Approximate Automata
for Omega-regular Languages⋆

Rayna Dimitrova[1], Bernd Finkbeiner[2], and Hazem Torfah[2]

[1] University of Leicester
[2] Saarland University

Abstract. Automata over infinite words, also known as ω-automata, play a key role in the verification and synthesis of reactive systems. The spectrum of ω-automata is defined by two characteristics: the *acceptance condition* (e.g. Büchi or parity) and the *determinism* (e.g., deterministic or nondeterministic) of an automaton. These characteristics play a crucial role in applications of automata theory. For example, certain acceptance conditions can be handled more efficiently than others by dedicated tools and algorithms. Furthermore, some applications, such as synthesis and probabilistic model checking, require that properties are represented as some type of deterministic ω-automata. However, properties cannot always be represented by automata with the desired acceptance condition and determinism.

In this paper we study the problem of approximating linear-time properties by automata in a given class. Our approximation is based on preserving the language up to a user-defined precision given in terms of the size of the finite lasso representation of infinite executions that are preserved. We study the state complexity of different types of approximating automata, and provide constructions for the approximation within different automata classes, for example, for approximating a given automaton by one with a simpler acceptance condition.

1 Introduction

The specification of linear-time properties is a key ingredient of all typical frameworks for the verification and synthesis of reactive systems. The application of both automata-theoretic and symbolic algorithms requires that specifications are translated to some kind of ω-automata. Depending on the considered problem, or on the applied methods and tools, there are often constraints on the type of the resulting automaton, that is, on its acceptance condition, and on whether it is deterministic or not. For example, while for model checking of non-stochastic systems it suffices to consider nondeterministic Büchi automata, synthesis and

probabilistic model checking require deterministic automata (e.g., deterministic parity automata). Furthermore, it is often the case that efficient specialized methods and tools exist for specific classes of automata i.e., specific acceptance conditions. For instance, efficient synthesis algorithms exist for the class GR(1) of linear-time temporal logic specifications [2], which defines properties that are expressible as deterministic parity automata with three colors.

Finding an equivalent automaton with a simpler acceptance condition is not always possible. The canonical example is the property defined by the linear-time temporal logic (LTL) formula $\Diamond \Box p$, for which no deterministic Büchi automaton exists. A more interesting example is given by the LTL formula $\varphi = (\Box \Diamond p \rightarrow \Box \Diamond q) \wedge (\Box \Diamond r \rightarrow \Box \Diamond s)$, which requires that if the proposition p holds infinitely often then the proposition q should hold infinitely often as well, and the same for the propositions r and s. Requirements of this form occur often in the synthesis of reactive systems, but the formula φ cannot be represented by a deterministic parity automaton with three colors, and cannot be transformed to a formula in the efficient class of GR(1) specifications. Moreover, automata with simpler acceptance conditions can often be larger in size than automata with more general acceptance conditions. For instance, there are languages for which deterministic Streett automata are exponentially smaller than nondeterministic Büchi automata [17].

Motivated by this, we study the problem of approximating linear-time properties (respectively ω-automata) by automata in a given class (respectively automata from a given subclass). The choice of language approximation is inspired by applications in bounded model checking [5] and bounded synthesis [9]. These methods are based on the observation that for finite-state systems, it suffices to consider lasso-shaped executions of bounded size. Our approximation exploits the same idea for the construction and transformation of automata. Furthermore, equivalent ω-regular languages share the same set ultimately-periodic words [4], and thus lasso-shaped words of bounded size provide an approximation to this set of words, one that improves when considering larger bounds on the size of lassos.

Given an ω-language L and a bound $n \in \mathbb{N}$, we consider the language L_n of the ultimately-periodic words in L representable in the form $u \cdot v^{\omega}$, and where $|u \cdot v| \leq n$. That is, the language $L_n \subseteq L$ consists of the words in L representable as *lassos of length n or smaller*. We are then interested in approximations of L that are precise with respect to the language L_n, termed *n-lasso-precise approximations*.

We study the properties of n-lasso-precise approximations across the three dimensions of the complexity of the automata for such languages: size, acceptance condition, and determinism. More precisely, we establish worst case bounds, in terms of n, on the size of automata for n-lasso-precise approximations. We also show that we can approximate a parity automaton with m colors by one with $m' < m$ colors, with at most polynomial increase in the size of the automaton. For example, considering the formula φ above, if we underapproximate the language of φ with a language that is precise with respect to the set of words

representable by lassos of length n for a fixed n, we can represent the resulting language by a safety automaton (a parity automaton with one color). Furthermore, if, for example, $n = 2$ the resulting automaton has 4 states, while the minimal deterministic parity automaton for the language of φ has 95 states and 10 colors. We also study the approximation of nondeterministic by deterministic automata, and show that the worst-case exponential blow-up in the size is unavoidable for n-lasso-precise approximations.

As another example, consider the property described by the LTL formula $(\Diamond \Box p) \wedge (\Box \Diamond q)$, where p and q are some atomic propositions. This is a conjunction of a stability property and a liveness property, which is also not expressible in the fragment GR(1). We can approximate this property by an n-lasso-precise deterministic Büchi automaton, enabling the application of efficient synthesis tools. Most importantly, unlike existing approaches, our method is not limited to approximating liveness properties by safety properties, which benefits the precision of the approximation.

The paper is structured as follows. In Section 2 we start with a short background on linear-time properties and ω-automata. In Section 3 we introduce the notion of n-lasso-precise approximation of linear-time temporal properties, and present all relevant automata constructions for these approximations. Here, we establish property-independent upper and lower bounds on the size of ω-automata for n-lasso-precise approximations, and study the overhead in terms of size incurred when approximating an automaton by one with a simpler acceptance condition. In Section 4 we show that the problem of computing lasso-precise automata of bounded size for properties given as LTL formulas is in Σ_2^P. In Section 5 we conclude our results with a discussion on our approach and its potential for the development of new verification and synthesis algorithms.

Related Work. Our definition of bounded lasso-precise approximation is motivated by bounded model checking [5], bounded synthesis [9], synthesis for bounded environments [6], and synthesis of approximate implementations [15]. We extend these ideas of focusing on small counterexamples, small implementations, or bounded-state environments, respectively, to the realm of specifications.

