Efficient Symmetry Breaking for SAT-Based Minimum DFA Inference

Ilya Zakirzyanov[1,2](✉), Antonio Morgado[3], Alexey Ignatiev[3,4], Vladimir Ulyantsev[1], and Joao Marques-Silva[3]

[1] ITMO University, St. Petersburg, Russia
{zakirzyanov,ulyantsev}@corp.ifmo.ru
[2] JetBrains Research, St. Petersburg, Russia
[3] Faculty of Science, University of Lisbon, Lisbon, Portugal
{ajmorgado,aignatiev,jpms}@ciencias.ulisboa.pt
[4] ISDCT SB RAS, Irkutsk, Russia

Abstract. Inference of deterministic finite automata (DFA) finds a wide range of important practical applications. In recent years, the use of SAT and SMT solvers for the minimum size DFA inference problem (MinDFA) enabled significant performance improvements. Nevertheless, there are many problems that are simply too difficult to solve to optimality with existing technologies. One fundamental difficulty of the MinDFA problem is the size of the search space. Moreover, another fundamental drawback of these approaches is the encoding size. This paper develops novel compact encodings for Symmetry Breaking of SAT-based approaches to MinDFA. The proposed encodings are shown to perform comparably in practice with the most efficient, but also significantly larger, symmetry breaking encodings.

Keywords: DFA inference · Boolean satisfiability · Symmetry breaking

1 Introduction

The inference of minimum-size deterministic finite automata (DFA) from (positive and negative) examples of their behavior has been investigated since the early days of computing, with continued improvements until the present day. The importance of topic is illustrated not only by recent improvements to tools for computing minimum-size DFAs [27,30], but also by recent and ever growing list of applications [29]. The problem of computing the minimum-size

IZ was supported by RFBR (project 18-37-00425). AM, AI and JMS were supported by FCT grants ABSOLV (PTDC/CCI-COM/28986/2017), FaultLocker (PTDC/CCI-COM/29300/2017), SAFETY (SFRH/BPD/120315/2016), and SAMPLE (CEECIND/04549/2017). VU was supported by the Government of Russia (Grant 08-08).

DFA (MinDFA) witnessed seminal work in the early 70s [6]. Moreover, a number of visible contributions were made in the 90s. These include the use of graph coloring [8], constraint programming techniques [9,22], and state merging approaches [17,18]. Approaches based on SAT and SMT were proposed in the last decade, with promising results [12,13,20,21,25]. Nevertheless, the size of existing propositional encodings do not scale for large DFA inference problems. The use of SMT does not represent a clear improvement, since SMT solving approaches for the MinDFA problem will also encode to propositional logic. This paper revisits SAT encodings for the MinDFA problem as well as recent work on exploiting symmetry breaking [25,30], and proposes a (novel) tighter propositional representation of state-of-the-art symmetry breaking predicates, but it also devises new symmetry breaking constraints which serve to achieve more effective pruning of the search space. The new propositional encoding proposed in this paper enables clear performance gains over the state of the art [13,14,26,30].

The paper is organized as follows. Section 2 introduces the definitions used throughout the paper and briefly overviews related work. Section 3 develops new ideas to encode symmetry breaking predicates. Section 4 compares a new tool for the MinDFA problem with the existing state of the art, showing clear performance gains. Section 5 concludes the paper.

2 Background

2.1 Preliminaries

Throughout the paper we assume that automata are defined over some set of symbols Σ, also known as the *alphabet*. The number of symbols in the alphabet is $L = |\Sigma|$. For earlier DFA inference examples, it was often the case that $\Sigma = B = \{0, 1\}$ [18,22]. For more recent DFA inference examples [28], larger alphabets are often considered.

A *deterministic finite automaton* (DFA) is a tuple $\mathcal{D} = (D, \Sigma, \delta, d_1, D^+, D^-)$, where D is a finite set of states, Σ is the (input) alphabet, $\delta : D \times \Sigma \to D$ is the transition function, d_1 is the initial state, D^+ is the set of accepting states and $D^- = D \setminus D^+$ is the set of rejecting sets. For input strings $\pi \in \Sigma^*$ we define $\hat{\delta}(d_1, \pi)$ inductively as follows [16]: (i) $\hat{\delta}(d_1, \epsilon) = d_1$; (ii) If $\pi = \pi' c$, then $\hat{\delta}(d_1, \pi) = \delta(\hat{\delta}(d_1, \pi'), c)$.

We assume the standard setting of inferring a minimum-size DFA given a set of samples of its behavior [7,15], i.e. the training set, each sample represented by an input string that is either accepted or rejected by some DFA $\mathcal{U} = (U, \Sigma, \mu, u_1, U^+, U^-)$, which is not known. This form of learning is often referred to as *passive learning*, as opposed to *active learning* [2,20], which enables a learning algorithm (aiming to create a target DFA) to formulate queries to some teacher (which knows of the unknown DFA).

