Reducing the size of NFAs by using equivalences and preorders

Lucian Ilie[⋆,⋆⋆], Roberto Solis-Oba[⋆⋆⋆], and Sheng Yu[†]

Department of Computer Science, University of Western Ontario
London, Ontario, N6A 5B7, CANADA
ilie|solis|syu@csd.uwo.ca

Abstract. The efficiency of regular expression matching algorithms depends very much on the size of the nondeterministic finite automata (NFA) obtained from regular expressions. Reducing the size of these automata by using equivalences has been shown to reduce significantly the search time. We consider the problem of reducing the size of arbitrary NFAs using equivalences and preorders. For equivalences, we give an algorithm to optimally combine equivalent states for reducing the size of the automata. We also show that the problem of optimally using preorders to reduce the size of an automaton is NP-hard.

Keywords: regular expression matching, finite automata, state complexity, equivalences, preorders

1 Introduction

Regular expressions lie at the heart of many applications, such as linguistics, computational biology, pattern recognition, text retrieval, and so on. A powerful and elegant theory provides tools to easily and efficiently solve many complex problems by mapping them to regular expressions, then obtaining nondeterministic finite automata (NFA) that recognize them, and finally constructing deterministic finite automata (DFA). However, a severe obstacle in any real implementation of the above scheme is the size of the DFA, which can be exponential in the length of the original regular expression. A simple algorithm for minimizing DFAs exists [7], but it has the drawback of requiring first the construction of the DFA to later minimize it. This might not be practical because of memory requirements and construction cost of the DFA.

A more promising (and more challenging) alternative is directly reducing the NFA before converting it into a DFA. This has the advantage of working over a much smaller structure (of size polynomial in the length of the regular expression) and of building the smaller DFA without the need to go through a larger one first. However, the NFA state minimization problem is hard (PSPACE-complete,

⋆ corresponding author; e-mail: ilie@csd.uwo.ca
⋆⋆ Research partially supported by NSERC.
⋆⋆⋆ Research partially supported by NSERC grant 227829-04.
† Research partially supported by NSERC.

[14]) and, therefore, algorithms such as [15–17] cannot be used in practice. There are also algorithms which build small NFAs from regular expressions, see [10, 6]. These algorithms consider the total size of NFAs, that is, they count both states and transitions, and they increase artificially the number of states to reduce the number of transitions. As the implementation crucially depends on the number of states, such algorithms may not help.

The idea of reducing the size of NFAs by merging states was first introduced by Ilie and Yu [12] who used left and right equivalence relations. Later, Champarnaud and Coulon [2] modified the idea to work for preorders. An algorithm to compute the equivalences in $O(m \log n)$ time on an NFA with n states and m transitions and an $O(mn)$ algorithm for computing preorders are presented in [11].

The above mentioned algorithms identify sets of states that could be merged without modifying the language accepted by the automaton. The number of states of the resulting NFA depends on the order in which the states are merged. Randomly choosing the order in which these mergings take place, as used so far, does not guarantee that the smallest NFAs that can be built with these techniques are produced.

In this paper we investigate optimal ways to use the information in equivalences and preorders to reduce NFAs. We first give an efficient algorithm for optimally combining the left and right equivalences for achieving the maximum reduction in the size of an NFA. We show that the same problem for preorders, however, is NP-hard. Since, potentially, preorders could produce a better reduction, a number of open problems remain, such as looking for alternative ways, e.g., approximation algorithms, to reduce NFAs using preorders.

Notice that we do not claim that the above techniques achieve optimal reduction in the size of nondeterministic finite automata, as this problem is PSPACE-complete. Rather, we use the adjective "optimal" to refer to the maximum reduction in size that these techniques can achieve. We will explain this more precisely in the following sections.

We expect our results to have applications to regular expression matching. A single equivalence was shown by Ilie et al. [11] to reduce significantly the search time, more so than the special properties of the Glushkov automaton (see [18, 19]). It remains to be tested how much we can further speed up regular expression search using the present reduction algorithms. Several important research directions are mentioned in the conclusions section.

2 Basic notions

We recall here the basic definitions we need throughout the paper. For further details we refer to [9] or [21].