The structural complexity of ω–automata has been studied in [3,14], where the acceptance conditions of deterministic automata are related to their complexity. Here, on the other hand, we study complexity questions in the context of language approximations.

There is a rich body of work on the minimization of Büchi automata. Typical approaches, such as [8,12,21] are based on merging states according to simulation and bisimulation relations. In [7] the authors propose a SAT-solver based minimization method. All these approaches consider language equivalence, while in this paper we study language approximation.

Reducing the size of automata by language approximation has been studied in the context of languages over finite words. The approach in [10] fixes a bound in the number of the states of a deterministic finite automaton for a safety language, and computes an automaton within that bound that approximates the

original language. In addition to the fact that their method applies to languages over finite words, the key difference to our work is that while their goal is to optimize precision within a state budget, we approximate automata with ones with simpler acceptance conditions that guarantees a desired precision. In descriptive complexity, there is a related notion to our n-lasso precision, which is the notion of the automaticity [20] of a language which is the size of the minimal automaton that is precise for that language on words of length up to a given bound n. As automaticity is defined for finite-word languages, n-lasso precision can be seen as lifting these ideas to ω-languages.

The approximation of ω-regular properties by ones with simpler acceptance conditions has not been, to the best of our knowledge, systematically studied so far. Standard approaches, such as [19, 1], approximate liveness and other temporal properties via safety properties. In contrast, our approximation allows us to approximate temporal properties with other temporal properties that are not necessarily safety.

2   Preliminaries

Linear-time Properties and Lassos. A *linear-time property* φ over an alphabet Σ is a set of infinite words $\varphi \subseteq \Sigma^\omega$. Elements of φ are called *models* of φ. A *lasso of length n* over an alphabet Σ is a pair (u, v) of finite words $u \in \Sigma^*$ and $v \in \Sigma^+$ with $|u \cdot v| = n$ that induces the ultimately-periodic word $u \cdot v^\omega$. We call $u \cdot v$ the *base* of the lasso or ultimately-periodic word, and n the *length of the lasso*. The set $Bases(\varphi, n)$ is the set of bases of lassos of length n that induce words that are models of φ.

For a bound $n \in \mathbb{N}$, we define the language $L_n(\varphi) = \{\sigma \in \Sigma^\omega \mid \exists u \cdot v \in Bases(\varphi, n). \ \sigma = u \cdot v^\omega\}$ as the language of models of φ that can be represented by lassos of length n. We call the elements of $L_n(\varphi)$ the *n-models* of φ.

If a finite word $w \in \Sigma^*$ is a prefix of a word $\sigma \in \Sigma^* \cup \Sigma^\omega$, we write $w \preceq \sigma$. For a language $L \subseteq \Sigma^* \cup \Sigma^\omega$, we define $Prefix(L) = \{w \in \Sigma^* \mid \exists \sigma \in L : w \preceq \sigma\}$ as the set of all finite words that are prefixes of words in the language L. For a word $w = \alpha_1 \alpha_2 \ldots \alpha_n \in \Sigma^*$ we define $w(i) = \alpha_i$ for each $i \in \{1, \ldots, n\}$.

Automata Over Infinite Words. A *nondeterministic parity automaton* over an alphabet Σ is a tuple $\mathcal{A} = (Q, Q_0, \delta, \mu)$, where Q denotes a finite set of states, $Q_0 \subseteq Q$ denotes a set of initial states, $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ denotes a transition function that maps a state and an input letter to a set of states, and $\mu : Q \to C \subset \mathbb{N}$ is a coloring function with a finite set of colors C.

A *run* of $\mathcal{A} = (Q, Q_0, \delta, \mu)$ on an infinite word $\sigma = \alpha_1 \alpha_2 \cdots \in \Sigma^\omega$ is an infinite sequence $\rho = q_0 q_1 q_2 \ldots \in Q^\omega$ of states such that $q_0 \in Q_0$, and for every $i \in \mathbb{N}$ it holds that $q_{i+1} \in \delta(q_i, \alpha_{i+1})$. A run $\rho = q_0 q_1 q_2 \ldots$ is *accepting* if it satisfies the *parity condition*, which requires that the highest number occurring infinitely often in the sequence $\mu(q_0)\mu(q_1)\mu(q_2)\cdots \in C^\omega$ is even. An infinite word σ is *accepted* by an automaton $\mathcal{A}$ if there exists an accepting run of $\mathcal{A}$ on σ. The set of infinite words accepted by an automaton $\mathcal{A}$ is called its *language $L(\mathcal{A})$*.

We say that a run ρ *has size* $n \in \mathbb{N}$ if ρ is an ultimately-periodic run and n is the smallest natural number such that $\rho = \rho_1 \cdot (\rho_2)^\omega$ and $|\rho_1 \cdot \rho_2| = n$.

An automaton is *deterministic* if $|Q_0| = 1$, and for all states q and input letters α, $|\delta(q, \alpha)| \leq 1$. For a deterministic automaton we will see δ as a partial function $\delta : Q \times \Sigma \to Q$. We use the notation $\delta(q, \alpha) = \emptyset$ to denote that state q has no successor for the letter α. We define the size $|\mathcal{A}|$ of an automaton $\mathcal{A}$ to be the number of its states, i.e., $|\mathcal{A}| = |Q|$.

A parity automaton is called a *Büchi automaton* if and only if the image of μ is contained in $\{1, 2\}$, and a *safety* automaton if the image of μ is $\{0\}$. Büchi automata are denoted by (Q, Q_0, δ, F), where $F \subseteq Q$ denotes the states with the higher color. Safety automata are denoted by (Q, Q_0, δ). A run of a Büchi automaton is thus accepting, if it contains infinitely many visits to F. For safety automata, every infinite run is accepting.

We define an *automaton type* to indicate whether the automaton is deterministic or nondeteministic, and its acceptance condition. We abbreviate deterministic as D and nondeterministic as N. For the acceptance conditions we use the abbreviations P (parity) and B (Büchi). Thus, for example, DPA stands for deterministic parity automaton, while NBA stands for Nondeterministic Büchi automaton.