A *training set* is a set of pairs $\mathbb{T} = \{(\pi_1, o_1), \ldots, (\pi_R, o_R)\}$, where each pair $(\pi_r, o_r) \in \Sigma^* \times \{0, 1\}$ denotes the output o_r observed given input string π_r. If $o_r = 1$ ($o_r = 0$), then π_r is referred to as a *positive* (*negative*) example. Given

```
Function MinimumDFA($\mathcal{T}$)
    Input  : $\mathcal{T}$: APTA
    Output: $\mathcal{S}$: minimum size DFA
1   $M \leftarrow$ FindLowerBound($\mathcal{T}$)
2   while true do
3       $\mathcal{S} \leftarrow$ FindConsistentDFA($\mathcal{T}, M$)
4       if $\mathcal{S} \neq \emptyset$ then return $\mathcal{S}$
5       $M \leftarrow M + 1$
```

Algorithm 1: General lower bound refinement algorithm

a training set, we can construct an APTA (*augmented prefix tree acceptor*) [1, 13, 24], defined as the DFA $\mathcal{T} = (T, \Sigma, \tau, t_1, T^+, T^-)$, where any input string sharing the same prefix ends up in the same state. Concretely, given input strings $\pi_1 = \pi_a\pi_{b_1}$ and $\pi_2 = \pi_a\pi_{b_2}$ the common prefix π_a will be associated to a unique sequence of states in the APTA. For an APTA $\mathcal{T}$, we have $T^+ \cup T^- \neq T$, and we define $N = |T|$. When clear from the context, the states of $\mathcal{T}$ are referred to by their index, t_i by i, $i = 1, \ldots, N$. In some settings, $\theta(i)$ will be used to denote the distance from the APTA root state t_1 to state t_i.

The *minimum-size DFA* inference problem (MinDFA) is to identify a DFA $\mathcal{S} = (S, \Sigma, \sigma, s_1, S^+, S^-)$, with a minimum number of states, such that for any training pair (π_r, o_r), $\hat{\sigma}(s_1, \pi_r) \in S^+$ iff $o_r = 1$ and $\hat{\sigma}(s_1, \pi_r) \in S^-$ iff $o_r = 0$. For a prospective DFA $\mathcal{S}$, we define $M = |S|$.

Throughout the paper $[R]$ is used to denote the set $\{1, \ldots, R\}$, for some positive integer R. Moreover, we will use integers to refer to either symbols or states. For a given alphabet, by associating states and symbols with integers facilitates imposing a fixed lexicographic order, which will be required later in the paper (see Sect. 3). Additionally, standard SAT definitions are assumed and used [5].

2.2 Minimum Size DFA Inference

This paper focuses on constraint-based exact approaches for the MinDFA problem. Different constraint programming approaches for solving the MinDFA problem have been proposed over the years. More recently, the use of SAT [12–14] and SMT [20, 21] has been investigated. A more detailed account of past work is available for example in Neider's PhD thesis [20, Chap. 3].

Algorithm 1 summarizes the most widely used approach for computing a minimum size DFA consistent with a given APTA $\mathcal{T}$ (obtained from the training set). Initially a lower bound on the size of the inferred DFA is computed. An often used heuristic is to compute a maximal clique on states of the APTA that cannot be assigned to the same DFA state [12–14, 20–22, 26]. Afterwards, starting from the lower bound and for each possible value on the number of states of the DFA, some algorithm decides whether there exists a DFA $\mathcal{S}$ which can be shown consistent with the samples of behavior summarized as the APTA $\mathcal{T}$.

Algorithm 1 is referred to as LSUS (linear-search, UNSAT until SAT) and is used in different settings. Other algorithms can be envisioned. These include binary search, assuming some upper bound is known or can be identified (e.g. with merge-based algorithms). Another alternative is unbounded search with a final binary search step. These algorithms have been used in recent years for solving MaxSAT [19] and for extracting MUSes [4]. The use of propositional encodings can be traced to the work of Grinchtein, Leucker & Piterman [12]. By using two different representations for integers, one in unary and the other in binary, this work proposes two propositional encodings. For the unary representation, the encoding size is in $\mathcal{O}(N \times M^2 + N^2 \times M)$ over $\mathcal{O}(N \times M)$ variables[1]. For the binary representation, the encoding size is in $\mathcal{O}(N \times M \times \log M + N^2 \times M)$ on $\mathcal{O}(N \times \log M)$ variables. More recent work by Heule&Verwer (HV) [13,14] proposed encodings that have been shown effective in practice [28]. The HV encoding builds on the graph coloring analogy proposed in earlier work [8]. The proposed encoding has size $\mathcal{O}(M^3 + N \times M^2)$ over $\mathcal{O}(M^2 + N \times M)$ variables. This encoding is revisited in Sect. 2.3.

2.3 SAT-Based MinDFA

Given an APTA $\mathcal{T}$ and a bound M on the number of states of the inferred DFA $\mathcal{S}$, this subsection provides a derivation of the HV encoding [13,14], based on a different motivation. By careful analysis of this formulation, we achieve a more compact propositional encoding. Instead of relating the MinDFA problem with graph coloring, we formulate it as the problem of matching the N states of the APTA $\mathcal{T}$ to the M states of a target DFA $\mathcal{S}$. The sets of variables of the propositional encoding are as follows:

1. $m_{i,p}$ which is 1 iff state t_i in $\mathcal{T}$ is matched with state s_p in $\mathcal{S}$.
2. $e_{v,p,q}$ which is 1 iff there is a transition from s_p to s_q on symbol l_v in $\mathcal{S}$.
3. a_p which is 1 iff s_p is accepting in $\mathcal{S}$.

The constraints of the proposed encoding are summarized in Table 1. Observe that for encoding the Equals1 constraints, [14] uses a clause to encode an AtLeast1 constraint, and the Pairwise Encoding for encoding an AtMost1 constraint. A simple improvment is to use a more compact encoding, among the many that exist. Concrete examples include sequential counters [23], cardinality networks [3], the ladder encoding [11], sorting networks [10], among several other options. As can be concluded, the proposed encoding grows with $\mathcal{O}(N \times M^2)$. Thus, the encoding is asymptotically (somewhat) tighter than the encoding proposed in [13], in that the encoding of the cardinality constraints changes from $\mathcal{O}(M^3)$ to $\mathcal{O}(M^2)$. This difference can be significant for large values of M. As observed in earlier work [13,14], for some benchmarks [18], the target DFA has hundreds of states, and so an encoding in $\mathcal{O}(M^3)$ is expected to be beyond the memory capacity of existing compute servers. It is straightforward to map the

[1] The encoding size shown is adapted from the results in [20], taking into account that both $|T^+|$ and $|T^-|$ can grow with $N = |T|$. The size of $|\Sigma|$ is assumed constant.