Let A be an alphabet of constant size and A^* be the set of all words over A; ε denotes the empty word. A *language* over A is a subset of A^*. A *nondeterministic finite automaton* (*NFA*) is a tuple $M = (Q, A, \delta, I, F)$, where Q is the set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and

$\delta : Q \times A \to 2^Q$ is the transition mapping; δ is extended to $\delta : 2^Q \times A^* \to 2^Q$ by $\delta(S, a) = \bigcup_{q \in S} \delta(q, a)$, $\delta(S, \varepsilon) = S$, and $\delta(S, aw) = \delta(\delta(S, a), w)$, for $S \subseteq Q$, $w \in A^*$. The *language* recognized by M is

$$\mathcal{L}(M) = \{w \in A^* \mid \delta(I, w) \cap F \neq \emptyset\}.$$

For $p, q \in Q$, we denote

$$\begin{aligned}
&\mathcal{L}_L(M, p) = \{w \in A^* \mid p \in \delta(I, w)\},\\
&\mathcal{L}_R(M, p) = \{w \in A^* \mid \delta(p, w) \cap F \neq \emptyset\},\\
&\mathcal{L}(M, p, q) = \{w \in A^* \mid q \in \delta(p, w)\};
\end{aligned}$$

when M is understood, we simply write $\mathcal{L}_L(p)$, $\mathcal{L}_R(p)$, and $\mathcal{L}(p, q)$, respectively. The *reversed* automaton of M is $M^r = (Q, A, \delta^r, F, I)$, where $q \in \delta^r(p, a)$ iff $p \in \delta(q, a)$.

3 NFA reduction with equivalences

The idea of reducing the size of NFAs by merging states was investigated first by Ilie and Yu [12]; see also [13]. We describe it briefly in this section.

Let $M = (Q, A, \delta, I, F)$ be an NFA. We define $\equiv_R$ as the coarsest equivalence relation over Q that satisfies:

$(\mathcal{P}_1)$ $\equiv_R \cap (F \times (Q - F)) = \emptyset$,
$(\mathcal{P}_2)$ $\forall p, q \in Q, \forall a \in A,\ \big(p \equiv_R q \Rightarrow \forall q' \in \delta(q, a), \exists p' \in \delta(p, a), q' \equiv_R p'\big)$.

The equivalence $\equiv_R$ is the largest equivalence over Q which is right-invariant with respect to M; see [12, 13]. Given $\equiv_R$, the algorithm to reduce the automaton M is simple: while there are non-trivial equivalence classes, merge all states in a non-trivial equivalence class and modify the transitions accordingly.

Symmetrically, the relation $\equiv_L$ can be defined using the reversed automaton. An automaton M can be reduced according to either equivalence.

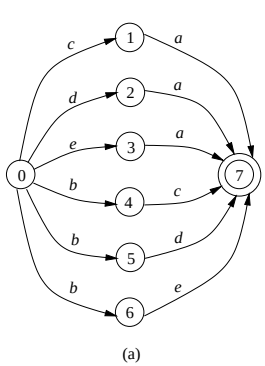
Example 1. Consider, for example, the automaton in Figure 1(a) where the equivalence classes are also shown. States 1, 2, and 3 belong to the same equivalence class of $\equiv_R$, and so they would be merged into a single state as shown in Fig 1(b). As Figure 1(c) shows, there are NFAs that can be reduced more by simultaneously using both equivalences.

Furthermore, there might not be a unique way to use optimally both $\equiv_R$ and $\equiv_L$ as the following example (from [13]) shows.

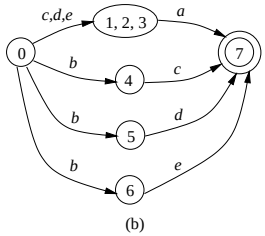
Example 2. In Figure 2, the automaton from the left has only two pairs of equivalent states: $1 \equiv_R 3$ and $1 \equiv_L 2$. We may either merge 1 with 3 or 1 with 2 but not both, since merging all three states into one introduces the word bd which is not in the language. Therefore, we have two different optimal ways of reducing the automaton.

Classes of $\equiv_R$: $\{0\}$
$\{1,2,3\}$
$\{4\}$, $\{5\}$
$\{6\}$, $\{7\}$

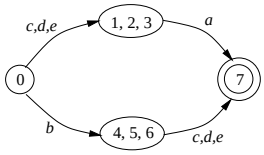
Classes of $\equiv_L$: $\{0\}$, $\{1\}$
$\{2\}$, $\{3\}$
$\{4,5,6\}$
$\{7\}$



(a)



(b)



(c)

Fig. 1. (a) An NFA. (b) Its reduced version using $\equiv_R$. (c) Its reduced version using $\equiv_R$ and $\equiv_L$.

