Bounded Model Construction for Monadic Second-Order Logics

Abdelwaheb Ayari and David Basin

Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Germany.
`www.informatik.uni-freiburg.de/~{ayari|basin}`

Abstract. The monadic logics M2L-Str and WS1S have been successfully used for verification, although they are nonelementary decidable. Motivated by ideas from bounded model checking, we investigate procedures for bounded model construction for these logics. The problem is, given a formula ϕ and a bound k, does there exist a word model for ϕ of length k. We give a bounded model construction algorithm for M2L-Str that runs in a time exponential in k. For WS1S, we prove a negative result: bounded model construction is as hard as validity checking, *i.e.*, it is nonelementary. From this, negative results for other monadic logics, such as S1S, follow. We present too preliminary tests using a SAT-based implementation of bounded model construction; for certain problem classes it can find counter-examples substantially faster than automata-based decision procedures.

1 Introduction

The monadic logics M2L-Str, WS1S, and S1S are among the most expressive decidable logics known. The logic M2L-Str [11] is a logic on finite words and also appears in the literature (with slight variations) under the names MSO[S] [20] and SOM[+1] [19]. In the early 1960's, Büchi and Elgot gave decision procedures for these logics by exploiting the fact that models can be encoded as words and that the language of models satisfying a formula can be represented by an automaton [5,6,9]. These decision procedures provide nonelementary upper-bounds for these logics, which are also the lower-bounds [16].

Despite their atrocious complexity, the decision procedures for M2L-Str and WS1S have been implemented in numerous tools, *e.g.*, Mona [14], Mosel [18], MOSEL [12], and the STEP system [15], and have been successfully applied to problems in diverse domains including hardware [2] and protocol [11] verification. Not surprisingly though, many large systems cannot be verified due to state explosion. This is analogous to state explosion in model checking where the state-space is exponential in the number of state variables, except for monadic logics the states in the constructed automaton can be nonelementary in the size of the input formula! For LTL model checking, a way of finessing this problem has recently been proposed: *bounded model checking* [4]. The idea is that one can finitely represent counter-examples (using the idea of a loop, see §4.3), and, by bounding the size of these representations, satisfiability checkers can be used to search for them. This often succeeds in cases where symbolic model checking fails.

Motivated by the bounded model checking approach and the goal of quick generation of counter-examples for falsifiable formulae, we investigate an analogous problem for monadic logics. Namely, given a formula ϕ and a natural number k, determine if ϕ has a word model of length k. Since we are concerned with *constructing* models for formulae, as opposed to *checking* their satisfiability with respect to a given model, we call our problem *bounded model construction* or BMC for short.

We show that for M2L-Str, given a formula ϕ and a natural number k, we can generate a formula in quantified Boolean logic that is satisfiable if and only if ϕ has a word model of length k. The formula generated is polynomial in the size of ϕ and k and can be tested for satisfiability in polynomial space. For generating length k counter-models, this yields a nonelementary improvement over the automata-based decision procedure for M2L-Str. Moreover, we show that the use of SAT-based techniques can have acceptable running times in practice.

We also investigate bounded model construction for other monadic logics and establish negative results. For WS1S we show that BMC is as hard as checking validity, which is nonelementary. This result is somewhat surprising since WS1S has the same expressiveness and complexity as M2L-Str and their decision procedures differ only slightly. Indeed, there has been a recent investigation of the differences of these logics by Klarlund who concluded that WS1S is preferable to M2L-Str due to its simpler semantics and its wider applicability to arithmetic problems [13]. Our results suggests that the issue is not so clear cut and depends on whether error detection through counter-example generation versus full verification is desired, that is, whether one is interested in finding a single model for a formula or computing a description of all models. We also formulate BMC for S1S and several first-order monadic logics and establish similar negative results.

We proceed as follows. In §2 we briefly review quantified Boolean logic and finite automata on words. In §3 we describe the syntax and semantics of M2L-Str and WS1S and their relationship to finite automata. In §4 we present the bounded model construction approach and complexity results. In §5 we present experimental results and in §6 we draw conclusion.

2 Background

Boolean Logic Boolean formulae are built from the constants true and false, variables $x \in \mathcal{V}$, and are closed under the standard connectives. The formulae are interpreted in $\mathbb{B} = \{0, 1\}$. A (Boolean) *substitution* $\sigma : \mathcal{V} \to \mathbb{B}$ is a mapping from variables to truth values that is extended homomorphically to formulae. We say σ *satisfies* ϕ if $\sigma(\phi) = 1$.

Quantified Boolean logic (QBL) extends Boolean logic (BL) by allowing quantification over Boolean variables, *i.e.*, $\forall x.\, \phi$ and $\exists x.\, \phi$. A substitution σ

satisfies $\forall x.\, \phi$ if σ satisfies $\phi[\mathsf{true}/x] \wedge \phi[\mathsf{false}/x]$ and dually for $\exists x.\, \phi$. In the remainder of the paper, we write $\sigma \models_{\mathrm{QBL}} \phi$ to denote that σ satisfies ϕ.

QBL is not more expressive than BL, but it is more succinct. The satisfiability problem for Boolean logic is *NP-complete* [8], whereas it is *PSPACE-complete* for QBL [17].