Table 1. Constraints of the SAT encoding

Constraint	Range	
$(\sum_{p=1}^{M} m_{i,p}) = 1$	$i \in [N]$	Each state t_i in $\mathcal{T}$ is matched with exactly one state in $\mathcal{S}$
$m_{i,p} \rightarrow a_p$	$i \in [N]$; $t_i \in \mathcal{T}^+$; $p \in [M]$	Each accepting state t_i in $\mathcal{T}$ is matched with an accepting state in $\mathcal{S}$
$m_{i,p} \rightarrow \neg a_p$	$i \in [N]$; $t_i \in \mathcal{T}^-$; $p \in [M]$	Each rejecting state t_i in $\mathcal{T}$ is matched with a rejecting state in $\mathcal{S}$
$(\sum_{q=1}^{M} e_{v,p,q}) = 1$	$v \in [L]$; $p \in [M]$	There is exactly one transition from s_p on some symbol l_v in $\mathcal{S}$
$m_{i,p} \wedge m_{k,q} \rightarrow e_{v,p,q}$	$i, k \in [N]$; $v \in [L]$; $\sigma(t_i, l_v) = t_k$; $p, q \in [M]$	A transition between t_i and t_k on l_v in $\mathcal{T}$ forces a transition between its mapped nodes on the same l_v in $\mathcal{S}$
$m_{i,p} \wedge e_{v,p,q} \rightarrow m_{k,q}$	$i, k \in [N]$; $v \in [L]$; $\sigma(t_i, l_v) = t_k$; $p, q \in [M]$	A transition between t_i and t_k on l_v in $\mathcal{T}$, with a transition between the mapped state p and a state q on l_v in $\mathcal{S}$, forces a mapping between t_k and q

sets of clauses in the HV formulation [13,14] into the constraints described above. The main difference is that we explicitly use a tighter encoding for the AtMost1 constraints, which are listed as sets of clauses (capturing the well-known pairwise encoding) in [13]. Additionally, the HV formulation [13] considers different sets of redundant constraints to the basic formulation above. A technique that has been proposed for the SAT formulation is the breaking of symmetries of the DFA constructed [26,30]. Symmetry breaking for the SAT formulation is described in depth in Sect. 3, together with new improvements.

3 Efficient Symmetry Breaking

This section revisits recent symmetry breaking for the MinDFA problem, which imposes an order on the states of the DFA [26,30]. Although effective in practice, the existing propositional encoding is not tight, and so unlikely to scale for larger DFAs. Section 3.2 develops a significantly tighter encoding. Section 3.3 devises novel constraints that serve to furhter prune the search space that a SAT solver needs to explore.

3.1 Propositional Formulation for Breaking Symmetries

This section summarizes the recent work on breaking symmetries of the DFA being constructed, by imposing an ordering on the states of the DFA [26,30]. In this section we follow the original formulation [26]. The approach can be formalized as follows. Assume a target DFA $\mathcal{S} = (S, \Sigma, \sigma, s_1, S^+, S^-)$. The states of the DFA $\mathcal{S}$ are required to be numbered according to the tree induced by a breadth-first search (BFS) of the target DFA. As a result, the formulation of symmetry breaking depends only on the states and transitions of the target DFA $\mathcal{S}$ (independent of the APTA $\mathcal{T}$). In this section we require some fixed (e.g. lexicographic) ordering on the symbols of Σ. Any order of the symbols is valid. The symbols will be numbered from 1 to L, but the numbers respect the fixed ordering.

The propositional variables used in the formulation are as follows:

1. $p_{q,r}$, with $1 \leq r < q \leq M$. $p_{q,r} = 1$ iff state r is the parent of q in the BFS tree.
2. $t_{p,q}$, with $1 \leq p < q \leq M$. $t_{p,q} = 1$ iff there is a transition from p to q in $\mathcal{S}$.
3. $m_{v,p,q}$, with $v \in \Sigma$ and $1 \leq p < q \leq M$. $m_{v,p,q} = 1$ iff there is a transition from state p to state q on symbol l_v and there is no such transition with a lexicographically smaller symbol.

The clauses of the propositional formulation are summarized in Eqs. (1–6).

$$\textstyle\bigwedge_{2 \leq q \leq M} (p_{q,1} \vee p_{q,2} \vee \cdots \vee p_{q,q-1}) \tag{1}$$
$$\textstyle\bigwedge_{1 \leq r < s < q < M} (p_{q,s} \rightarrow \neg p_{q+1,r}) \tag{2}$$
$$\textstyle\bigwedge_{1 \leq r < q \leq M} (t_{r,q} \leftrightarrow e_{1,r,q} \vee \cdots \vee e_{L,r,q}) \tag{3}$$
$$\textstyle\bigwedge_{1 \leq r < q \leq M} (p_{q,r} \leftrightarrow t_{r,q} \wedge \neg t_{r-1,q} \wedge \cdots \wedge \neg t_{1,q}) \tag{4}$$
$$\textstyle\bigwedge_{1 \leq r < q \leq M} \bigwedge_{1 \leq v \leq L} (m_{v,r,q} \leftrightarrow e_{v,r,q} \wedge \neg e_{v-1,r,q} \wedge \cdots \wedge \neg e_{1,r,q}) \tag{5}$$
$$\textstyle\bigwedge_{1 \leq r < q < M} \bigwedge_{1 \leq u < v \leq L} (p_{q,r} \wedge p_{q+1,r} \wedge m_{v,r,q} \rightarrow \neg m_{u,r,q+1}) \tag{6}$$

