_n = \left\{bw \mid w \in \Sigma^{2^n - 1}\right\}$, which contains words of length 2^n that start with b. H_n can be recognized by a nondeterministic automaton with 2^n states. Consequently, by a result from [15], there exists an alternating automaton $A_1 = (\{a,b\}, Q_1, q_0^1, \delta_1, F_1)$ with n states which recognizes the reverse language $H_n^R = \left\{wb \mid w \in \Sigma^{2^n - 1}\right\}$.

Let $G_n = \{uav \mid u \in \Sigma^*, v \in \Sigma^\omega\}$, and let $L_n = (H_n^R \cdot \Sigma^\omega) \cap G_n$. L_n is the language of infinite words in which the letter b appears in position 2^n and the letter a appears anywhere in the word. L_n can be recognized by an alternating automaton A_L, defined by $A_L = (\{a,b\}, Q_1 \cup \{q_0, q_a\}, q_0, \delta_L, F_1)$, where we assume w.l.o.g. that $q_0, q_a \notin Q_1$, and δ_L is defined as follows:

- For all $q \in Q_1$ and $\sigma \in \{a, b\}$, $\delta(q, \sigma) = \delta_1(q, \sigma)$. (This part of the automaton behaves exactly like A_1.)
- $\delta(q_0, a) = \delta_1(q_0^1, a)$. (When reading an a from the initial state, the obligation that a must eventually be seen is discharged immediately, and the automaton behaves like A_1 from that point onward.)
- $\delta(q_0, b) = q_a \wedge \delta_1(q_0^1, b)$. (When reading a b from the initial state, the automaton splits into two parts, one that behaves like A_1 and one that waits to see an a.)
- $\delta(q_a, a) = \mathbf{true}$ and $\delta(q_a, b) = q_a$. (In q_a, the automaton waits to see an a. Note that since $q_a \notin F_1$, a branch along which only q_a appears is not accepting. Thus, an a must appear eventually in order for the word to be accepted.)

The automaton A_L has $n + 2$ states. Let r be a partial run of height h of $M \times A$. The only path in M is $\pi = s^\omega$, labeled $L(\pi) = b^\omega$. From the definition of δ_L, the run r contains a branch of the form $(s, q_0)(s, q_a)^h$. But q_a is not an accepting state in A_L, and therefore (s, q_a) is not accepting in $M \times A_L$. Thus, if $h > 1$, then the branch $(s, q_0)(s, q_a)^h$ contains an unnecessary loop, and r is not canonical. Consequently we have that $M \times A_L$ has no canonical partial run of length 2 or greater, and $\mathrm{CT}(M, A_L) = 1$.

Now suppose that A' is a nondeterministic automaton which recognizes L_n. Then A' accepts the word $w = b^{2^n} a^\omega \in L_n$. Let $t = q_0 q_1 \ldots$ be an accepting run of A' on w, and consider the path $\pi = s^\omega$ in M. Note that $\mathrm{pref}_{2^n - 1}(L(\pi)) = \mathrm{pref}_{2^n - 1}(w) = b^{2^n}$, and hence it is easy to verify that the sequence $t_M = (s, q_0)(s, q_1) \ldots (s, q_{2^n})$ is a partial run of height $2^n - 1$ of $M \times A'$.

Suppose by way of contradiction that $\mathrm{CT}(M, A') < 2^n - 1$, that is, $M \times A'$ has no canonical partial runs of height $2^n - 1$. Then the run t_M cannot be canonical, since its height is $2^n - 1$. We therefore have that t_M contains a useless loop; in particular, there exist $i < j \leq 2^n$ such that $q_i = q_j$. It is easily shown that the run

$t' = q_0 \dots q_{i-1} q_j q_{j+1} \dots$ is an accepting run of A' on the word $w' = b^{2^n-(j-i)} a^\omega$. But because $i < j$, the letter b does not appear in position 2^n of w', and $w' \notin L_n$. This contradicts our assumption that $\mathcal{L}(A') = L_n$. □

Note that the completeness threshold of the alternating automaton from Theorem 3 *remains constant* as n grows, while the completeness threshold of the corresponding nondeterministic automaton grows exponentially with n.

6 Conclusion

We developed a dynamic completeness criterion for bounded model-checking, which we believe explains and generalizes several dynamic completeness criteria that were developed for various encodings and temporal logics. By using automata as the specification mechanism we were able to abstract away the details of the specific temporal logic and encoding, and obtain a notion of a "beginning" of a bad behavior which is applicable to the full class of ω-regular properties.

We also showed that alternating automata are better suited to serve as a basis for completeness criteria than nondeterministic automata: alternating automata can "separate concerns" and track different requirements in different branches of the run-tree, which can lead to termination as soon as we verify that at least one of the requirements cannot be fulfilled. We are interested in developing an encoding for our completeness criterion based on the encoding from [11] for weak alternating Büchi automata.