Automata on Words Let Σ denote a finite alphabet. Σ^* (respectively Σ^ω) denotes the set of finite (respectively infinite) words over Σ. A finite automaton $\mathcal{A}$ over Σ is a tuple (S, s_0, Δ, F) where S is a nonempty finite set of states, $s_0 \in S$ is the initial state, $\Delta \subseteq S \times \Sigma \times S$ is a transition relation, and $F \subseteq S$ is a set of final states. A run of $\mathcal{A}$ on a finite word $w = a_1 a_2 \dots a_n$ (respectively, an infinite word $w = a_1 a_2 \dots$) is a finite sequences of states $s_0 s_1 \dots s_n$ (respectively, an infinite sequence of states $s_0 s_1 \dots$) with $(s_i, a_i, s_{i+1}) \in \Delta$. A finite word is accepted by an automaton if it has a run whose last state is final. To accept infinite words, finite automata are equipped with a *Büchi acceptance condition*, which says that an infinite word is accepted if it has a run in which some final state occurs infinitely often. We will often use the alphabet $\mathbb{B}^n$, with $n \in \mathbb{N}$. Note that $\mathbb{B}^0$ stands for the singleton set $\{()\}$, *i.e.*, the set whose only member is the degenerate tuple "()".

3 Monadic Second-Order Logics on Finite Words

In this section we provide background material on M2L-Str and WS1S. These logics have the same syntax but slightly different semantics. We also explain their relationship to regular languages.

Let $\mathcal{V}_1 = \{x_i \mid i \in \mathbb{N}\}$ be a set of first-order variables and $\mathcal{V}_2 = \{X_i \mid i \in \mathbb{N}\}$ be a set of second-order variables. We will use n, p, q, ... as meta-variables ranging over $\mathcal{V}_1$ and we use X, Y, ... as meta-variables ranging over $\mathcal{V}_2$.

3.1 Language

Monadic second order (MSO) formulae are formulae in a language of second-order arithmetic specified by the grammar:

$$t ::= \mathsf{0} \mid p, \qquad p \in \mathcal{V}_1$$

$$\phi ::= \mathsf{s}(t,t) \mid X(t) \mid \neg\phi \mid \phi \vee \phi \mid \exists p.\, \phi \mid \exists X.\, \phi, \qquad p \in \mathcal{V}_1 \text{ and } X \in \mathcal{V}_2$$

Hence terms are built from the constant $\mathsf{0}$ and first-order variables. Formulae are built from predicates $\mathsf{s}(t, t')$ and $X(t)$ and are closed under disjunction, negation, and quantification over first-order and second-order variables. Other connectives and quantifiers can be defined using standard classical equivalences, *e.g.*, $\forall X.\, \phi \overset{def}{=} \neg\exists X.\, \neg\phi$. In other presentations, s is usually a function. We have specified it as a relation for reasons that will become apparent when we give the semantics. In the remainder of this section, formula means MSO-formula.

3.2 Semantics

For X a set, by $\mathcal{F}(X)$ we denote the set of finite subsets of X. A (MSO) substitution σ is a pair of mappings $\sigma = (\sigma_1, \sigma_2)$, with $\sigma_1 : \mathcal{V}_1 \to \mathbb{N}$ and $\sigma_2 : \mathcal{V}_2 \to \mathcal{F}(\mathbb{N})$ and for $x \in \mathcal{V}_1$, $\sigma(x) = \sigma_1(x)$ and for $X \in \mathcal{V}_2$, $\sigma(X) = \sigma_2(X)$. With this in hand, we can now define satisfiability for M2L-Str and WS1S.

The Logic M2L-STR Formulae in M2L-Str are interpreted relative to a natural number $k \in \mathbb{N}$. We will write $[k]$ for the set $\{0, \ldots, k-1\}$ and we call the elements of $[k]$ *positions*. First-order variables are interpreted as positions. The constant $\mathsf{0}$ denotes the natural number 0 and the symbol s is interpreted as the relation $\{(i,j) \mid j = i+1 \text{ and } i,j \in [k]\}$. Note that $k-1$ has no successor. Second-order variables denote subsets of $[k]$ and the formula $X(t)$ is true when the position denoted by t is in the set denoted by X.

More formally, the semantics of a formula ϕ is defined inductively relative to a substitution σ and a $k \in \mathbb{N}$. In the following, we write σ^k for the pair (σ, k).

Definition 1 *Satisfiability for* M2L-Str

$$\begin{array}{lll}
\sigma^k \models_{M2L} \mathsf{s}(t,t'), & \textit{if} & \sigma(t') = 1 + \sigma(t) \textit{ and } \sigma(t') \in [k] \\
\sigma^k \models_{M2L} X(t), & \textit{if} & \sigma(t) \in \sigma(X) \\
\sigma^k \models_{M2L} \neg\phi, & \textit{if} & \sigma^k \not\models_{M2L} \phi \\
\sigma^k \models_{M2L} \phi_1 \vee \phi_2, & \textit{if} & \sigma^k \models_{M2L} \phi_1 \textit{ or } \sigma^k \models_{M2L} \phi_2 \\
\sigma^k \models_{M2L} \exists p.\ \phi, & \textit{if} & (\sigma[i/p])^k \models_{M2L} \phi, \textit{ for some } i \in [k] \\
\sigma^k \models_{M2L} \exists X.\ \phi, & \textit{if} & (\sigma[M/X])^k \models_{M2L} \phi, \textit{ for some } M \subseteq [k]
\end{array}$$

If $\sigma^k \models_{\mathrm{M2L}} \phi$, we say that σ^k *satisfies*, or is a *model* of, ϕ. We call a formula ϕ *valid*, and we write $\models_{\mathrm{M2L}} \phi$, if for every natural number k and substitution σ, σ^k satisfies ϕ.