3 Lasso-precise Approximations of Linear-time Properties

We begin this section with a formal definition of the approximation of linear-time properties discussed in the introduction. More precisely, we introduce the notion of *lasso-precise under-* and *overapproximation* of a linear-time property φ for a given bound $n \in \mathbb{N}$, in which we underapproximate (overapproximate) φ with a linear-time property that has the same n-models as φ. That is, the approximation is precise for n-models.

3.1 Lasso-precise Approximations

Definition 1 (Lasso-precise Underapproximation). *For a bound $n \in \mathbb{N}$, we say that a linear-time property φ' is an n-lasso-precise underapproximation of a linear-time property φ, denoted $\varphi' \subseteq_n \varphi$, if $\varphi' \subseteq \varphi$ and $L_n(\varphi') = L_n(\varphi)$.*

Definition 2 (Lasso-precise Overapproximation). *For a bound $n \in \mathbb{N}$, we say that a linear-time property φ' is an n-lasso-precise overapproximation of a linear-time property φ, denoted $\varphi' \supseteq_n \varphi$, if $\varphi' \supseteq \varphi$ and $L_n(\varphi') = L_n(\varphi)$.*

In the rest of the paper we focus on underapproximations. All the results extend easily to lasso-precise overapproximations. In fact, if we have also the complement language of φ, an n-lasso-precise overapproximation of a property φ can be computed by computing an n-lasso-precise underapproximation of the complement of φ.

In the next sections we show how to construct automata for n-lasso-precise approximations of linear-time properties. For a property φ the automata will

recognize the language $L_n(\varphi)$. This language includes also all words in φ that are representable by a lasso of size $n' \leq n$, a fact that we establish with the next lemma.

Lemma 1. *For any linear-time property φ and bounds $n, n' \in \mathbb{N}$, we have that $L_n(\varphi) \subseteq L_{n'}(\varphi)$, if $n \leq n'$.*

Proof. Every lasso of length n can be unrolled to a lasso of length n' by unrolling the loop $n' - n$ times. □

3.2   The Size of Lasso-precise Automata for Linear-time Properties

Since for any φ the language $L_n(\varphi)$ is a safety language, we can always construct a deterministic safety automaton that is n-lasso-precise. In the following we provide a construction which yields a deterministic safety automaton for a language $L_n(\varphi)$, and establish a lower bound on the size of an automaton for $L_n(\varphi)$.

Theorem 1 (Safety automata for n-lasso-precise approximations). *For every linear-time property φ over an alphabet Σ and a bound $n \in \mathbb{N}$, there is a deterministic safety automaton $\mathcal{A}$ of size $O(|\Sigma|^n \cdot 2^{n \log n})$, such that $L(\mathcal{A}) \subseteq_n \varphi$.*

Idea & Construction. The automaton $\mathcal{A}$ accepts a word in two phases. The states used in the first phase are of the form $w \cdot \#^{m-1} \in (\Sigma \cup \{\#\})^n$, where w is the portion of the prefix of length n of the input word that has been read so far. In this phase, the automaton reads the prefix of length n and stores it in the automaton state. Once the whole prefix is read, it checks whether the prefix of length n is in $Bases(\varphi, n)$. If this is the case, then it transitions to the second phase, and checks if the word being read is an n-lasso, with this base.

The states in the second phase are of the form $(w, (t_1, \ldots, t_n)) \in \Sigma^n \times \{-, 1, \ldots, n\}^n$, where $w \in \Sigma^n$ is the prefix read in the first phase, and $(t_1, \ldots, t_n)$ are indices of letters in w, whose role is explained below. To check that the word is an n-lasso, the automaton has to check if for some $\ell \in \{1, \ldots, n\}$ the input word is of the form $w(1) \ldots w(\ell - 1)(w(\ell) \ldots w(n))^\omega$, that is, there is an ℓ which is a loop start position. To this end, the automaton tracks the possible loop start positions, starting with all positions, and for each new letter α it eliminates those positions that are not compatible with α. More precisely, if the automaton reads a letter α in state $(w, (t_1, \ldots, t_n))$, it uses each t_i to check whether the loop can start in position i of w. Intuitively, t_i is a position in w that points to the letter that has to be read next in order for i to still be a possible loop start position. If the next letter α is not the same as $w(t_i)$, then i cannot be a loop start position, and t_i is eliminated by replacing it by $-$. Otherwise, t_i is incremented, or set back to the loop start i if the end of w is reached. A run of $\mathcal{A}$ is accepting if it never reaches a state $(w, (-, \ldots, -))$, that is, a state in which each position is no longer a possible start of a loop. A run of $\mathcal{A}$ is accepting if it never reaches a state $(w, (-, \ldots, -))$, that is, a state in which each position is no longer a possible start of a loop.

Formally, the states of the automaton are given by $\mathcal{A} = (Q, \{q_0\}, \delta)$ where:

- $Q = Q_1 \cup Q_2$, where $Q_1 = (\Sigma \cup \{\#\})^n$ and $Q_2 = \Sigma^n \times \{-, 1, \ldots, n\}^n$
- In the initial state no letter has been read: $q_0 = \#^n$.
- The transition relation δ is defined as follows.
  - In the first phase if we are at a state $q = w \cdot \#^m$ for some $1 < m \leq n$ and $w \in \Sigma^{n-m}$, then

    $$\delta(q, \alpha) = w \cdot \alpha \cdot \#^{m-1}$$

  - In the transition between the first and the second phase, which happens once the prefix of length n has been read, and when we are at a state $q = w \cdot \#$ for some $w \in \Sigma^{n-1}$ the transition is given by

    $$\delta(q, \alpha) = (w \cdot \alpha, (t_1, \ldots, t_n))$$

    where

    $$t_i = \begin{cases} i & w(1) \ldots (w(i) \ldots w(n))^\omega \in \varphi \\ - & \text{otherwise} \end{cases}$$