There are six types of conjunction of clauses considered. (1) relates to the states, and with the exception of the initial state (numbered 1), each clause says that a state must have a parent with smaller number. (2) says that a state q must be enqueued (in the BFS traversal) before the next state $q+1$, and so the parent r of $q+1$ cannot be less than the parent s of q. (3) and (4) define the $t_{q,r}$ variables based on the $e_{v,q,r}$ variables and relate them to the parent variables $p_{q,r}$. (5) defines the $m_{v,p,q}$ variables using DFA transitions, and the (6) imposes consecutive states q and $q+1$ with the same parent r to be arranged in the order of the symbols. It is plain to conclude that the size of the encoding grows with $\mathcal{O}(M^3 + M^2L + M^2L^2)$. Observe that the contribution of M^3, which dominates the other components assuming $L \ll M$, results from (2) and (4). Moreover, when $|\Sigma| = 2$, [26] proposes to replace (5) and (6) with

$$\bigwedge_{1 \leq r < q < M} (p_{q,r} \wedge p_{q+1,r} \rightarrow e_{1,r,q}) \tag{7}$$

3.2 A Tighter SAT Encoding

A propositional encoding in $\mathcal{O}(M^3 + M^2L^2 + M^2L)$ is impractical for the larger DFA inference instances [18,28]. This section shows how to modify the symmetry breaking propositional encoding of Sect. 3.1 such that the encoding size becomes $\mathcal{O}(M^2L)$. The new encoding develops alternative representations for (2) and the (4), but also for (5) and (6). In addition, one needs to require:

$$\textstyle\sum_{r=1}^{q-1} p_{q,r} = 1 \;\; 1 < q \leq M \tag{8}$$

We first investigate the encoding of (2) and (4). We can view the values of $p_{q,r}$, with $1 \leq r \leq q-1$, as a binary string, with $q-1$ bits, and compare this string with the one of $p_{q+1,r}$, with $1 \leq r \leq q$, and so with q bits. We introduce $p_{q,q} = 0$, and so can also view the values of $p_{q,r}$ as a binary string with q bits (same size).

Observe that (2) encodes the value associated to the binary string of the $p_{q,r}$ variables to be smaller or equal than the value associated to the binary string of the $p_{q+1,r}$ variables. To compare the binary strings, we inspect the bits in order, starting at position q, and moving down to position 1. We consider variables $ng_{q,r}$, such that $ng_{q,r} = 1$ iff the most significant $q-r+1$ bits of the string associated with $p_{q,r}$ are lexicographically *no greater* than those of $p_{q+1,r}$. The value associated to the binary string of the $p_{q,r}$ variables is smaller or equal than the value associated to the binary string of the $p_{q+1,r}$ variables iff $ng_{q,1} = 1$ holds. Since we enforce $p_{q,q} = 0$, then we must have $ng_{q,q} = 1$. Moreover, we also require $ng_{q,1} \leftrightarrow 1$. Thus we obtain:

$$(ng_{q,1} \leftrightarrow 1) \wedge (ng_{q,q} \leftrightarrow 1) \wedge \bigwedge_{1 \leq r < q} (ng_{q,r} \leftrightarrow ng_{q,r+1} \wedge eq_{q,r} \vee p_{q,r} \wedge \neg p_{q+1,r}) \tag{9}$$

where, $eq_{q,r} \leftrightarrow (p_{q,r} \leftrightarrow p_{q+1,r})$.

Second, a similar approach can be exploited for encoding of (4). We introduce variables $nt_{r,q}$, where $nt_{r,q} = 1$ iff there exists *no* $t_{s,q} = 1$ with $s < r$. Thus, $nt_{r,q}$ can be defined inductively as follows:

$$(nt_{0,q} \leftrightarrow 1) \wedge \bigwedge_{1 < r < q} (nt_{r,q} \leftrightarrow nt_{r-1,q} \wedge \neg t_{r,q}) \tag{10}$$

Thus, (4) can be rewritten, using the $nt_{r,q}$ variables as follows:

$$p_{q,r} \leftrightarrow t_{r,q} \wedge nt_{r-1,q} \tag{11}$$

As can be concluded, by using auxiliary variables $ng_{q,r}$ and $nt_{r,q}$, and Eqs. (9), (10) and (11), we achieve an overall propositional encoding in $\mathcal{O}(M^2L + M^2L^2)$.