The Logic WS1S Whereas M2L-Str can be seen as a logic on bounded sets of positions or, as we shall see, finite words, WS1S is best viewed as a logic based on arithmetic. First-order variables range over $\mathbb{N}$ and are not a priori bounded by any natural number. Second-order variables range over finite subsets of the natural numbers, $\mathcal{F}(\mathbb{N})$, and are not restricted to subsets of some $[k]$. Finally, the symbol s is interpreted as the successor relation over $\mathbb{N}$. Formally, we define satisfiability in WS1S, $\sigma \models_{\mathrm{WS1S}} \phi$, as follows:

Definition 2 *Satisfiability for* WS1S

$$\begin{array}{lll}
\sigma \models_{WS1S} \mathsf{s}(t,t'), & \textit{if} & \sigma(t') = 1 + \sigma(t) \\
\sigma \models_{WS1S} X(t), & \textit{if} & \sigma(t) \in \sigma(X) \\
\sigma \models_{WS1S} \neg\phi, & \textit{if} & \sigma \not\models_{WS1S} \phi \\
\sigma \models_{WS1S} \phi_1 \vee \phi_2, & \textit{if} & \sigma \models_{WS1S} \phi_1 \textit{ or } \sigma \models_{WS1S} \phi_2 \\
\sigma \models_{WS1S} \exists p.\ \phi, & \textit{if} & \sigma[i/p] \models_{WS1S} \phi, \textit{ for some } i \in \mathbb{N} \\
\sigma \models_{WS1S} \exists X.\ \phi, & \textit{if} & \sigma[M/X] \models_{WS1S} \phi, \textit{ for some } M \in \mathcal{F}(\mathbb{N})
\end{array}$$

A formula is *valid* in WS1S if it is satisfied by every substitution σ.

Word Models Models in both M2L-Str and WS1S can be encoded as finite words. Let $\phi(\overline{X})$ be a formula, where $\overline{X}$ is the tuple of second-order variables $X_1, \ldots, X_n$ occurring free in ϕ.[1] We encode a M2L-Str model σ^k for ϕ by the word $w_{\sigma^k} \in (\mathbb{B}^n)^k$, such that the length of w_{σ^k} is k and for every position $i \in [k]$, $w_{\sigma^k}(i) = (b_1, \ldots, b_n)$ and for $1 \leq j \leq n$, $b_j = 1$ iff $i \in \sigma(X_j)$. We call w_{σ^k} *a word model* for ϕ and define $\mathcal{L}_{\text{M2L}}(\phi)$ as the set of all M2L-Str word models for ϕ. We shall also write $w \models_{\text{M2L}} \phi$ for $\sigma^k \models_{\text{M2L}} \phi$, where w encodes σ^k.

Similarly, a WS1S model σ for ϕ can be encoded as a finite word w_σ such that $w_\sigma(i) = (b_1, \ldots, b_n)$ where $b_j = 1$ iff $i \in \sigma(X_j)$. We also call w_σ a *word model* for ϕ in WS1S. We define $\mathcal{L}_{\text{WS1S}}(\phi)$ as the set of WS1S word models for ϕ. Note that the encoding of M2L-Str models as words is a bijection, whereas this is not the case for WS1S. In particular, if σ is a WS1S model and w_σ encodes it, then any finite word of the form $w_\sigma aa \cdots a$, where a is $(0, \ldots, 0) \in \mathbb{B}^n$, also encodes σ. We shall also write $w \models_{\text{WS1S}} \phi$ for $\sigma \models_{\text{WS1S}} \phi$, where w encodes σ.

Example Consider the formula $\phi \stackrel{def}{=} X(0) \wedge \forall p.\, X(p) \leftrightarrow (\exists q.\, \mathsf{s}(p,q) \wedge Y(q))$ and the substitution σ with $\sigma(X) = \{0, 2\}$ and $\sigma(Y) = \{0, 1, 3\}$. σ^4 is a model for ϕ in M2L-Str and σ is a model for ϕ in WS1S. The words w and w' below encode σ^4 and σ, respectively.

w	0	1	2	3
X	1	0	1	0
Y	1	1	0	1

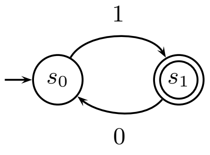
w'	0	1	2	3	4	5
X	1	0	1	0	0	0
Y	1	1	0	1	0	0

As a second example, the formula $\exists X.\, \forall p.\, X(p)$ is valid in M2L-Str, whereas it is unsatisfiable in WS1S.

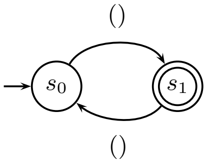
Connection to Regular Languages We have seen that monadic formulae define sets of word models. Büchi and Elgot proved in [5,9] that the languages formalized by formulae in WS1S and M2L-Str are regular and, conversely, that every regular language is both WS1S and M2L-Str-definable. To show regularity, they proved constructively that, given a formula ϕ, there exists an automaton A_ϕ that accepts all WS1S (respectively, M2L-Str) word models for ϕ. This construction yields a decision procedure: a closed formula is valid in WS1S (respectively, in M2L-Str) iff its corresponding automaton accepts the language $()^*$. This decision procedure (and indeed any decision procedure for these logics) is nonelementary [16,21]: the minimal automaton representing a formula of size n may require space whose lower bound is a stack of exponentials of height n.

As noted previously, in WS1S any word model over $\Sigma = \mathbb{B}^n$ can be suffixed by arbitrarily many $(0, \ldots, 0) \in \mathbb{B}^n$ and the result is again a word model. Hence we explain in which sense regular languages are definable in both monadic logics, as this is not completely straightforward. Let $\Sigma = \{a_1, \ldots, a_n\}$ and let $\theta : \mathbb{B}^n \to$

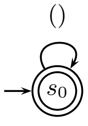
[1] First-order variables can be encoded using second-order variables as we will show in §4.1.