    Note that determining the successor state in this case requires checking if a given word is in φ. Initially, only loop start positions i for which $w(1) \ldots (w(i) \ldots w(n))^\omega \in \varphi$ are allowed, so the second phase starts with state $(w, (t_1, \ldots, t_n))$, in which each pointer t_i points to the start of the corresponding loop if $w(1) \ldots (w(i) \ldots w(n))^\omega \in \varphi$, and is set to $-$ otherwise.
  - In the second phase, for a state $q = (w, (t_1, \ldots, t_n))$ with $w \in \Sigma^n$ and where there exists $i \leq n$ with $t_i \neq -$, the transition for such a state is given by

    $$\delta(q, \alpha) = (w, (t'_1, \ldots, t'_n))$$

    where

    $$t'_i = \begin{cases} - & t_i = - \\ & \text{or } w(t_i) \neq \alpha \\ t_i + 1 & t_i < n \ \wedge \ w(t_i) = \alpha \\ i & t_i = n \ \wedge \ w(t_i) = \alpha \end{cases}$$

    Here we track valid loop start position as follows. If $\alpha \neq w(t_i)$, then the loop start i is eliminated by replacing t_i by $-$. Otherwise, we move the pointer one step to the right by incrementing t_i. In case t_i is equal to n, i.e., at the end of the lasso, t_i is reset to the corresponding loop start position i.
  - If only $-$ remain in the tuple $(t_1, \ldots, t_n)$, the automaton rejects

    $$\delta((w, (-, \ldots, -)), \alpha) = \emptyset$$

  for any $w \in \Sigma^n$.

The number of states in Q_1 is $(|\Sigma|+1)^n$, and for Q_2 it is $|\Sigma|^n \cdot (n+1)^n$. $\square$

The number of states of the deterministic safety automaton defined above is exponential in the parameter n on the length of the lassos for which the approximation should be precise. In the next theorem we exhibit a family of linear-time properties for which this exponent is unavoidable, that is, the minimal n-lasso-precise NPA has size exponential in n.

Theorem 2. *There is a family of linear-time properties φ_n for $n \in \mathbb{N}$ over an alphabet Σ, such that, every parity automaton that is n-lasso-precise for φ_n has at least $|\Sigma|^n$ states.*

Proof. Let Σ be an alphabet. We define $\varphi_n = \{\sigma^\omega \mid \sigma \in \Sigma^n\}$ for $n \in \mathbb{N}$. We show that the family φ_n of linear-time properties has the required properties.

Fix $n \in \mathbb{N}$, and consider the language φ_n. By definition of φ_n, every lasso-precise automaton for φ_n for the bound n is in fact an automaton for φ_n. Let $\mathcal{A} = (Q, Q_0, \delta, \mu)$ be a nondeterministic parity automaton for φ_n. For each $\sigma^\omega \in \varphi_n$ there exists at least one accepting run $\rho = q_0 q_1 q_2, \ldots$ of $\mathcal{A}$ on σ^ω. We denote with $q(\rho, n)$ the state q_n that appears at the position indexed n of a run ρ. Let us define the set

$$Q_n = \{q(\rho, n) \mid \exists \sigma^\omega \in \varphi_n : \rho \text{ is an accepting run of } \mathcal{A} \text{ on } \sigma^\omega\}.$$

That is, Q_n consists of the states that appear at position n on some accepting run on some word from φ_n. We will show that $|Q_n| \geq |\Sigma|^n$.

Assume that this does not hold, that is, $|Q_n| < |\Sigma|^n$. Since $|\varphi_n| = |\Sigma|^n$, this implies that there exist $\sigma_1, \sigma_2 \in \Sigma^n$, such that $\sigma_1 \neq \sigma_2$ and there exists accepting runs ρ_1 and ρ_2 of $\mathcal{A}$ on σ_1^ω and σ_2^ω respectively, such that $q(\rho_1, n) = q(\rho_2, n)$. That is, since we assumed that the number of states in Q_n is smaller than the number of words in φ_n, there must be two different words who have accepting runs visiting the same state at position n. We now construct a run $\rho_{1,2}$ that follows ρ_1 for the first n steps, ending in state $q(\rho_1, n)$, and from there on follows ρ_2. It is easy to see that $\rho_{1,2}$ is a run on the word $\sigma_1 \cdot \sigma_2^\omega$. It is accepting, since ρ_2 is accepting. This is a contradiction, since $\sigma_1 \cdot \sigma_2^\omega \notin L(\mathcal{A})$ as $\sigma_1 \neq \sigma_2$.

Thus, we have shown that $|Q| \geq |Q_n| \geq |\Sigma|^n$. Since $\mathcal{A}$ was an arbitrary NPA for φ_n, this implies that the minimal NPA for φ_n has at least $|\Sigma|^n$ states. $\square$

In the theorems above we established an upper and a lower bound on the size of automata for n-lasso-precise approximations. These bounds are independent of the way the original language is represented. If a language L is given as an ω-automaton, this automaton is clearly an automaton for the most precise n-lasso-precise underapproximation of L. In practice, however, we might be interested in finding a smaller/minimal automaton of the same type for an n-lasso-precise approximation of L. Note that the minimal n-lasso-precise automaton of the same type will never be larger than the given automaton.

3.3   Lasso-precise Approximations with Simpler Acceptance Conditions

We now turn to establishing the upper bounds for approximating Büchi automata with safety automata, and, more generally, approximating parity automata with parity automata with fewer colors. More precisely, we present constructions for approximating linear-time properties with automata with certain acceptance conditions and show that the size of the constructed automaton is polynomial in the size of an automaton for the original property.

Theorem 3 (Approximating Büchi automata by safety automata). *For every (deterministic or nondeterministic) Büchi automaton $\mathcal{A} = (Q, Q_0, \delta, F)$ and a bound $n \in \mathbb{N}$, there is a (deterministic or nondeterministic, respectively) safety automaton $\mathcal{A}'$ with $n \cdot |Q \setminus F|^2 + |F|$ states, such that, $L(\mathcal{A}') \subseteq_n L(\mathcal{A})$.*

Idea & Construction. We construct a safety automaton $\mathcal{A}'$ using the following idea: If an ultimately-periodic word $\sigma = u \cdot v^\omega$ with $|u \cdot v| = n$ is accepted by a Büchi automaton $\mathcal{A} = (Q, Q_0, \delta, F)$, then $\mathcal{A}$ has a run for σ, where it takes no more than $n \cdot |Q \setminus F|$ steps to observe a state in F, and, furthermore F is visited at least once every $n \cdot |Q \setminus F|$ steps. In the automaton $\mathcal{A}'$, we keep track of the number of steps without seeing an accepting state, and reset the counter every time we visit one. If the counter exceeds $n \cdot |Q \setminus F|$, then $\mathcal{A}'$ rejects.