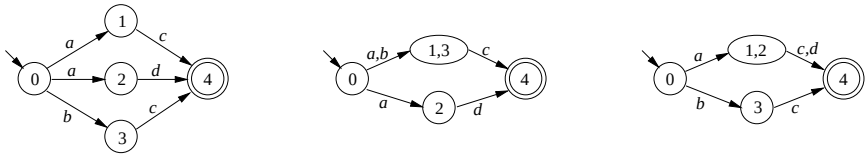


Fig. 2. An NFA, and its reduced versions using $\equiv_R$ and $\equiv_L$.

In [13] it is posed as an open problem to find a best way to do the reduction using both equivalences. In Section 4 we give an efficient algorithm for solving this problem.

A classical algorithm of Paige and Tarjan [20] was used in [11] to compute the equivalences fast. Given an automaton with n states and m transitions, the algorithm of [11] runs in time $\mathcal{O}(m \log n)$ and space $\mathcal{O}(m+n)$.

4 Efficient use of equivalences

We describe now an efficient algorithm for reducing the size of an NFA $M = (Q, A, \delta, I, F)$ by merging states according to the equivalence classes $\equiv_R$ and $\equiv_L$. In a certain sense, to be made precise below, we use the information in the two equivalences optimally.

Each of the two equivalences $\equiv_R$ and $\equiv_L$ defines a partition of the set Q of states. Let these partitions be

$$\Pi_R = \{X_1, X_2, \ldots, X_r\}$$
$$\Pi_L = \{X_{r+1}, X_{r+2}, \ldots, X_{r+s}\},$$

respectively. Two states p, q belong to the same set X_i, $i \leq r$ ($i > r$), if and only if $p \equiv_R q$ ($p \equiv_L q$).

A reduction merges states belonging to the same equivalence class into a single state. Let a reduction be $X^* = \{X_1^*, X_2^*, \ldots, X_\ell^*\}$, where each X_i^* represents a set of equivalent states that is merged into a single state in the reduced automaton. The reduced NFA has, then, ℓ states. Observe that each set X_i^* is a subset of at least one of the sets X_j. Let $X_i^* \subseteq X_{\pi(i)}$, where $1 \leq \pi(i) \leq r + s$. Clearly, $\ell \leq \min\{r, s\}$.

Note that $\cup_{i=1}^{\ell} X_i^* = Q$, where Q is the set of states of the NFA. Then, $\cup_{i=1}^{\ell} X_{\pi(i)} = Q$, and so $\{X_{\pi(1)}, X_{\pi(2)}, \ldots, X_{\pi(\ell)}\}$ is a *set cover* for Q from the family of sets $X = \Pi_R \cup \Pi_L$.

We can identify optimal use of the equivalences by finding an optimal solution for the instance $\langle Q, X = \Pi_R \cup \Pi_L \rangle$ of the *set covering problem*. In the set covering problem, given a finite set Q and a family X of subsets of Q, the goal is to find the smallest subset of X that includes all elements in Q. Any subset of X that includes all elements in Q is called a *set cover* for Q.

Let S^* be a smallest set cover for $\langle Q, X \rangle$. To achieve the maximum possible reduction in the size of the NFA that can be obtained with the equivalences, we first remove duplicated occurrences of the same state from the sets in S^*; i.e., if state p belongs to two subsets $S_i^*, S_j^* \in S^*$, then we remove p from either S_i^* or S_j^* so that p appears in only one subset of S^*. Then, all the states in the same subset $S_i^* \in S^*$ are merged into a single state in the reduced NFA. The reduced NFA will be called the EQ-*reduced NFA*.

The set covering problem is NP-hard, so the above algorithm for reducing NFAs might not be practical. Fortunately, the instance $\langle Q, X \rangle$ of the set covering problem defined by $\equiv_R$ and $\equiv_L$ is not an arbitrary one, but it has a very special structure. In particular, every state $p \in Q$ belongs to at most two of the subsets of X. Therefore, the algorithm of Bar-Yehuda end Even [1] gives a 2-approximate solution for this problem.

We can do better than this, though. Let us model the set covering problem as a bipartite graph $G_B = (L \cup R, E)$. Each set X_i is a vertex in this graph. Put vertices $X_1, \ldots, X_r$ in L and the rest in R. Every edge $p \in E$ corresponds to a state p of the NFA and it joins the two sets containing it (see Figure 3).

An optimal set cover for $\langle Q, X \rangle$ corresponds to a minimum *vertex cover* for G_B (a vertex cover for G_B is a subset of vertices incident on all edges). Since the graph G_B is bipartite, a minimum vertex cover can be easily derived from a maximum matching, which can be computed in $O(m'\sqrt{n}) = O(n^{3/2})$ time using the algorithm of Hopcroft and Karp [8], where m' is the number of edges in the bipartite graph and n is the number of vertices. The number of edges is equal to the number n of states in the NFA, and the number of vertices is the number of equivalence classes in $\equiv_R$ and $\equiv_L$, which is at most n.