(a) Automaton for $1(01)^*$



(b) $\exists X.\ \mathit{alternate}_{\mathrm{M2L}}(X)$



(c) $\exists X.\ \mathit{alternate}_{\mathrm{WS1S}}(X)$

Fig. 1. Automata for Example

Σ be the substitution defined by $\theta(b_1,\ldots,b_n) = a_i$, where $b_j = 1$ iff $j = i$, and let $\sim$ be the congruence relation over $(\mathbb{B}^n)^*$ defined by $u \sim v$ iff $u = x.(0,\ldots,0)^i$ and $v = x.(0,\ldots,0)^j$ with $x \in (\mathbb{B}^n)^*$ and $i, j \in \mathbb{N}$. We straightforwardly extend θ to words over $(\mathbb{B}^n)^*$, sets of words, $\sim$-classes, and sets of $\sim$-classes. Now, for a regular language $L \subseteq \Sigma^*$, we can construct formulae $\phi(X_1,\ldots,X_n)$ and $\psi(X_1,\ldots,X_n)$ such that for M2L-Str we have $L = \theta(\mathcal{L}_{\mathrm{M2L}}(\phi(\overline{X})))$ and for WS1S we have $L = \theta(\mathcal{L}_{\mathrm{WS1S}}(\psi(\overline{X}))/\sim)$.

Example Consider the automaton $\mathcal{A}$ depicted in Figure 1(a) that accepts the language $1(01)^* = \{1,\ 101,\ 10101,\ldots\}$. This language is defined by the formula[2]

$$\begin{aligned} \mathit{alternate}_{\mathrm{M2L}}(X) \ &\overset{def}{=} \exists n.\ \neg\exists p.\ \mathsf{s}(n,p) \wedge && (1)\\ & X(\mathsf{0}) \wedge X(n) \wedge && (2)\\ & \forall p.\ p < n \to \exists q.\ \mathsf{s}(p,q) \wedge (X(q) \leftrightarrow \neg X(p)) && (3) \end{aligned}$$

interpreted in M2L-Str. (1) formalizes that n denotes (the last position) k of Definition 1. (2) states that the first and last positions are in X, and, by (3), the positions in X alternate. Observe that if we existentially quantify the variable X in $\mathit{alternate}_{\mathrm{M2L}}(X)$, then we obtain a closed formula that is neither valid nor unsatisfiable; its corresponding automaton, given in Figure 1(b), is the same as $\mathcal{A}$ except its transitions are labeled with $() \in \mathbb{B}^0$.

For WS1S we can define the same language with the formula

$$\mathit{alternate}_{\mathrm{WS1S}}(X) \ \overset{def}{=} \ \exists n.\ (\forall p.\ n < p \to \neg X(p)) \wedge (2) \wedge (3)\,.$$

The only difference is that to state that n is the last position we require that X contains no positions greater than n. The language $\mathcal{L}_{\mathrm{WS1S}}(\mathit{alternate}_{\mathrm{WS1S}}(X))$

[2] The less-than relation $<$ is definable in M2L-Str, WS1S, and S1S (introduced in §4.3).

is $1(01)^*0^*$ and $\mathcal{L}_{\text{WS1S}}(\psi(\overline{X}))/\sim$ is $1(01)^*$. In contrast to M2L-Str, if we existentially quantify the variable X in $\mathit{alternate}_{\text{WS1S}}(X)$, then we obtain a valid formula and its automaton is depicted in Figure 1(c).

4 Bounded Model Construction

In this section we present bounded model construction, which can generate counter-examples for non-theorems nonelementary faster than its automata-theoretic counterpart. We show this for M2L-Str and give negative results, showing the impossibility of such procedures, for other monadic logics.

The problem we analyze is how to generate counter-examples of a given size and do this quickly (elementary!) with respect to the size parameter. We express this in the format of a parameterized complexity problem (cf. [1]). For L either M2L-Str or WS1S, we define:

Definition 3
Bounded Model Construction for L *(*BMC(L)*)*
Instance: *A formula ϕ and a natural number k.*
Parameter: *k.*
Question: *Does ϕ have a satisfying word model of length k with respect to L? (That is, is there a word w of length k with $w \models_{\mathsf{L}} \phi$?)*

4.1 Bounded Model Construction for M2L-Str

We proceed by defining a family of functions $(\lceil . \rceil_k)_{k\in\mathbb{N}}$ that transforms MSO-formulae into quantified Boolean formulae such that there is word model of length k for ϕ iff $\lceil \phi \rceil_k$ is satisfiable. The size of the resulting formula is polynomial in the size of ϕ and k.

To simplify matters, we reduce MSO to a minimal kernel, called MSO$_0$, which is as expressive as MSO. The language MSO$_0$ has the grammar:

$$\phi ::= \mathsf{Succ}(X,Y) \mid X \subseteq Y \mid \neg\phi \mid \phi \vee \phi \mid \exists X.\, \phi, \qquad X, Y \in \mathcal{V}_2 \ .$$

$\mathsf{Succ}(X,Y)$ means that X and Y are singletons $\{p\}$ and $\{q\}$, where $q = p + 1$. The symbol $\subseteq$ denotes the subset relation. Note that first-order variables are omitted and can be encoded as singletons. There is a simple polynomial time translation from MSO formulae into MSO$_0$ [20].