However, we can tighten further the propositional encoding for breaking symmetries using a BFS tree. This is achieved by devising alternative encodings for (5) and (6). As shown next, this yields a propositional encoding in $\mathcal{O}(M^2L)$. With respect to (5), we use the additional variables $ne_{v,r,q}$ such that $ne_{v,r,q} = 1$ iff all variables $e_{u,r,q} = 0$ with $u < v$, i.e. there are *no* variables $e_{u,r,q}$ taking value 1, when $u < v$.

$$(ne_{1,r,q} \leftrightarrow \neg e_{1,r,q}) \wedge \bigwedge_{1 < v < L} (ne_{v,r,q} \leftrightarrow \neg e_{v,r,q} \vee ne_{v-1,r,q}) \tag{12}$$

Thus given (12), (5) can be rewritten as follows:

$$\bigwedge_{1 \le r < q \le M} \bigwedge_{1 \le v \le L} (m_{v,r,q} \leftrightarrow e_{v,r,q} \wedge ne_{v-1,r,q}) \tag{13}$$

With respect to (6), we use the additional variables $zm_{v,r,q}$ such that $zm_{v,r,q} = 1$ iff all variables $m_{u,r,q}$ are 0-*valued*, $m_{u,r,q} = 0$, for $u < v$.

$$(zm_{1,r,q} \leftrightarrow \neg m_{1,r,q}) \wedge \bigwedge_{1<v<L} (zm_{v,r,q} \leftrightarrow \neg m_{v,r,q} \wedge zm_{v-1,r,q}) \tag{14}$$

Thus given (14), (6) can be rewritten as follows:

$$\bigwedge_{1 \le r < q \le M} \bigwedge_{1 \le v \le L} (p_{q,r} \wedge p_{j+1,r} \wedge m_{v,r,q} \rightarrow zm_{v-1,r,q+1}) \tag{15}$$

One can thus conclude that the resulting propositional encoding size is in $\mathcal{O}(M^2L)$.

3.3 Exploiting BFS-Based Breaking of Symmetries

This section investigates techniques for developing additional constraints when imposing the ordering of states dictated by a BFS tree of the DFA. Figure 1 shows a possible BFS tree illustrating the *largest* state numbers that can be the children of some other state. The additional constraints proposed in this section will relate with Fig. 1.

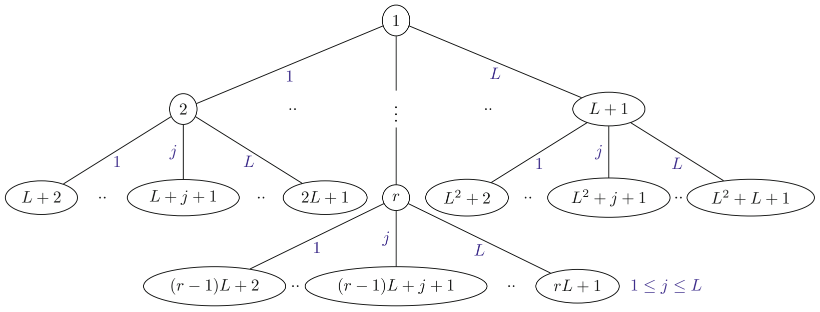

Fig. 1. (Worst case) BFS tree with the largest state numbers that can be the children of some other state. Note that $1 \le j < L$.

BFS-Induced Properties. Although we have introduced $p_{q,r}$ such that $r < q \le M$, it is possible to refine the range of q given r.

Property 1. Given a state r, with $1 \le r \le M$, in the BFS tree, r can be the parent of states in the range $r + 1$ to $rL + 1$.

Figure 1 illustrates the argument for the upper bound on the number of the children of r. We can conclude that the value of $p_{q,r}$ can be non-zero for $r+1 \leq q \leq rL+1$, which also impacts the possible values of some of the $e_{v,r,q}$ and the $t_{r,q}$ variables.

Property 2. For $q > rL+1$ and $v \in [L]$, then $p_{q,r} = 0$, $e_{v,r,q} = 0$, and $t_{r,q} = 0$.

Given that the BFS tree assumes a fixed ordering not only on the states but also on the input alphabet, it is possible to identify other transitions that must be forced to value 0 (based on the ordering of the symbols). Hence, we have the following.

Property 3. $e_{v,r,rL+2-j} = 0$ for $j \in [L-1]$ and $v \in [L-j]$.

The above observations enable to devise the additional constraints described in the remainder of this section. The constraints are organized as *shape* or *range*, but also result from information from the APTA and the BFS distance.

Shape Constraints. The possible values of $p_{q,r}$ respect a *continuity* property, dictated by the BFS traversal, in that all children of r are consecutively numbered, and there can be *at most* L of these. This continuity property can be encoded using additional variables. Let $lnp_{q,r}$ be assigned value 1 iff r is the parent of $q+1$ but not of q (*lnp* stands for *left-no-parent*). Thus,

$$\neg p_{q,r} \wedge p_{q+1,r} \rightarrow lnp_{q,r} \tag{16}$$

Moreover, we have the following:

$$(lnp_{q,r} \rightarrow \neg p_{q,r}) \wedge \bigwedge_{r+1<q\leq M} (lnp_{q,r} \rightarrow lnp_{q-1,r}) \tag{17}$$

Thus, $lnp_{q,r}$ is 1 from $q=1$ until the value of q such that $p_{q+1,r}$ holds.

In a similar fashion, let $rnp_{q,r}$ be assigned value 1 if and only if r is the parent of $q-1$ but not of q (in this case, *rnp* stands for *right-no-parent*). Thus,

$$p_{q-1,r} \wedge \neg p_{q,r} \rightarrow rnp_{q,r} \tag{18}$$

Similarly to the previous case, one can exploit the $rnp_{q,r}$ variables, and derive the following constraints:

$$\begin{array}{ll} rnp_{q,r} \rightarrow rnp_{q+1,r} & r \leq q < M \\ rnp_{q,r} \rightarrow \neg p_{q,r} & \\ rnp_{q,r} \rightarrow \neg e_{v,q,r} & v \in [L] \end{array} \tag{19}$$

Thus, $rnp_{q,r}$ is 1 from $q=M$ until the value of q such that $p_{q-1,r}$ holds.