Formally, we define $\mathcal{A}' = (Q', Q_0', \delta')$ as follows:

- $Q' = ((Q \setminus F) \times \{1, \ldots, n \cdot |Q \setminus F|\}) \cup (F \times \{0\})$
- $Q_0' = (Q_0 \cap F) \times \{0\} \cup (Q_0 \setminus F) \times \{1\}$
- For the transition relation we distinguish two cases. For $c < n \cdot |Q \setminus F|$

$$\delta((q, c), \alpha) = \{(q', d) \mid q' \in \delta(q, \alpha),\ d = 0 \text{ if } q' \in F,\ d = c + 1 \text{ if } q' \notin F\}$$

otherwise $\delta((q, c), \alpha) = \emptyset$.

Note that, if the given Büchi automaton is deterministic, then our construction also produces a deterministic safety automaton.   □

Theorems 1 and 3 provide safety automata of different sizes: the safety automaton obtained by Theorem 1 is exponential in the bound, the safety automaton obtained by Theorem 3 is linear in the bound. The reason for this difference is that the size of the automaton constructed according to Theorem 1 is independent of the linear-time property, whereas the size of the automaton constructed according to Theorem 3 is for a specific linear-time property (given as a Büchi automaton, whose size enters as a quadratic factor). The following theorem shows that a further reduction, below the linear number of states in the bound, is impossible.

Theorem 4. *There is a linear-time property φ, such that, for every bound $n \in \mathbb{N}$, every safety n-lasso-precise automaton for φ has at least n states.*

Proof. Let $\Sigma = \{0, 1\}$. We define φ as the language over Σ that consists of all words where the letter 1 occurs infinitely often. Let $\mathcal{A} = (Q, Q_0, \delta)$ be a safety n-lasso-precise automaton for φ. We consider the set $Q' \subseteq Q$ of states on the first n positions of an accepting run of the word $(0^{n-1}1)^\omega$. We show that $|Q'| = n$ and, therefore, $|Q| \geq n$.

Assume that this does not hold, i.e., $|Q'| < n$; then some state q must appear on two different positions among the first n positions of the run. By repeating the part of the run between the two occurrences of q infinitely often, we obtain an accepting run for the word 0^ω, which contradicts our assumption that $\mathcal{A}$ is n-lasso-precise for φ. ☐

With a construction similar to Theorem 3, we can approximate a parity automaton with $m + 1$ colors by a parity automaton with m colors.

Theorem 5 (Approximating parity automata by parity automata with one color less). *For every deterministic parity automaton $\mathcal{A} = (Q, Q_0, \delta, \mu)$ with $m + 1$ colors and a bound $n \in \mathbb{N}$, there is a deterministic parity automaton $\mathcal{A}'$ with m colors and $n \cdot |Q \setminus F|^2 + |F|$ states, where F is the set of states with highest color, such that $L(\mathcal{A}') \subseteq_n L(\mathcal{A})$.*

By iteratively applying Theorem 5, we can approximate any parity automaton with m colors by a corresponding parity automaton with $m' < m$ colors. This, however, will incur a blow-up in the size of the automaton that is exponential in the number m of colors. We now provide a direct construction, which is polynomial both in m and in the size of $\mathcal{A}$.

Theorem 6 (Approximating parity automata by parity automata with fewer colors). *For every deterministic parity automaton $\mathcal{A} = (Q, Q_0, \delta, \mu)$ with m colors, a bound $n \in \mathbb{N}$ and $0 < m' < m$, there is a deterministic parity automaton $\mathcal{A}'$ with m' colors and $(n \cdot |Q| + 1) \cdot |Q| \cdot (m - m' + 2)$ states such that $L(\mathcal{A}') \subseteq_n \mathcal{L}(\mathcal{A})$.*

Idea & Construction. Our automaton construction is based on the following idea. An ultimately-periodic word in $L(\mathcal{A})$ representable by a lasso of length n has an ultimately-periodic run in $\mathcal{A}$ of size at most $n \cdot |Q|$. The ultimately-periodic run is accepting if the highest color occurring in its period is even. For a given ultimately-periodic word with lasso of length n, our constructed automaton $\mathcal{A}'$ checks whether this word has an ultimately-periodic accepting run of size $n \cdot |Q|$ in $\mathcal{A}$. Adapting the same idea as in Theorem 3, we check whether the colors we wish to eliminate appear within $n \cdot |Q|$ steps. We reject words with runs where these colors appear with distances larger than $n \cdot |Q|$. On the other runs we use the acceptance condition of the remaining colors.

Let $\mathcal{A} = (Q, Q_0, \delta, \mu)$ where $\mu : Q \to \{0, \ldots, m - 1\}$. We construct the parity automaton $\mathcal{A}' = (Q', Q'_0, \delta', \mu')$ with $\mu' : Q \to \{0, \ldots, m' - 1\}$ and where:

$$Q' = (Q \times \{0, \ldots, n \cdot |Q|\}) \cup (Q \times \{0, \ldots, n \cdot |Q|\} \times \{-1, m', \ldots, m - 1\})$$

and

$$Q'_0 = \{(q, 0) \mid q \in Q_0\}.$$

The transition relation and coloring function are given as follows. In contrast to Theorem 3 we now need to first check which is the highest color that appears in the period of the run. This check is done respecting the following cases.

Case (1): $\delta'((q, c), \alpha) = \{(q', c + 1) \mid q' \in \delta(q, \alpha)\}$ if $c < n \cdot |Q| - 1$

As we are only interested in the highest color that appears in the period of the run, case (1) makes sure that we reach this period by skipping the first $n \cdot |Q|$ steps, i.e., we simply follow the transition relation of $\mathcal{A}$ and increase the counter (denoted by c).

Case (2): $\delta'((q, c), \alpha) = \{(q', 0, \mu(q')) \mid q' \in \delta(q, \alpha), \mu(q') \geq m'\} \cup$
$$\{(q', 0, -1) \mid q' \in \delta(q, \alpha), \ \mu(q') < m'\}   \text{if } c = n \cdot |Q| - 1$$

In Case (2) is the transition to the second phase, once we have skipped the first $n \cdot |Q|$ states. From here on we save the highest color seen that is larger than $m' - 1$.