The problem of computing a minimum vertex cover for a bipartite graph has been widely studied in the literature. For completeness, we present a simple algorithm for finding a minimum vertex cover for G_B from a maximum matching M^* of G_B. First, build the *residual network* G_{M^*} corresponding to the matching:

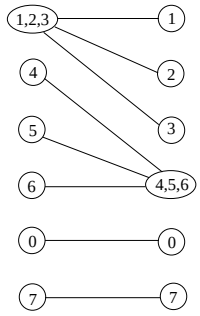


Fig. 3. Bipartite graph G_B for the NFA of Figure 1(a).

- Add a source node n_s to G_B and connect it to every vertex in L.
- Add a sink node n_t to G_B and connect it to every vertex in R.
- Direct every edge in the matching from R to L. Direct every edge (n_s, u), where u is incident to an edge in the matching, from u to n_s. Direct every edge (v, n_t) incident on the matching from n_t to v.
- All other edges are directed from left to right.

Do a depth first search traversal on this graph starting at vertex n_s, marking every vertex that can be reached from n_s. A minimum vertex cover C^* is formed by all marked vertices in R plus all un-marked vertices in L. Every marked vertex $v \in R$ is incident on an edge of the matching, because otherwise the path n_s, u, v used to reach v would include an edge (u, v) not incident on M^*, contradicting that M^* is maximum. Also, every un-marked vertex $u \in L$ must be incident on M^* since the edge between s and u must be directed from u to s. Hence, $|C^*| \leq |M^*|$. This procedure requires linear time.

Lemma 1. *C^* is a minimum vertex cover for the bipartite graph G_B, and the size of C^* is equal to the size of M^*.*

Proof. The above algorithm simply finds a cut of capacity C^* for a flow network G that has the same topology as G_{M^*}, but in which all edges (of unit capacity) are directed from left to right. By the max-flow min-cut theorem, we know that the size of M^* is equal to the capacity of a minimum cut, and thus, that the size of M^* is equal to the size of C^*. To see that C^* is a vertex cover, note that a minimum cut of G intersects every path from n_s to n_t. Therefore, for every edge (u, v) of G, the path n_s, u, v, n_t is intersected by a minimum cut and, thus, in the residual network G_{M^*} either v (but not n_t) is reachable from n_s or u is not reachable from n_s. This implies that either u or v belong to C^*, so edge (u, v) is covered by C^*.

We have proved the following result.

Theorem 1. *Given an NFA with n states and m transitions, there is an algorithm that computes the corresponding* EQ*-reduced automaton in $O(n^{3/2} + m \log n)$ time.*

5 NFA reduction with preorders

Champarnaud and Coulon [2] noticed that a better reduction can be obtained if the axioms $(\mathcal{P}_1)$ and $(\mathcal{P}_2)$ above are used to construct a preorder relation instead of an equivalence. Let us denote the largest (w.r.t. inclusion) preorder which satisfies $(\mathcal{P}_1)$ and $(\mathcal{P}_2)$ by $\subseteq_R$. It is then immediate that $p \subseteq_R q$ implies $\mathcal{L}_R(p) \subseteq \mathcal{L}_R(q)$.

As in the case of equivalences, the relation $\subseteq_L$ is symmetrically defined using the reversed automaton. Then, $p \subseteq_L q$ implies $\mathcal{L}_L(p) \subseteq \mathcal{L}_L(q)$.

The reduction with preorders is more complicated than with equivalences. We can merge two states p and q as soon as any of the following conditions is met:

(i) $p \subseteq_R q$ and $q \subseteq_R p$,
(ii) $p \subseteq_L q$ and $q \subseteq_L p$,
(iii) $p \subseteq_R q$, $p \subseteq_L q$, and $\mathcal{L}(p, p) = \{\varepsilon\}$.

However, after merging two states, the preorders $\subseteq_R$ and $\subseteq_L$ must be updated such that their relation with the languages $\mathcal{L}_R$ and $\mathcal{L}_L$ (see above) is preserved. For instance, in the case (i), assuming the merged state of p and q is denoted q, the update amounts to removing from $\subseteq_L$ all pairs (q, s) for which $p \not\subseteq_L s$. Case (ii) is handled similarly and (iii) does not need any update.

The condition (iii) appears in [2, 3] without the requirement $\mathcal{L}(p, p) = \{\varepsilon\}$ and, as noticed by [4], it is incorrect. In fact [4] removes this condition because its proof is incorrect. We give below a counterexample showing that, indeed, condition (iii) without the requirement $\mathcal{L}(p, p) = \{\varepsilon\}$ does not work.