Translation to QBL Let $k \in \mathbb{N}$ be fixed. We now describe how to calculate the QBL formula $\lceil \phi \rceil_k$ for a MSO$_0$-formula ϕ. The idea is simple: a set $M \subseteq [k]$ can be represented by k Boolean variables $x_0, \ldots, x_{k-1}$ such that $x_i = 1$ iff $i \in M$. Building on this, we encode relations between finite sets and formulae over these relations.

Let $\mathcal{V}_0 = \{x^i_j \mid i, j \in \mathbb{N}\}$ be a set of Boolean variables and $\mathsf{singleton}$ be the proposition

$$\mathsf{singleton}(x_0, \ldots, x_{k-1}) \overset{def}{=} \bigvee_{0 \le i \le k-1} (x_i \wedge \bigwedge_{\substack{0 \le j \le k-1 \\ j \ne i}} \neg x_j) \ .$$

The mapping $\lceil . \rceil_k$ is inductively defined as follows:

Definition 4 (Translation)

$$\begin{aligned}
\lceil X_m \subseteq X_n \rceil_k &= \textstyle\bigwedge_{0 \le i \le k-1} (x^m_i \rightarrow x^n_i) \\
\lceil \mathsf{Succ}(X_m, X_n) \rceil_k &= \mathsf{singleton}(x^m_0, \ldots, x^m_{k-1}) \wedge \mathsf{singleton}(x^n_0, \ldots, x^n_{k-1}) \wedge \\
&\quad \textstyle\bigvee_{0 \le i < k-1} (x^m_i \rightarrow x^n_{i+1}) \\
\lceil \phi_1 \vee \phi_2 \rceil_k &= \lceil \phi_1 \rceil_k \vee \lceil \phi_2 \rceil_k \\
\lceil \neg \phi \rceil_k &= \neg \lceil \phi \rceil_k \\
\lceil \exists X_m . \, \phi \rceil_k &= \exists x^m_0, \ldots, x^m_{k-1} . \, \lceil \phi \rceil_k
\end{aligned}$$

Definition 5 *For a substitution $\sigma : \mathcal{V}_2 \rightarrow \mathcal{F}(\mathbb{N})$, we define the Boolean substitution $\widehat{\sigma} : \mathcal{V}_0 \rightarrow \mathbb{B}$, by $\widehat{\sigma}(x^m_i) = 1$ iff $i \in \sigma(X_m)$.*

Lemma 1 *Let σ be a substitution and $k \in \mathbb{N}$. Then $\sigma^k \models_{M2L} \phi$ iff $\widehat{\sigma} \models_{QBL} \lceil \phi \rceil_k$.*

Proof. By induction on the construction of ϕ.

We first establish the claim for atomic formulae. To begin with, $\sigma^k \models_{\text{M2L}} X_m \subseteq X_n$ iff for all i, $0 \le i \le k-1$, $i \in \sigma(X_m)$ implies that $i \in \sigma(X_n)$, which is equivalent to $\widehat{\sigma} \models_{\text{QBL}} \bigwedge_{0 \le i \le k-1} x^m_i \rightarrow x^n_i$. Similarly, if $\sigma^k \models_{\text{M2L}} \mathsf{Succ}(X_m, X_n)$, then $\sigma(X_m)$ and $\sigma(X_n)$ are singletons. Moreover, $\sigma(X_m)$ contains a natural number p, with $0 \le p < k-1$, whose successor $p+1$ is in $\sigma(X_n)$. Hence there is some i, where $0 \le i < k-1$, such that $\widehat{\sigma} \models_{\text{QBL}} x^m_i \rightarrow x^n_{i+1}$, so $\widehat{\sigma} \models_{\text{QBL}} \bigvee_{0 \le i < k-1} x^m_i \rightarrow x^n_{i+1}$; the converse is argued similarly.

In the inductive step we consider only the case where ϕ is of the form $\exists X_m . \, \psi$ as the remaining cases are straightforward. By Definition 1, $\sigma^k \models_{\text{M2L}} \exists X_m . \, \psi$ iff there is some set $M \subseteq [k]$ such that $(\sigma[M/X_m])^k \models_{\text{M2L}} \psi$. From the induction hypothesis, $(\sigma[M/X_m])^k \models_{\text{M2L}} \psi$ iff $\widehat{\delta} \models_{\text{QBL}} \lceil \psi \rceil_k$, where $\delta = \sigma[M/X_m]$. Note that $\widehat{\delta} = \widehat{\sigma}[b_0/x^m_0, \ldots, b_{k-1}/x^m_{k-1}]$, where $b_i = 1$ iff $i \in M$, for $0 \le i \le k-1$. Further, $\widehat{\delta} \models_{\text{QBL}} \lceil \psi \rceil_k$ iff $\widehat{\sigma} \models_{\text{QBL}} \exists x^m_0, \ldots, x^m_{k-1} . \, \lceil \psi \rceil_k$. Thus $\sigma^k \models_{\text{M2L}} \exists X_m . \, \psi$ iff $\widehat{\sigma} \models_{\text{QBL}} \exists x^m_0, \ldots, x^m_{k-1} . \, \lceil \psi \rceil_k$. □

Observe that given a Boolean substitution τ, it is trivial to define a MSO substitution σ where $\widehat{\sigma} = \tau$, namely by stipulating that $\sigma(X_i) = \{j \mid \tau(x^i_j) = 1\}$. Hence, from the above Lemma we can conclude:

Theorem 1 (Correctness) *Let ϕ be a* MSO *formula. For $k \in \mathbb{N}$, there exists a* MSO *substitution σ where $\sigma^k \models_{M2L} \phi$ iff there exists a Boolean substitution τ where $\tau \models_{QBL} \lceil \phi \rceil_k$. Moreover, ϕ is valid in* M2L-Str *iff for all $k \ge 0$, the* QBL *formula $\lceil \phi \rceil_k$ is valid.*

We define the size of a formula (in any of the logics we consider) as the number of symbols occurring in its string representation. Exploiting the fact that satisfiability for QBL is PSPACE complete, we prove:

Theorem 2 (Complexity) BMC(M2L-Str) *is PSPACE-complete.*

Proof. Let ϕ and k be a problem instance. The size of $\lceil \phi \rceil_k$ is $O(k^2|\phi|)$. It follows that BMC(M2L-Str) can be reduced in polynomial time to satisfiability in QBL, which establishes membership in PSPACE.