Another observation is that r can be the parent of at most L states, due to L outgoing transitions. As a result, we get,

$$\begin{array}{ll} p_{q,r} \rightarrow rnp_{q+L,r} & \text{if}\, q+L \leq M \\ p_{q,r} \rightarrow lnp_{q-L,r} & \text{if}\, q-L \geq r+1 \end{array} \tag{20}$$

The $lnp_{q,r}$ and $rnp_{q,r}$ variables serve to force $p_{q,r}$ variables to be assigned value 0. However, under some circumstances, we can infer that some $p_{q,r}$ variables must be assigned value 1. For example, for the range of values of q for which both $lnp_{q,r}$ and $rnp_{q,r}$ are 0, the value of $p_{q,r}$ must be 1. Thus,

$$\neg lnp_{q_1,r} \wedge \neg rnp_{q_2,r} \rightarrow p_{q',r} \qquad \begin{array}{c} q_1 < q' < q_2 \\ q_1 < q_2 \leq \min(q_1 + L - 1, rL + 1, M) \\ r + 1 \leq q_1 < \min(rL + 1, M) \end{array} \tag{21}$$

For any q_1, q_2 can range from $q_1 + 1$ to at most $q_1 + L$. Similarly, we can write,

$$p_{q,r} \wedge p_{s,r} \rightarrow p_{s-1,r} \qquad \begin{array}{c} q < s \leq \min(q + L - 1, rL + 1, M) \\ r + 1 \leq q < \min(rL + 1, M) \end{array} \tag{22}$$

As above, for any r, s can range from $r + 1$ to at most $r + L$.

Range Constraints. Given a reference state r, we have shown above that the states of which r can be a parent of range from $r + 1$ until $rL + 1$. Moreover, we also know there is a continuity property, which causes r to be the parent of at most L states, numbered consecutively. This information can be used for constraining the $p_{q,r}$ variables, between states for which r cannot be a parent, as follows,

$$p_{q,r} \rightarrow \neg p_{q+L,r} \qquad q \in \{l \mid (l \geq r + 1) \wedge (l + L \leq M) \wedge (l + L \leq rL + 1)\} \tag{23}$$

In addition, we get the following stronger condition by directly forcing the value of $e_{v,r,q}$ variables,

$$p_{q,r} \rightarrow \neg e_{v,r,q+L} \qquad \begin{array}{c} q \in \{l \mid (l \geq r + 1) \wedge (l + L \leq M) \wedge (l + L \leq rL + 1)\} \\ v \in [L] \end{array} \tag{24}$$

Furthermore, we can exploit Property 3, and the imposed ordering of the symbols in the BFS to identify a similar extension to (24) as follows,

$$p_{q,r} \rightarrow \neg e_{v,r,q+j} \qquad \begin{array}{c} r + 1 \leq q \leq \min(rL + 1, M) \\ j \in \{l \mid l \in [L - 1] \wedge (q + l \leq M) \wedge (q + l \leq rL + 1)\} \\ v \in [j] \end{array} \tag{25}$$

Minimum BFS Distance. Given the way the BFS vertices are visited, one can guarantee a minimum BFS shortest path distance for each state. For state q, the shortest BFS path length is given by $D_{\min}(q) = \lceil \log_L (q(L - 1) + 1) - 1 \rceil$, with $q > 1$, i.e. no matter how the BFS is organized starting at state 1, the shortest path from 1 to q is never less than $D_{\min}(q)$. As a result, if $D_{\min}(q) > \theta(i)$, then $m_{i,q} = 0$. Observe that, under any possible setting in the DFA, the shortest path to q is larger than the distance to state i in the APTA. Thus, to get to q it would require more transitions that those allowed to get from inital state to i.

Exploiting APTA Information. By exploiting the variables and constraints used for breaking symmetries and using a BFS tree on the target DFA, we can devise additional constraints. Observe that, if the depth of a state i in the APTA is some value K, then in the DFA, we *must* be able to move from 1 to q in K of fewer transitions. However, if the shortest path from 1 to q in the DFA exceeds the depth K of vertex of i in the APTA, then it would be impossible to move from state 1 to state q in K or fewer transitions.

We consider the propositional variables $d_{q,j}$, with $q \in [M]$ and $1 \leq j < q$, such that $d_{q,j} = 1$ iff the length of the shortest path in the BFS tree from state 1 to q is j. Moreover, we consider propositional variables $se_{q,j}$, with $q \in [M]$ and $1 \leq j < q$, such that $se_{q,j} = 1$ iff the length of the shortest path in the BFS tree from state 1 to q is *smaller* than or *equal* to j. We can use an inductive definition for $se_{q,j}$ as follows:

$$se_{q,0} \leftrightarrow 0 \qquad \text{and} \qquad se_{q,j} \leftrightarrow se_{q,j-1} \vee d_{q,j} \tag{26}$$

Similarly to Sect. 3.2, we devise a tight encoding for the definition of the $d_{q,j}$ variables, suitable for larger problem instances. The insight is to introduce additional variables, which are inductively defined. Let $er_{q,r,j}$ be such that $er_{q,r,j} = 1$ iff there *exists* some index $r < q$ such that $p_{q,r} = 1$ and $d_{r,j} = 1$.

$$\begin{aligned} &er_{q,r,j} \leftrightarrow p_{q,r} \wedge d_{r,j} \vee er_{q,r+1,j} \qquad j < r < q-1 \\ &er_{q,q-1,j} \leftrightarrow p_{q,q-1} \wedge d_{q-1,j} \end{aligned} \tag{27}$$

we can now derive constraints on the $m_{i,p}$ variables. Let t_i be a state of the APTA such that the depth of t_i is I. We can define $d_{q,j}$ as follows:

$$d_{q,j} \leftrightarrow \neg se_{q,j-1} \wedge er_{q,j,j-1} \tag{28}$$

$$\neg se_{q,I} \rightarrow \neg m_{i,q} \tag{29}$$

One can conclude that the modified constraints have an encoding size in $\mathcal{O}(N \times M^2)$.