Example 3. Consider the automaton in Figure 4. We have $\mathcal{L}_L(1) = ab^* \subseteq ab^* \cup \{x\} = \mathcal{L}_L(2)$ and $\mathcal{L}_R(1) = b^*c \subseteq b^*c \cup \{y\} = \mathcal{L}_R(2)$, but we cannot merge states 1 and 2 as this would add xb^*y to the language.

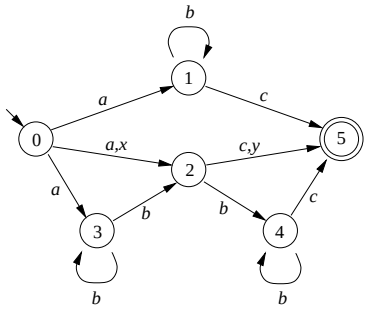


Fig. 4. The condition (iii) without $\mathcal{L}(p, p) = \{\varepsilon\}$; 1 and 2 cannot be merged.

Let us prove that our condition (iii) as stated above is correct. Denote the initial automaton by M and the one obtained after merging p and q by M'. Denote also the merged state in M' by q. We need to prove that $\mathcal{L}(M') \subseteq \mathcal{L}(M)$. The only problem might come from a word $w \in \mathcal{L}(M')$ such that $w_1 \in \mathcal{L}_L(M', q)$ for some prefix w_1 of w. This word w can be decomposed as $w = w_1w_2w_3$ such that: there is a path labelled w_1 in M' from an initial state to q which does not pass through q twice; there is a path labelled w_3 in M' from q to a final state which does not pass through q twice; and, there is a path labelled w_2 in M' which starts and ends in q. We have then

$$\begin{aligned} &w_1 \in \mathcal{L}_L(M,p) \cup \mathcal{L}_L(M,q) = \mathcal{L}_L(M,q),\\ &w_3 \in \mathcal{L}_R(M,p) \cup \mathcal{L}_R(M,q) = \mathcal{L}_R(M,q), \text{ and}\\ &w_2 \in (\mathcal{L}(M,p,p) \cup \mathcal{L}(M,p,q) \cup \mathcal{L}(M,q,p) \cup \mathcal{L}(M,q,q))^*. \end{aligned}$$

Since $\mathcal{L}(M,p,p) = \{\varepsilon\}$, at least one of $\mathcal{L}(M,q,p)$ and $\mathcal{L}(M,p,q)$ must be empty. Assume $\mathcal{L}(M,q,p)$ is empty. (The other case is similarly proved.) Then, using

$$\begin{aligned} &\mathcal{L}(M,q,q)\mathcal{L}_R(M,q) \subseteq \mathcal{L}_R(M,q) \text{ and}\\ &\mathcal{L}(M,p,q)\mathcal{L}_R(M,q) \subseteq \mathcal{L}_R(M,p) \subseteq \mathcal{L}_R(M,q), \end{aligned}$$

we obtain $w \in \mathcal{L}_L(M,q)\mathcal{L}_R(M,q) \subseteq \mathcal{L}(M)$, as claimed.

Since the preorder requirement is weaker than the equivalence requirement, $p \equiv_R q$ implies that $p \subseteq_R q$ and $q \subseteq_R p$. The converse is not true in general (see [2] for an example). Therefore, using preorders we have a chance to obtain a better reduction of the NFA. It remains to investigate how much better. Notice that the experiments in [3] are no longer valid since one of the conditions they used in the reduction was invalid. The preorders $\subseteq_R$ and $\subseteq_L$ can be computed in time $\mathcal{O}(mn)$ and space $\mathcal{O}(n^2)$; see [11] and [3].

6 Optimal use of preorders is hard

Contrary to the case of equivalences, it is hard to find the optimal way to use preorders to achieve best possible reductions. Again, what we mean by "optimal" is precisely defined below. All proofs in this section will be omitted due to lack of space.