To prove PSPACE-hardness, we show that satisfiability for QBL can be reduced in log-space to BMC(M2L-Str). Let E be a fresh second-order variable and empty be the M2L-Str proposition defined by $\mathsf{empty}(X) \overset{def}{=} \forall Y.\ X \subseteq Y$. We encode each Boolean variable x with a M2L-Str variable X. For a QBL formula ϕ, let $\widetilde{\phi}$ be the M2L-Str formula obtained from ϕ as follows: replace occurrences of Boolean variables x by $X \subseteq E$, and replace the Boolean quantifiers as well as the propositional connectives by the corresponding quantifiers and connectives of M2L-Str. Now, the encoding of ϕ in M2L-Str is the formula $\exists E.\ \mathsf{empty}(E) \wedge \widetilde{\phi}$. For example, the QBL formula $\forall x\, \exists y.\, x \vee y$ is encoded as $\exists E.\, \mathsf{empty}(E) \wedge \forall X.\, \exists Y.\, X \subseteq E \vee Y \subseteq E$. Under this encoding it is only relevant whether or not a second-order variable is interpreted by the empty set. We immediately conclude that a QBL formula is satisfiable iff its encoding has a word model of length 1. □

4.2 Bounded Model Construction for WS1S

The previously given translation cannot be employed for WS1S. If ϕ is the formula $\exists X_m.\, \forall X_n.\, X_n \subseteq X_m$, the translation yields the quantified Boolean formula $\exists x_0^m, \dots, x_{k-1}^m.\, \forall x_0^n, \dots, x_{k-1}^n.\, \bigwedge_{0 \le i \le k-1} x_i^n \to x_i^m$, which is valid for every k, whereas ϕ is unsatisfiable in WS1S. We now prove that there is no translation that will yield an elementary bounded model construction procedure.

Theorem 3 BMC(WS1S) *is nonelementary.*

Proof. For a closed formula ϕ, $\sigma \models_{\text{WS1S}} \phi$ iff $\sigma' \models_{\text{WS1S}} \phi$ for all substitutions σ and σ', *i.e.*, the satisfiability of a closed formula does not depend on the substitution. Hence, every closed WS1S formula is either valid or unsatisfiable. Equivalently, for ϕ be a closed formula, either $\mathcal{L}_{\text{WS1S}}(\phi) = ()^*$ or $\mathcal{L}_{\text{WS1S}}(\phi) = \emptyset$. Consequently, if a closed formula ϕ has a word model, then $\mathcal{L}_{\text{WS1S}}(\phi) = ()^*$ and therefore ϕ is valid. In other words, computing a word model of any length for ϕ is equivalent to checking ϕ's validity. □

This proof can be easily adapted for other monadic logics. For example, WFO[<] is the first-order fragment of WS1S augmented by the relation $<$. Meyer showed in [16] that this logic is nonelementary; this result, combined with the above argument, shows that bounded model construction for this logic (BMC(WFO[<])) is also nonelementary.

The reader may wonder what causes these differences. We can gain some insight by comparing semantics. From the semantics of M2L-Str, $\phi(X)$ has a word model of length k iff $\exists X.\, \phi(X)$ has a word model of length k. This semantic property was employed in the proof of Lemma 1, where in order to use the induction hypothesis we require that the witness set M is a subset of $[k]$. Unfortunately, this property fails for WS1S. As can be seen in Figure 1, existential quantification can change the size of the minimal word model in WS1S. A more dramatic example is the family of formulae (written here with sugared syntax) $\phi_n(X) \overset{def}{=} X(n)$, for $n \in \mathbb{N}$. The minimal length word model for $\phi_n(X)$ is n, whereas it is 0 for $\exists X.\, \phi_n(X)$. In general, to determine if a formula has a small, e.g., length 0, word model, we must consider word models for their subformulae that are non-elementary larger in the worst case.

4.3 Bounded Model Construction for S1S

Here we consider monadic logics over *infinite* words. We start with the logic S1S, which is closely related to WS1S and differs only by allowing infinite subsets of $\mathbb{N}$ as interpretations for second-order variables. A substitution in S1S can be encoded as an infinite word and Büchi showed in [6] that S1S exactly captures the ω-regular languages. In doing so, he provided an effective nonelementary transformation of S1S formulae into *Büchi automata.*

Here we prove a negative result analogous to the previous one: there is no elementary BMC procedure for S1S. This problem must first be properly defined, since bounded model construction is defined above for finite words, and here there are only infinite word models.

We begin with some basic definitions and results for ω-regular languages [20]. From the definition of Büchi acceptance, every nonempty ω-regular language contains an infinite word $w \overset{def}{=} uvv\ldots$, for u and v finite words in Σ^*. If uv is of length k, we say the word w *contains* (or *is*) *a lasso of length* k, consisting of a *prefix* u and a *loop* v. Consequently, a S1S formula ϕ is satisfiable iff it has a satisfying word model that is a lasso of length k, for some $k \in \mathbb{N}$.