Case (3): $\delta'((q, c, h), \alpha) = \{(q', 0, \mu(q')) \mid q' \in \delta(q, \alpha), \mu(q') > h, \mu(q') \geq m'\} \cup$
$$\{(q', 0, h) \mid q' \in \delta(q, \alpha), \ \mu(q') = h\} \cup$$
$$\{(q', c + 1, h) \mid q' \in \delta(q, \alpha), \ \mu(q') < h \vee \mu(q') < m'\}$$
$$\text{if } c \leq n \cdot |Q| - 1$$

In case (3) we track the highest color h seen so far. If h is higher than $m' - 1$ we save this color and check how long it takes for this color to reappear. In case it appears in less that $n \cdot |Q|$ steps ($\mu(q') = h$) we reset the counter for this color. If a higher color is observed ($\mu(q') > h$), h is replaced by the color and the counter is reset.

Case (4): $\delta'((q, c, h), \alpha) = \{(q', c, h) \mid q' \in \delta(q, \alpha), \ \mu(q') < m'\}$
$$\text{if } c = n \cdot |Q| \text{ and } h = -1$$

In the case where the counter exceeds $n \cdot |Q|$ for some saved color, the automaton rejects, but only if colors higher than m' were observed along the way. Otherwise, the automaton $\mathcal{A}'$ accepts as $\mathcal{A}$ with the non-eliminated colors. The coloring function is defined as follows

$$\mu'(\tilde{q}) = \begin{cases} 0 & \tilde{q} = (q, c) \\ 1 & \tilde{q} = (q, c, h), \ 0 \leq c < n \cdot |Q|, h \text{ is odd} \\ 0 & \tilde{q} = (q, c, h), \ 0 \leq c < n \cdot |Q|, h \text{ is even} \\ \mu(q) & \tilde{q} = (q, c, -1), \ c = n \cdot |Q| \end{cases}$$

$\square$

With this, we conclude the study of the approximation of linear-time properties represented by ω-automata with lasso-precise automata with simpler acceptance conditions preserving their determinism. In the next subsection, we turn to the approximation of nondeterministic automata with lasso-precise deterministic automata.

3.4   Lasso-precise Deterministic Approximations

We now study lasso-precise approximations from the point of view of the determinism of the automata representing ω-regular languages. The complexity of determinizing ω-automata, in particular the construction of deterministic parity automata, has been studied extensively (cf. [18]). The size of the deterministic automaton that recognizes the same language as the given nondeterministic automaton is, in the worst case, exponential in the number of states of the given automaton. By contrast, the size of the deterministic safety automaton provided by Theorem 1 is independent of the given language and exponential only in the bound. For small bounds, Theorem 1 thus provides a deterministic lasso-precise approximation with a small number of states. The following theorem shows that, for large bounds, it is not, in general, possible to produce small deterministic lasso-precise approximations. If the bound is as large as the number of states of the given nondeterministic automaton, then the deterministic lasso-precise approximation has, in the worst case, an exponential number of states.

Theorem 7. *For every $k \in \mathbb{N}$ there exists a nondeterministic parity automaton $\mathcal{A}$ with $O(k)$ states, such that, for every bound $n \geq |\mathcal{A}|$, the minimal deterministic parity automaton $\mathcal{A}'$ with $L(\mathcal{A}') \subseteq_n L(\mathcal{A})$ has at least 2^k states.*

Proof. Let $\Sigma = \{0, 1, 2\}$, and consider the language

$$\Omega = \{\{0, 1\}^i \cdot 1 \cdot \{0, 1\}^{(k-1)} \cdot 2 \cdot 1^\omega \mid i < k\}.$$

That is, Ω consists of the infinite words over $\{0, 1, 2\}$ in which the letter 2 appears exactly once, and the letter exactly k positions prior to that is a 1, preceded by at most $k - 1$ letters.

We can construct a nondeterministic parity automaton $\mathcal{A} = (Q, Q_0, \delta, \mu)$ for Ω with $2k + 1$ states as follows. We let $Q = \{0, 1\} \times \{1, \ldots, k\} \cup \{q_a\}$ and $Q_0 = \{(0, 1)\}$. The function μ is such that $\mu(q_a) = 0$, and $\mu(q) = 1$ for all $q \neq q_a$. We define the transition relation δ such that $\delta(q_a, 1) = \{q_a\}$ and $\delta(q_a, \alpha) = \emptyset$ if $\alpha \in \{0, 2\}$ and

$$\delta((b, i), \alpha) = \begin{cases} \{(0, i+1)\} & \text{if } b = 0, i < k, \alpha = 0, \\ \{(0, i+1), (1, 1)\} & \text{if } b = 0, i < k, \alpha = 1, \\ \{(1, 1)\} & \text{if } b = 0, i = k, \alpha = 1, \\ \{(1, i+1)\} & \text{if } b = 1, i < k, \alpha \neq 2 \\ \{q_a\} & \text{if } b = 1, i = k, \alpha = 2. \end{cases}$$

Let $n \in \mathbb{N}$ be a bound such that $n \geq 2k + 1$, and let $\mathcal{A}'$ be a DPA such that $L(\mathcal{A}') \subseteq_n L(\mathcal{A})$. By the definition of Ω and the fact that $n \geq 2k + 1$ we have that $L(\mathcal{A}') = \Omega$. We will show that $\mathcal{A}'$ has at least 2^k states.

Suppose that $|\mathcal{A}'| < 2^k$. This means that there exist two different words $\sigma_1, \sigma_2 \in \{0,1\}^k$ such that $\mathcal{A}'$ ends up in the same state when run on $\sigma_1 = \alpha_{1,1} \ldots \alpha_{1,k}$ and when run on $\sigma_2 = \alpha_{2,1} \ldots \alpha_{2,k}$. Since σ_1 and σ_2 are different, there must exist an i such that $\alpha_{1,i} \neq \alpha_{2,i}$. W.l.o.g., suppose that $\alpha_{1,i} = 1$ and $\alpha_{2,i} = 0$. Let $\sigma = 1^{i-1} \cdot 2 \cdot 1^\omega$. Consider the words $\sigma_1 \cdot \sigma$ and $\sigma_2 \cdot \sigma$. In $\sigma_1 \cdot \sigma$, the letter appearing k positions before the letter 2 is 1, and in $\sigma_2 \cdot \sigma$ this letter is 0. Thus, by the definition of Ω and $\mathcal{A}'$ we have that $\sigma_1 \cdot \sigma$ must be accepted by $\mathcal{A}'$, and $\sigma_2 \cdot \sigma$ must be rejected, which is a contradiction with the fact that $\mathcal{A}'$ is deterministic and the assumption that σ_1 and σ_2 lead to the same state.