As mentioned above, two states p, q of an automaton M can be merged into a unique state if any of the conditions (i)-(iii) is satisfied. Given the preorders $\subseteq_R$ and $\subseteq_L$, we let

$$\begin{aligned} p \cong_R q \quad &\text{iff} \quad p \subseteq_R q \text{ and } q \subseteq_R p, \text{ and}\\ p \cong_L q \quad &\text{iff} \quad p \subseteq_L q \text{ and } q \subseteq_L p. \end{aligned}$$

These new equivalences, $\cong_R$ and $\cong_L$, are coarser than $\equiv_R$ and $\equiv_L$, and they induce two partitions π_R and π_L of the set of states of M. Condition (iii) induces a partial order $\preceq$ on the set of states, where

$$p \preceq q \quad \text{iff} \quad p \subseteq_R q,\ p \subseteq_L q \text{ and } \mathcal{L}(p,p) = \{\varepsilon\}.$$

This partial order induces a family π_P of state subsets $P_1, P_2, \ldots, P_k$, $k \leq n$, where each P_i has a unique maximal element m_i, and two sets p, q belong to the same set P_i iff $p \preceq m_i$ and $q \preceq m_i$. Note that sets P_i cover all sates of M, but they are not necessarily a partition of Q.

We formulate the "optimal" use of preorders $\subseteq_R$ and $\subseteq_L$ for merging the states of M as an instance $\langle Q, \pi \rangle$ of the set covering problem: given a set Q of states and a family of state subsets $\pi = \pi_R \cup \pi_L \cup \pi_P$, the goal is to find the smallest subset S^* of π that includes all states of Q.

Consider an optimal solution S^* for the above set covering problem. Let $S^* = \{S_1^*, S_2^*, \ldots, S_\ell^*\}$; we reduce the automaton by merging all states in each set S_i^* into a unique state. Care must be taken that a state $p \in Q$ belonging to two different sets S_i^*, S_j^* is not merged twice. To ensure this, the sets S^* are considered in order. First, the set S_1^* is contracted to a single new state, and all states that belong to S_1^* are removed from the remaining sets $S_2^*, \ldots, S_\ell^*$. Then, S_2^* is contracted and all states in S_2^* are discarded from the remaining sets $S_3^*, \ldots, S_\ell^*$, and so on. The number of states in the final NFA M^* is ℓ; we call this automaton the PRE-*reduced NFA*.

The PRE-reduced NFA M^* has minimum size among those obtained by reducing M using preorders as explained above. This is because if there were another NFA, say M' with $\ell' < \ell$ states, that could be obtained from M by merging states as indicated by (i)-(iii), then the way in which the states are merged to produce M' defines a solution for the set covering problem $\langle Q, \pi \rangle$ of size $\ell' < \ell$, contradicting the optimality of S^*.

Observe that $\langle Q, \pi \rangle$ is a restricted instance of the set covering problem and, thus, the problem of optimally using preorders for reducing the number of states of a NFA might be simpler to solve than the general set covering problem. We show now that despite its restricted structure the set covering problem $\langle Q, \pi \rangle$ is still NP-hard. In fact, we show that an even more restricted version of the problem is NP-hard.

Let us consider only instances $\langle Q, \pi \rangle$, $\pi = \pi_R \cup \pi_L \cup \pi_P$, where the family π_P is a partition of Q. For these instances each state p appears in precisely 3 sets of π (one set belonging to each of π_R, π_L, and π_P). This particular class of instances of the set covering problem can be modeled as *3-partite hypergraphs* $H_M = (V_\pi, E_Q)$, where each vertex $v \in V_\pi$ corresponds to a subset in π and each hyperedge $e_p \in E_Q$ is incident on the vertices corresponding to the three sets $A_p \in \pi_R$, $B_p \in \pi_L$, and $C_p \in \pi_P$ containing p. A hypergraph is said to be 3-partite if its vertices can be partitioned into 3 disjoint sets such that every hyperedge is incident on exactly one vertex from each partition.

Note that every set cover of $\langle Q, \pi \rangle$ corresponds to a vertex cover (set of vertices incident on all hyperedges) of H_M. The vertex covering problem on 3-partite hypergraphs can be shown to be NP-hard via a reduction from the 3-SAT problem. We explain very briefly the idea.

Recall the 3-SAT problem. Given a boolean formula $f(x_1, \ldots, x_n) = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where each clause C_i contains exactly 3 literals (a literal is either a variable x_i or its negation $\bar{x}_i$), the problem is to decide whether there is an

assignment of values to the variables that satisfies all clauses. The 3-SAT problem is known to be NP-hard [5].