4 Experimental Results

This section evaluates the ideas described above, namely a compact SAT encoding and symmetry breaking predicates for solving the MinDFA problem. For this, the ideas were implemented on top of a known MinDFA solver called *DFA-Inductor* [26,30] written in Java[2]. The new prototype is referred to as *DFA-Inductor 2*. For comparison, two competitors were considered: the original DFA-Inductor and also *dfasat* [13]. All the selected tools apply the Glucose 4.1[3] SAT solver iteratively and *non-incrementally*, i.e. each call to the oracle is made *from scratch*. All the conducted experiments were performed in Ubuntu Linux on an AMD Opteron 6378 2.40 GHz processor with 496GByte of memory. For

[2] https://github.com/ctlab/DFA-Inductor.

[3] http://www.labri.fr/perso/lsimon/glucose/.

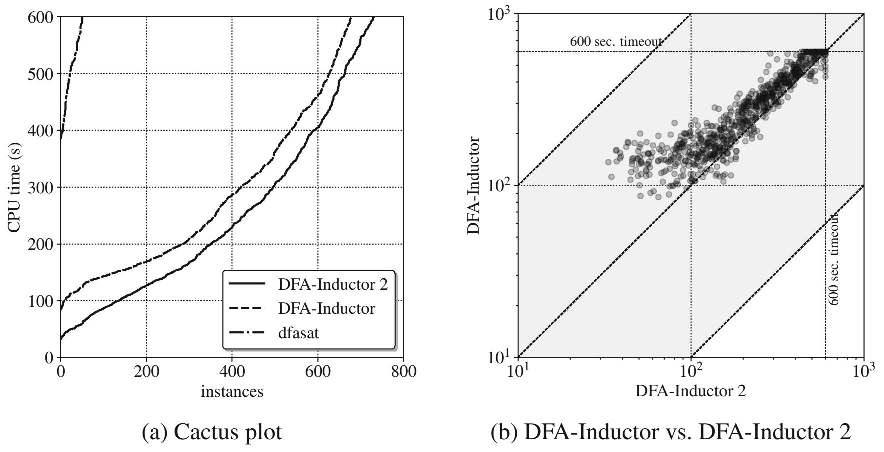

Fig. 2. Detailed performance of dfasat, DFA-Inductor, and DFA-Inductor 2

each individual process, the time limit was set to 600 s and the memory limit to 1 GByte. For the comparison, a number of benchmark instances were randomly generated, following the procedure described in [30]. Concretely, starting from a randomly generated APTA of *even* size N, $N \in [20, 36]$, $50 \times N$ samples were generated. The size of the Σ is two. For each even number $N \in [20, 36]$, exactly 100 benchmark instances were created such that given value N, the resulting DFA for each of the corresponding 100 instances is guaranteed to be N. This way, the number of benchmark families defined by values N is 9. Thus, the total number of instances considered is 900. Figure 2a shows a cactus plot depicting the performance of all the selected solvers. As one can observe, dfasat is significantly outperformed by the compact encoding implemented in DFA-Inductor. In total, dfasat is able to solve only 51 benchmark instances (out of 900). Also observe that the symmetry breaking predicates described above further improve the performance of DFA-Inductor (see DFA-Inductor 2 compared to DFA-Inductor in the Fig. 2a). A comparison between DFA-Inductor and DFA-Inductor 2 is detailed in Fig. 2b and also in Table 2. Except for a few outliers, the symmetry breaking predicates of DFA-Inductor 2 are responsible for 20–40% performance improvement on average. Also it is important to note that the harder the problems are, the smaller is the performance gap between the two configurations. Although this can be seen as a drawback, the phenomenon requires further investigation on the use of symmetry breaking with various SAT solvers and a multitude of families benchmark sets. In total, the number of instances solved by DFA-Inductor and DFA-Inductor 2 is 678 and 731, respectively, thus, comprising a gap of 53 benchmark instances. Therefore, symmetry breaking brings more 7.2% instances solved.

Table 2. The effect of applying the symmetry breaking predicates described above. The solver configuration using the proposed symmetry breaking is referred to as *DFA-Inductor 2* and compared to the base configuration, i.e. *DFA-Inductor*. If an instance is timed out, its contribution to the average time of the corresponding benchmark family is assumed to be 600 s. The corresponding values are written in *italic*.

N	DFA-Inductor				DFA-Inductor 2			
	min	*avg*	*max*	*# solved*	*min*	*avg*	*max*	*# solved*
20	86.8	148.3	221.0	**100**	33.3	91.9	228.4	**100**
22	85.5	*147.1*	—	**99**	49.2	*100.4*	—	**99**
24	128.6	181.5	287.8	**100**	80.4	136.8	262.5	**100**
26	158.1	*251.8*	—	**99**	114.8	*209.3*	—	**99**
28	223.4	317.9	534.5	**100**	164.2	*268.9*	—	99
30	307.2	*443.8*	—	91	227.1	*389.2*	—	**95**
32	326.0	*506.5*	—	76	249.2	*447.4*	—	**86**
34	414.5	*591.1*	—	13	392.1	*569.9*	—	**41**
36	—	*600.0*	—	0	448.4	*594.8*	—	**12**

5 Conclusions

This paper proposes a number of novel techniques for encoding and reasoning about symmetries when exploiting SAT oracles for inferring minimum-size DFAs. The experimental results provide evidence of the improvements that can be achieved when compared with the state of the art [26,30], also enabling significant gains over the best exact methods proposed in recent years [13]. The novel symmetry-breaking ideas described in the paper can be applied to other approaches for inferring minimum-size DFAs, including the use of SMT solvers [20], and also in other settings.