Since $\mathcal{A}'$ is an arbitrary deterministic parity automaton such that $L(\mathcal{A}') \subseteq_n L(\mathcal{A})$, we conclude that the minimal such automaton has at least 2^k states. $\square$

4 Automata with Bounded Size

In many cases, one is interested in constructing an automaton of minimal size for a given language. In this section, we solve the problem of computing n-lasso-precise automata of *bounded size*. By iteratively increasing the bound on the size of the automaton, this approach can be used to construct minimal automata.

Here we consider languages given as LTL formulas [16]. LTL formulas are a common starting point for many verification and synthesis approaches. Rather than going through an intermediate precise automaton, here we propose a symbolic approach that directly yields an automaton whose language is n-lasso-precise approximation for the LTL formula.

Theorem 8. *For a linear-time property φ given as an LTL formula over AP, and given bounds n, k and m, deciding whether there exists a deterministic parity automaton $\mathcal{A}$ of size k and number of colors m, such that, $L(\mathcal{A}) \subseteq_n \varphi$ is in Σ_2^P.*

Proof. We show that the problem can be encoded by a quantified Boolean formula with one quantifier alternation (2-QBF) of size polynomial in the length of the LTL formula φ, and the bounds k, n and m. Deciding quantified Boolean formulas in the 2-QBF fragment is in Σ_2^P [13].

Construction.

$\exists \{\delta_{s,\alpha,s'} \mid s, s' \in Q, \alpha \in 2^{AP}\}$.

$\exists \{\mu_{s,c} \mid s \in Q, 0 \leq c < m\}$

$\forall \{a_j \mid a \in AP, 0 \leq j < \max\{k,n\}\}$.

$\forall \{l_j \mid 0 \leq j < \max\{k,n\}\}$

$\forall \{s_j \mid s \in Q, 0 \leq j < n \cdot k\}$.

$\forall \{r_j \mid 0 \leq j < n \cdot k\}$

$\phi_{\mathrm{DPA}}^{k,m} \wedge (\phi_{\mathrm{loop}} \to \phi_{\mathcal{A} \subseteq \varphi} \wedge \phi_{=_n})$

where

- $\phi_{\mathcal{A} \subseteq \varphi} \equiv \phi_{k\text{-accrun}} \wedge \phi_{\text{match(k)}} \rightarrow \phi_{\in L_k(\varphi)}$
- $\phi_{=_n} \equiv \phi_{\in L_n(\varphi)} \wedge \phi_{\text{match}(n \cdot k)} \rightarrow \phi_{\in L_n(\mathcal{A})}$

The formula encodes the existence of a deterministic parity automaton $\mathcal{A} = (Q, q_0, \delta, \mu)$ with $L(\mathcal{A}) \subseteq_n \varphi$. The transition relation of the automaton is encoded in the variables $\delta_{s,\alpha,s'}$ that define whether the automaton has a transition from state $s \in Q$ to state $s' \in Q$ with a letter $\alpha \in 2^{AP}$. Additional variables $\mu_{s,c}$ define the coloring of the states of the guessed automaton. A variable $\mu_{s,c}$ is true if a state s has color c. Using the constraint $\phi_{\text{DPA}}^{k,m}$ we force a deterministic transition relation for the automaton and make sure that each state has exactly one color

The relation $\subseteq_n$ is encoded in the formula $\phi_{\text{loop}} \rightarrow \phi_{\mathcal{A} \subseteq \varphi} \wedge \phi_{=_n}$. To check whether $\mathcal{A} \subseteq_n \varphi$ we need to check that: (1) $\mathcal{A}$ is a strengthening of φ, i.e., $\mathcal{A} \subseteq \varphi$, and (2) $\mathcal{A}$ is precise up to ultimately-periodic words of size n, i.e., $L_n(A) = L_n(\varphi)$. The strengthening is encoded in the constraint $\phi_{\mathcal{A} \subseteq \varphi} \equiv \phi_{k\text{-accrun}} \wedge \phi_{\text{match(k)}} \rightarrow \phi_{\in L_k(\varphi)}$. To check whether $\mathcal{A}$ is a strengthening of φ we need to check that all accepting ultimately-periodic runs of size k of $\mathcal{A}$ induce ultimately-periodic words of size k that satisfy φ. This is encoded in the formulas $\phi_{k\text{-accrun}}$, $\phi_{\text{match(k)}}$ and $\phi_{\in L_k(\varphi)}$. The formula $\phi_{\text{match(k)}}$ encodes an ultimately-periodic run in $\mathcal{A}$ of size k using the variables s_j for $0 \le j < k$ which determine which state of the automaton is at each position in the run and variables r_j which determine the loop of the run. The formula $\phi_{k\text{-accrun}}$ checks whether this run is accepting by checking the highest color in the period of the run. If both formulas are satisfied then it remains to check whether the induced run satisfies φ, which is done using the constraint $\phi_{\in L_k(\varphi)}$. The constraint resembles the encoding given in [5] for solving the bounded model checking problem. It is defined over the variables a_j, where $a \in AP$ and $0 \le j < k$ and the variables l_j for $0 \le j < k$. A variable a_j is true if the transition at position j in the run that satisfies $\phi_{k\text{-accrun}}$ and $\phi_{\text{match(k)}}$ represents a letter where a is true and if φ allows a to be true at that position. Variables l_j define the position of the loop of the ultimately-periodic word induced by the run.