Given a boolean formula $f(x_1, \ldots, x_n)$ we build a 3-partite hypergraph $H_f = (V_f, E_f)$ as follows. Let n_i be the number of occurrences of x_i (either as x_i or as $\bar{x}_i$) in f. For each variable x_i we create $8n_i$ nodes: $x_{i1}, \ldots, x_{i\,n_i}$, $\bar{x}_{i1}, \ldots, \bar{x}_{i\,n_i}$, $d_{i1}, \ldots, d_{i\,n_i}$, $\bar{d}_{i1}, \ldots, \bar{d}_{i\,n_i}$, $s_{i1}, \ldots, s_{i\,4n_i}$. Nodes x_{ij}, $\bar{x}_{ij}$ represent the j-th occurrence of the variable x_i (in its j-th occurrence x_i can be either negated or not). Dummy nodes d_{ij}, $\bar{d}_{ij}$, and s_{ij} are used to ensure that the resulting hypergraph is 3-partite. These nodes are connected forming a cycle. Each hyperedge spans 3 nodes.

Furthermore, for each clause $C_k = \ell_{k1} \vee \ell_{k2} \vee \ell_{k3}$, we add a hyperedge incident on the 3 nodes corresponding to the literals ℓ_{ki} (note that different occurrences of the same variable are represented by different nodes). A very simple hypergraph, for the formula $f(x_1, x_2, x_3) = x_1 \vee \bar{x}_2 \vee x_3$, is shown in Fig. 5.

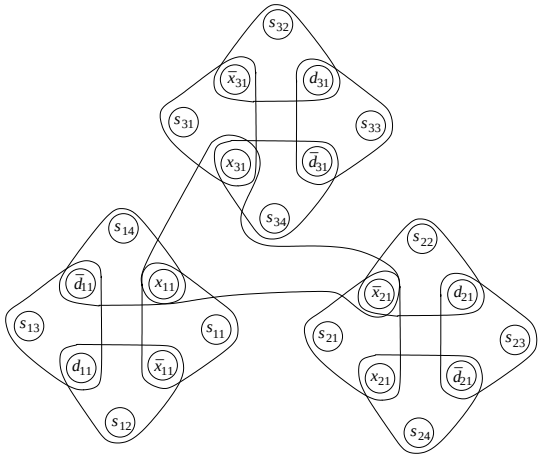


Fig. 5. Hypergraph corresponding to formula $f(x_1, x_2, x_3) = x_1 \vee \bar{x}_2 \vee x_3$.

The next step is to consider those 3-partite graphs as described above, which we call *3-SAT hypergraphs*, and show that the vertex cover problem on 3-SAT hypergraphs reduces to the problem of optimally using preorders for NFA reduction.

Consider a hypergraph H_f built from a boolean formula f. We construct an automaton M_f for which its preorders $\subseteq'_R$ and $\subseteq'_L$ define a hypergraph identical to H_f, but for 2 additional, isolated hyperedges. Denote the 3 partitions of the vertices of H_f by R_f, L_f, P_f. Let $\cong'_R$, $\cong'_L$, and $\preceq'$ be the equivalence relations and partial order, respectively, defined by the preorders $\subseteq'_R$ and $\subseteq'_L$.

The idea of the construction is as follows. The automaton is built so that each node of H_f corresponds to a set of states of M_f. Specifically, a vertex $u \in R_f$

defines a set of states that are equivalent under $\cong'_R$, a vertex $v \in L_f$ corresponds to an equivalence class under $\cong'_L$, and a vertex $w \in P_f$ corresponds to a member of the family of state subsets induced by the partial order $\preceq'$. Furthermore, each hyperedge of H_f incident on vertices u, v, and w, corresponds to a state of the automaton that belongs to classes u, v, and w of R_f, L_f, and P_f, respectively. The details of the construction are omitted due to lack of space.

The main result of this section is

Theorem 2. *Given an NFA, the problem of computing the corresponding* PRE*-reduced automaton is NP-hard.*

7 Conclusions and further research

We wish to point out that our algorithm for computing the EQ-reduced NFA only finds the best way of merging states with respect to the equivalence classes Π_R and Π_L. It is possible to reduce the size of an NFA by first merging some equivalent states in $\Pi_R \cup \Pi_L$ to get a new NFA and new equivalence classes Π'_R and Π'_L. Then, some equivalent states in $\Pi'_L \cup \Pi'_R$ could be merged to produce new equivalence classes, and so on. It is an open problem to find the best way of reducing an NFA by using this method.

We note that our algorithm for equivalences can be iterated for better results, but this does not solve the above problem. After reducing the size of an NFA using the equivalences, we can compute the new right and left equivalences $\tilde{\Pi}_R$ and $\tilde{\Pi}_L$ for the reduced automaton. Then, we apply the algorithm to $\tilde{\Pi}_R$ and $\tilde{\Pi}_L$. We can continue this process until no further reduction is achieved.

For preorders, we proved that the problem of optimally (as defined above) using them to attain the maximum possible reduction in the size of the automaton is NP-hard. Therefore, reductions with preorders, potentially more powerful than those with equivalences, might be too expensive to compute. This opens a new research topic: designing efficient approximation algorithms for using preorders in reducing NFAs, and testing their performance in practice.