Acknowledgements. The author is grateful to Orna Grumberg, Yoram Moses and Avi Yadgar for many fruitful discussions.

References

1. Armoni, R., Bustan, D., Kupferman, O., Vardi, M.Y.: Resets vs. aborts in linear temporal logic. In: Garavel, H., Hatcliff, J. (eds.) TACAS 2003. LNCS, vol. 2619, pp. 65–80. Springer, Heidelberg (2003)
2. Awedh, M., Somenzi, F.: Proving more properties with bounded model checking. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 96–108. Springer, Heidelberg (2004)
3. Awedh, M., Somenzi, F.: Termination criteria for bounded model checking: Extensions and comparison. Electr. Notes Theor. Comput. Sci. 144(1), 51–66 (2006)
4. Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: Symbolic model checking without bdds. In: Cleaveland, W.R. (ed.) TACAS 1999. LNCS, vol. 1579, pp. 193–207. Springer, Heidelberg (1999)
5. Clarke, E., Biere, A., Raimi, R., Zhu, Y.: Bounded model checking using satisfiability solving. Form. Methods Syst. Des. 19(1), 7–34 (2001)
6. Clarke, E., Kroening, D., Strichman, O., Ouaknine, J.: Completeness and Complexity of Bounded Model Checking. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, pp. 85–96. Springer, Heidelberg (2004)
7. de Moura, L., Rueß, H., Sorea, M.: Bounded model checking and induction: From refutation to verification. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 1–13. Springer, Heidelberg (2003)

8. Eisner, C., Fisman, D., Havlicek, J., Lustig, Y., McIsaac, A., Campenhout, D.V.: Reasoning with temporal logic on truncated paths. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 27–39. Springer, Heidelberg (2003)
9. Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy. In: SFCS 1991, Washington, DC, USA, pp. 368–377. IEEE Computer Society, Los Alamitos (1991)
10. Heljanko, K., Junttila, T., Latvala, T.: Incremental and complete bounded model checking for full PLTL. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 98–111. Springer, Heidelberg (2005)
11. Heljanko, K., Junttila, T.A., Keinänen, M., Lange, M., Latvala, T.: Bounded model checking for weak alternating büchi automata. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 95–108. Springer, Heidelberg (2006)
12. Jehle, M., Johannsen, J., Lange, M., Rachinsky, N.: Bounded model checking for all regular properties. In: BMC 2005. Electr. Notes in Theor. Comp. Sc., vol. 144, pp. 3–18. Elsevier, Amsterdam (2005)
13. Kupferman, O., Ta-shma, A., Vardi, M.Y.: Counting with automata. Technical report (1999)
14. Kupferman, O., Vardi, M.Y., Wolper, P.: An automata-theoretic approach to branching-time model checking. J. ACM 47(2), 312–360 (2000)
15. Leiss, E.L.: Succint representation of regular languages by boolean automata. Theor. Comput. Sci. 13, 323–330 (1981)
16. Miyano, S., Hayashi, T.: Alternating finite automata on omega-words. In: CAAP 1984, pp. 195–210 (1984)
17. Oshman, R.: Bounded model-checking for universal branching-time logic. Master's thesis, Technion – Israel Institute of Technology (August 2008)
18. Oshman, R., Grumberg, O.: A new approach to bounded model checking for branching time logics. In: Namjoshi, K.S., Yoneda, T., Higashino, T., Okamura, Y. (eds.) ATVA 2007. LNCS, vol. 4762, pp. 410–424. Springer, Heidelberg (2007)
19. Penczek, W., Wozna, B., Zbrzezny, A.: Bounded model checking for the universal fragment of CTL. Fundam. Inf. 51(1), 135–156 (2002)
20. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. In: Moller, F., Birtwistle, G. (eds.) Logics for Concurrency. LNCS, vol. 1043, pp. 238–266. Springer, Heidelberg (1996)
21. Y. Vardi, M.: Reasoning about the past with two-way automata. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 628–641. Springer, Heidelberg (1998)
22. Wang, B.-Y.: Proving forall-mu-calculus properties with SAT-based model checking. In: Wang, F. (ed.) FORTE 2005. LNCS, vol. 3731, pp. 113–127. Springer, Heidelberg (2005)
23. Woźna, B.: Bounded Model Checking for the universal fragment of CTL*. Fundamenta Informaticae 63(1), 65–87 (2004)
24. Zhang, W.: Verification of actl properties by bounded model checking. In: Moreno Díaz, R., Pichler, F., Quesada Arencibia, A. (eds.) EUROCAST 2007. LNCS, vol. 4739, pp. 556–563. Springer, Heidelberg (2007)