If $\mathcal{A}$ satisfies the strengthening condition it remains to check whether $\mathcal{A}$ accepts all ultimately-periodic words of size n that satisfy φ. This condition is encoded in the constraint $\phi_{=_n} \equiv \phi_{\in L_n(\varphi)} \wedge \phi_{\text{match}(n \cdot k)} \rightarrow \phi_{\in L_n(\mathcal{A})}$. If an ultimately-periodic word of size n encoded by the variables a_j for $0 \le j < n$ and loop variables l_j satisfies φ (checked by the formula $\phi_{\in L_n(\varphi)}$), then we match this ultimately-periodic word to its run in $\mathcal{A}$ (using the formula $\phi_{\text{match}(n \cdot k)}$). Notice that we have to match the word to a run in $\mathcal{A}$ of size $n \cdot k$ as words of length n might induce runs of size $n \cdot k$. If the latter formulas are satisfied it remains to check whether the run in the automaton is accepting.

Finally, the formula ϕ_{loop} asserts that only one loop is allowed at a time. All formulas are of size polynomial in k, n, m and φ.   □

The construction above can also be used for computing nondeterministic automata by changing the constraints on the transition relation of the automaton.

5   Discussion

The key idea behind algorithmic methods like bounded model checking [5] and bounded synthesis [9] is that, for finite-state systems, it suffices to consider lasso-shaped executions of bounded size. The notion of n-lasso-precise approximation, introduced in this paper, exploits the same observation for the construction and transformation of automata.

The new constructions for n-lasso-precise underapproximations have attractive properties. Theorem 1 shows that it is possible to approximate a given language with a deterministic safety automaton whose size is exponential in the bound, but independent of the given language. For small bounds, any language can thus be effectively approximated by a deterministic safety automaton. Theorem 6 shows that reducing the number of colors of a parity automaton incurs at most a polynomial increase in the number of states of the original automaton.

The results indicate significant potential for new verification and synthesis algorithms that work with n-lasso-precise approximations instead of precise automata. A key novelty is that our constructions allow us to approximate a given temporal property with a property of a simpler type without necessarily reducing all the way to safety. For example, we can approximate a given temporal property with a parity automaton with three colors, for which efficient synthesis algorithms exist [2].

The constructions of the paper allow us to directly construct automata for the approximations. An interesting topic for future work is to complement these constructions with fast techniques that reduce the number of states of an automaton without necessarily producing a minimal automaton. Similar techniques, which, however, guarantee full language equivalence rather than n-lasso precision, are commonly used in the translation of LTL formulas to Büchi automata (cf. [11]).

References

1. Armin Biere, Cyrille Artho, and Viktor Schuppan. Liveness checking as safety checking. *Electr. Notes Theor. Comput. Sci.*, 66(2):160–177, 2002.
2. Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. *Journal of Computer and System Sciences*, 78(3):911 – 938, 2012. In Commemoration of Amir Pnueli.
3. Udi Boker. Why these automata types? In Gilles Barthe, Geoff Sutcliffe, and Margus Veanes, editors, *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 57 of *EPiC Series in Computing*, pages 143–163. EasyChair, 2018.
4. Hugues Calbrix, Maurice Nivat, and Andreas Podelski. Ultimately periodic words of rational w-languages. In *Proceedings of the 9th International Conference on Mathematical Foundations of Programming Semantics*, pages 554–566, London, UK, UK, 1994. Springer-Verlag.
5. Edmund Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Form. Methods Syst. Des.*, 19(1):7–34, July 2001.

6. Rayna Dimitrova, Bernd Finkbeiner, and Hazem Torfah. Synthesizing approximate implementations for unrealizable specifications. In *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I*, pages 241–258, 2019.
7. Rüdiger Ehlers and Bernd Finkbeiner. On the virtue of patience: Minimizing Büchi automata. In *Model Checking Software - 17th International SPIN Workshop, Enschede, The Netherlands, September 27-29, 2010. Proceedings*, pages 129–145, 2010.
8. Kousha Etessami and Gerard J. Holzmann. Optimizing Büchi automata. In *CONCUR 2000 - Concurrency Theory, 11th International Conference, University Park, PA, USA, August 22-25, 2000, Proceedings*, pages 153–167, 2000.
9. Bernd Finkbeiner and Sven Schewe. Bounded synthesis. *International Journal on Software Tools for Technology Transfer*, 15(5-6):519–539, 2013.
10. Graeme Gange, Pierre Ganty, and Peter J. Stuckey. Fixing the state budget: Approximation of regular languages with small DFAs. In *Automated Technology for Verification and Analysis - 15th International Symposium, ATVA 2017, Pune, India, October 3-6, 2017, Proceedings*, pages 67–83, 2017.
11. Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Computer Aided Verification*, pages 53–65, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
12. Dimitra Giannakopoulou and Flavio Lerda. From states to transitions: Improving translation of LTL formulae to Büchi automata. In *Formal Techniques for Networked and Distributed Systems - FORTE 2002, 22nd IFIP WG 6.1 International Conference Houston, Texas, USA, November 11-14, 2002, Proceedings*, pages 308–326, 2002.
13. Hans Kleine Büning and Uwe Bubeck. Theory of quantified boolean formulas. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 735–760. IOS Press, 2009.
14. Sriram C. Krishnan, Anuj Puri, and Robert K. Brayton. Structural complexity of omega-automata. In *STACS*, pages 143–156, 1995.
15. Orna Kupferman, Yoad Lustig, Moshe Y. Vardi, and Mihalis Yannakakis. Temporal synthesis for bounded systems and environments. In *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, pages 615–626, 2011.
16. Amir Pnueli. The temporal logic of programs. In *the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, Washington, DC, USA, 1977. IEEE Computer Society.
17. Shmuel Safra. *Complexity of Automata on Infinite Objects*. PhD thesis, Weizmann Institute of Science, Rehovot, Israel, March 1989.
18. Sven Schewe and Thomas Varghese. Determinising parity automata. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I*, pages 486–498, 2014.
19. Viktor Schuppan and Armin Biere. Efficient reduction of finite state model checking to reachability analysis. *STTT*, 5(2-3):185–204, 2004.
20. Jeffrey Shallit and Yuri Breitbart. Automaticity I: properties of a measure of descriptional complexity. *J. Comput. Syst. Sci.*, 53(1):10–25, 1996.
21. Fabio Somenzi and Roderick Bloem. Efficient Büchi automata from LTL formulae. In *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, pages 248–263, 2000.