More theoretical and experimental work is needed to determine which of the two relations, equivalences or preorders, is better in practice and which approach achieves greatest speedup for regular expression matching.

References

1. Bar-Yehuda, R., and Even, S., A linear-time approximation algorithm for the weighted vertex cover problem, *Journal of Algorithms* **2**, 1981, 198–203.
2. Champarnaud, J.-M., and Coulon, F., NFA reduction algorithms by means of regular inequalities, in: Z. Ésik, Z. Fülöp, eds., *Proc. of DLT 2003* (Szeged, 2003), Lecture Notes in Comput. Sci. **2710**, Springer-Verlag, Berlin, Heidelberg, 2003, 194 – 205.
3. Champarnaud, J.-M., and Coulon, F., NFA reduction algorithms by means of regular inequalities, *Theoret. Comput. Sci.* **327**(3) 241 – 253.

4. Champarnaud, J.-M., and Coulon, F., NFA reduction algorithms by means of regular inequalities – correction, *Theoret. Comput. Sci.*, to appear.
5. Garey M.R. and Johnson D.S., *Computers and intractability: a guide to the theory of NP-completeness*, W.H. Freeman and Company, New York, 1979.
6. Hagenah, C., and Muscholl, A., Computing ϵ-free NFA from regular expressions in $O(n\log^2(n))$ time, *Theor. Inform. Appl.* **34** (4) (2000) 257 – 277.
7. Hopcroft, J., An $n\log n$ algorithm for minimizing states in a finite automaton, *Proc. Internat. Sympos. Theory of machines and computations*, Technion, Haifa, 1971, Academic Press, New York, 1971, 189–196.
8. Hopcroft, J.E., and Karp, R., An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, *SIAM Journal on Computing*, bf 2(4) (1973), 225–231.
9. Hopcroft, J.E., and Ullman, J.D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Mass., 1979.
10. Hromkovic, J., Seibert, S., and Wilke, T., Translating regular expressions into small ϵ-free nondeterministic finite automata, *J. Comput. System Sci.* **62** (4) (2001) 565 – 588.
11. Ilie, L., Navarro, G., and Yu, S., On NFA reductions, in: J. Karhumaki, H. Maurer, G. Paun, G. Rozenberg, eds., *Theory is Forever* (Salomaa Festschrift), Lecture Notes in Comput. Sci. 3113, Springer-Verlag, Berlin, Heidelberg, 2004, 112 – 124.
12. Ilie, L., and Yu, S., Algorithms for computing small NFAs, in: K. Diks, W. Rytter, eds., *Proc. of the 27th MFCS*, (Warszawa, 2002), Lecture Notes in Comput. Sci., **2420**, Springer-Verlag, Berlin, Heidelberg, 2002, 328 – 340.
13. Ilie, L., and Yu, S., Reducing NFAs by invariant equivalences, *Theoret. Comput. Sci.* **306** (2003) 373 – 390.
14. Jiang, T., and Ravikumar, B., Minimal NFA problems are hard, *SIAM J. Comput.* **22**(6) (1993), 1117 – 1141.
15. Kameda, T., and Weiner, P., On the state minimization of nondeterministic finite automata, *IEEE Trans. Computers* **C-19**(7) (1970) 617 – 627.
16. Melnikov, B. F., A new algorithm of the state-minimization for the nondeterministic finite automata, *Korean J. Comput. Appl. Math.* **6**(2) (1999) 277 – 290.
17. Melnikov, B. F., Once more about the state-minimization of the nondeterministic finite automata, *Korean J. Comput. Appl. Math.* **7**(3) (2000) 655–662.
18. Navarro, G., and Raffinot, M., Compact DFA Representation for Fast Regular Expression Search, *Proc. WAE'01*, Lecture Notes Comput. Sci. **2141**, Springer-Verlag, Berlin, Heidelberg, 2001, 1 – 12.
19. Navarro, G., and Raffinot, M., *Flexible Pattern Matching in Strings. Practical On-Line Search Algorithms for Texts and Biological Sequences*, Cambridge University Press, Cambridge, 2002.
20. Paige, R., and Tarjan, R.E, Three Partition Refinement Algorithms, *SIAM J. Comput.* (1987) **16**(6) 973 – 989.
21. Yu, S., Regular Languages, in: G. Rozenberg, A. Salomaa, *Handbook of Formal Languages, Vol. I*, Springer-Verlag, Berlin, 1997, 41 – 110.