References

1. Abela, J., Coste, F., Spina, S.: Mutually compatible and incompatible merges for the search of the smallest consistent DFA. In: ICGI, pp. 28–39 (2004)
2. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987)
3. Asín, R., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: Cardinality networks: a theoretical and empirical study. Constraints **16**(2), 195–221 (2011)
4. Belov, A., Lynce, I., Marques-Silva, J.: Towards efficient MUS extraction. AI Commun. **25**(2), 97–116 (2012)
5. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press, Amsterdam (2009)

6. Biermann, A.W., Feldman, J.A.: On the synthesis of finite-state machines from samples of their behavior. IEEE Trans. Comput. **21**(6), 592–597 (1972)
7. Bugalho, M.M.F., Oliveira, A.L.: Inference of regular languages using state merging algorithms with search. Pattern Recognit. **38**(9), 1457–1467 (2005)
8. Coste, F., Nicolas, J.: Regular inference as a graph coloring problem. In: IWGI (1997)
9. Coste, F., Nicolas, J.: How considering incompatible state mergings may reduce the DFA induction search tree. In: ICGI, pp. 199–210 (1998)
10. Eén, N., Sörensson, N.: Translating Pseudo-Boolean constraints into SAT. JSAT **2**(1–4), 1–26 (2006)
11. Gent, I.P., Nightingale, P.: A new encoding of all different into SAT. In: Workshop on Modelling and Reformulating Constraint Satisfaction Problems, pp. 95–110 (2004)
12. Grinchtein, O., Leucker, M., Piterman, N.: Inferring network invariants automatically. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 483–497. Springer, Heidelberg (2006). https://doi.org/10.1007/11814771_40
13. Heule, M.J.H., Verwer, S.: Exact DFA identification using SAT solvers. In: Sempere, J.M., García, P. (eds.) ICGI 2010. LNCS (LNAI), vol. 6339, pp. 66–79. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15488-1_7
14. Heule, M., Verwer, S.: Software model synthesis using satisfiability solvers. Empir. Softw. Eng. **18**(4), 825–856 (2013)
15. de la Higuera, C.: A bibliographical study of grammatical inference. Pattern Recognit. **38**(9), 1332–1348 (2005)
16. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation - International Edition, 2nd edn. Addison-Wesley, Boston (2003)
17. Lang, K.J.: Faster algorithms for finding minimal consistent DFAs. Technical report, NEC Research Institute (1999)
18. Lang, K.J., Pearlmutter, B.A., Price, R.A.: Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In: Honavar, V., Slutzki, G. (eds.) ICGI 1998. LNCS, vol. 1433, pp. 1–12. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054059
19. Morgado, A., Heras, F., Liffiton, M.H., Planes, J., Marques-Silva, J.: Iterative and core-guided MaxSAT solving: a survey and assessment. Constraints **18**(4), 478–534 (2013)
20. Neider, D.: Applications of automata learning in verification and synthesis. Ph.D. thesis, RWTH Aachen University (2014)
21. Neider, D., Jansen, N.: Regular model checking using solver technologies and automata learning. In: NFM, pp. 16–31 (2013)
22. Oliveira, A.L., Marques-Silva, J.: Efficient algorithms for the inference of minimum size DFAs. Mach. Learn. **44**(1/2), 93–119 (2001)
23. Sinz, C.: Towards an optimal CNF encoding of Boolean cardinality constraints. In: van Beek, P. (ed.) CP 2005. LNCS, vol. 3709, pp. 827–831. Springer, Heidelberg (2005). https://doi.org/10.1007/11564751_73
24. Trakhtenbrot, B.A., Barzdin, Y.M.: Finite Automata: Behavior and Synthesis. North-Holland Publishing Company, Amsterdam (1973)
25. Ulyantsev, V., Tsarev, F.: Extended finite-state machine induction using SAT-solver. In: ICMLA, pp. 346–349 (2011)

26. Ulyantsev, V., Zakirzyanov, I., Shalyto, A.: BFS-based symmetry breaking predicates for DFA identification. In: Dediu, A.-H., Formenti, E., Martín-Vide, C., Truthe, B. (eds.) LATA 2015. LNCS, vol. 8977, pp. 611–622. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15579-1_48
27. Verwer, S., Hammerschmidt, C.A.: flexfringe: a passive automaton learning package. In: ICSME, pp. 638–642 (2017)
28. Walkinshaw, N., Lambeau, B., Damas, C., Bogdanov, K., Dupont, P.: STAMINA: a competition to encourage the development and assessment of software model inference techniques. Empir. Softw. Eng. **18**(4), 791–824 (2013)
29. Wieman, R., Aniche, M.F., Lobbezoo, W., Verwer, S., van Deursen, A.: An experience report on applying passive learning in a large-scale payment company. In: ICSME, pp. 564–573 (2017)
30. Zakirzyanov, I., Shalyto, A., Ulyantsev, V.: Finding all minimum-size DFA consistent with given examples: SAT-based approach. In: Cerone, A., Roveri, M. (eds.) SEFM 2017. LNCS, vol. 10729, pp. 117–131. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74781-1_9