Using the fact that lassos can be finitely represented, we define an analog of BMC for S1S.

Definition 6
Bounded Model Construction for S1S *(*BMC(S1S)*)*
Instance: *A formula ϕ and a natural number k.*
Parameter: *k.*
Question: *Does ϕ have a satisfying lasso of length k in* S1S*?*

We now prove that no elementary BMC procedure for S1S exists.

Theorem 4 BMC(S1S) *is nonelementary.*

Proof. Our proof uses an embedding of WS1S in S1S. For this, we first show that finiteness is definable in S1S. Let finite and Finite be the following two propositions:

$$\mathsf{finite}(X) \overset{def}{=} \exists m.\, \forall p.\, X(p) \to p \leq m \text{ and } \mathsf{Finite}(\phi) \overset{def}{=} \bigwedge_{X \in freevars(\phi)} \mathsf{finite}(X)\,.$$

We define an embedding function $[\cdot]$ from WS1S into S1S by $[\phi] = \mathsf{Finite}(\phi) \wedge \lceil \phi \rceil$, where $\lceil \phi \rceil$ is:

$$\lceil \mathrm{s}(t,t') \rceil = \mathrm{s}(t,t') \qquad \lceil X(t) \rceil = X(t) \qquad \lceil \exists p.\, \phi \rceil = \exists p.\, \lceil \phi \rceil$$

$$\lceil \neg \phi \rceil = \neg \lceil \phi \rceil \qquad \lceil \phi_1 \vee \phi_2 \rceil = \lceil \phi_1 \rceil \vee \lceil \phi_2 \rceil \qquad \lceil \exists X.\, \phi \rceil = \exists X.\, \mathsf{finite}(X) \wedge \lceil \phi \rceil$$

We can show by induction over the structure of formulae that $[\cdot]$ preserves satisfiability. Namely, a formula ϕ has a word model of length k in WS1S iff $[\phi]$ has a satisfying lasso of length k in S1S.

The function that assigns to each BMC(WS1S)-instance (ϕ, k) the BMC(S1S)-instance $([\phi], k)$ reduces, in polynomial time, BMC(WS1S) to BMC(S1S). Using Theorem 3, the claim follows. $\square$

Again, we can apply the same proof idea to other monadic logics. Let FO[<] be the first-order fragment of S1S augmented by the less relation <. A similar proof establishes that bounded model construction for FO[<], *i.e.*, BMC(FO[<]), is nonelementary by reducing BMC(WFO[<]) to BMC(FO[<]).

5 Experimental Results

We have implemented bounded model construction for M2L-Str and describe here experimental results. Our system takes as input a natural number and a formula written in a "sugared" version of the syntax of §3.1. It first calculates from the inputs a QBL formula as described in §4.1. Second, it transforms the QBL formula into Boolean logic by eliminating the universal quantifiers (replacing $\forall x.\, \phi$ with $\phi[\mathsf{true}/x] \wedge \phi[\mathsf{false}/x]$) and dropping the remaining existential quantifiers (assuming all bound variables are uniquely named). Third, the resulting Boolean formula is converted into conjunctive normal form. Finally, the result is tested for satisfiability using the Sato system [23], which is an efficient implementation of the Davis-Putnam procedure. Of course, with minor changes other satisfiability checkers could be used.

We used the Mona system [14] for comparison. Mona is an automata-based implementation of decision procedures for the monadic logics M2L-Str, WS1S and their generalizations to trees. Mona compiles a formula into a minimal deterministic automaton, which it represents and manipulates using BDD's. Over the last few years Mona has been continually improved and is now highly optimized (we use version 1.4). For our tests, we used a 450 MHZ Sun Sparc Ultra workstation.

For completeness, we tested examples ranging from those that are easy for Mona to those that are difficult. Table 1 presents tests on several easy examples. In all of these, we find counter-examples (of the same length as Mona's) when they exist. However, for these examples, Mona is much faster.

The first example is a parameterized *n-bit ripple-carry adder*, taken from [2]. The input formula states the equivalence between a structural description of the parameterized adder family (described at the gate level) with a behavioral description, describing how bit-strings are added. We checked this equivalence for $k = 2$, 4, and 6. The second example involves a structural specification of a sequential *D-type flip-flop* circuit, and its behavioral model. The circuit is built from 6 *nand*-gates, each of which has a (unit) time-delay. We tested the correctness of this circuit with respect to a behavioral description proposed by Gordon in [10]. As has been discovered by [2,22], the specification has a subtle bug. Both Mona and our system find a (different) counter-example of length 8. The third example is a buggy mutual exclusion protocol taken from [3]; both systems successfully find a trace showing that the critical sections can be simultaneously accessed.

Next we consider some examples that are difficult for Mona. First, we consider reasoning about two concurrent processes that increment a shared integer variable N by each executing the program: `Load Reg N`, `Add Reg 1`, `Store Reg N`. If we assume an interleaving semantics, it is possible that N is incremented by either 1 or 2. We model the two parallel processes in M2L-Str and assert (incorrectly) that after execution N is incremented by 1. Table 4 gives the results, where we scale the problem by considering registers of different bit-width. For more than 4 bits, Mona runs out of memory as the automata accepting the computations (traces) of the two systems grow exponentially.

Finally, we consider two sequential circuits: a counter and a barrel shifter, which we parameterize in the width of the data-path. Tables 3 and 2 give the results of these experiments for data-paths of various widths. In the first example, the n-bit counter has two selection lines and n data lines. At each point in time the value of the data lines is incremented, reset or unchanged depending on the value of the selection lines. We verify this with respect to an incorrect specification, which asserts that, after eight time units, the data line is always incremented. In our experiments, Mona quickly runs into state explosion problems, whereas even for large data-paths, we can still generate counter-examples quickly. Our procedure finds that, for data-paths between 4 and 40, the short counter-examples have length $k = 8$. The results for the barrel shifter are similar.

6 Conclusion and Future Work

We have explored the possibility of providing more efficient alternatives to counter-example generation than using standard automata-theoretic decision procedures. We have obtained positive results, both in theory and practice, for

Examples	MONA sec	BMC k	BMC sec
Ripple-carry adder	0.06	1	0.05
		4	0.30
		6	8.85
Buggy ripple-carry adder	0.11	5	0.82
FlipFlop	0.19	7	0.23
Buggy mutual exclusion	0.20	6	0.58

Table 1. Simple Examples

	MONA		BMC ($k = 6$)	
bit	sec	MB	sec	MB
1	0.05	0	0.04	0
4	0.07	0	0.12	0
8	1.00	11	0.39	0
16	abort		1.79	11
32	-		10.00	53

Table 2. Barrel Shifter

	MONA		BMC ($k = 8$)	
bit	sec	MB	sec	MB
4	0.15	0	0.16	0
8	0.90	0	0.49	11
12	23.65	67	0.85	22
15	242.17	734	1.25	33
16	abort		1.39	38
24	-		3.14	88
32	-		6.36	167
40	-		11.00	284

Table 3. Counter

$N \leq 2^i$	MONA	BMC ($k = 6$)
i	sec	sec
2	0.60	0.49
3	5.48	0.85
4	57.18	1.60
5	abort	3.60
6	-	9.16
7	-	30.85
8	-	105.45

Table 4. Parallel Instruction

M2L-STR, and negative results for WS1S, S1S, WFO[<], and FO[<]. Hence, at least for counter-example generation, M2L-STR is the superior choice.

The theoretical issues seem clear-cut. On the experimental side there is still work to do. In our experiments, most of the time is spent translating QBL formulae into Boolean logic formulae and the resulting normal form calculation. Recent work on testing satisfiability for QBL [7] could offer improvements here; investigating this remains as future work.

References

[1] K. R. Abrahamson, J. A. Ellis, M. R. Fellows, and M. E. Mata. On the complexity of fixed parameter problems (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*. IEEE, 1989.

[2] D. Basin and N. Klarlund. Automata based symbolic reasoning in hardware verification. *The Journal of Formal Methods in Systems Design*, 13(3), November 1998.

[3] M. Ben-Ari. *Principles of Concurrent and Distributed Programming*. Prentice Hall, 1990.

[4] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *TACAS'99*, volume 1579 of *LNCS*. Springer, 1999.

[5] J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6, 1960.

[6] J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Proc. 1960 Int. Congr. for Logic, Methodology, and Philosophy of Science*. Stanford Univ. Press, 1962.

[7] H. K. Büning, M. Karpinski, and A. Flögel. Resolution for quantified Boolean formulas. *Information and Computation*, 117(1), 1995.

[8] S. A. Cook. The complexity of theorem-proving procedures. In *Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, Ohio, 3–5 1971 1971.

[9] C. C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the AMS*, 98, 1961.

[10] M. J. Gordon. Why higher-order logic is a good formalism for specifying and verifying hardware. In G. J. Milne and P. A. Subrahmanyam, editors, *Formal Aspects of VLSI Design*. North-Holland, 1986.

[11] J. G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, B. Paige, T. Rauhe, and A. Sandholm. Mona: Monadic second-order logic in practice. In *TACAS '95*, volume 1019 of *LNCS*, 1996.

[12] P. Kelb, T. Margaria, M. Mendler, and C. Gsottberger. Mosel: A sound and efficient tool for M2L(Str). In *CAV'97*, volume 1254 of *LNCS*. Springer–Verlag, 1997.

[13] N. Klarlund. A theory of restrictions for logics and automata. In *CAV '99*, volume 1633 of *LNCS*. Springer, 1999.

[14] N. Klarlund and A. Møller. *MONA Version 1.3 User Manual*. BRICS Notes Series NS-98-3 (second revision), Department of Computer Science, University of Aarhus, 1998.

[15] Z. Manna, N. Bjoerner, A. Browne, and E. Chang. STeP: The Stanford Temporal Prover. In *TAPSOFT'95: Theory and Practice of Software Development*, volume 915, 1995.

[16] A. Meyer. Weak monadic second-order theory of successor is not elementary-recursive. In *LOGCOLLOQ: Logic Colloquium*, volume 453. Springer-Verlag, 1975.

[17] A. R. Meyer and L. J. Stockmeyer. Word problems requiring exponential time. In *ACM Symposium on Theory of Computing*, pages 1–9, New York, April 1973. ACM Press.

[18] F. Morawietz and T. Cornell. On the recognizability of relations over a tree definable in a monadic second order tree description language. Research Report SFB 340-Report 85, Sonderforschungsbereich 340 of the Deutsche Forschungsgemeinschaft, 1997.

[19] H. Straubing. *Finite automata, formal logic, and circuit complexity*. Birkhäuser, 1994.

[20] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 4. Elsevier Science Publishers B. V., 1990.

[21] W. Thomas. Languages, automata and logic. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, Beyond Words. Springer-Verlag, Berlin, 1997.

[22] A. Wilk and A. Pnueli. Specification and verification of VLSI systems. In *IEEE/ACM International Conference on Computer-Aided Design*, 1989.

[23] H. Zhang. SATO: An efficient propositional prover. In *CADE'97*, volume 1249 of *LNAI*. Springer, 1997.