The Identity Checking Problem for Semigroups

Anna Gorbenko
Department of Intelligent Systems and Robotics
Ural Federal University
620083 Ekaterinburg, Russia
gorbenko.ann@gmail.com

Vladimir Popov
Department of Intelligent Systems and Robotics
Ural Federal University
620083 Ekaterinburg, Russia
Vladimir.Popov@usu.ru

Abstract
In this paper, we consider the identity checking problem for semigroups. We propose a genetic algorithm to solve the problem.

Keywords: identity checking problem, semigroup, genetic algorithm

There is a considerable interest in investigation of semigroup identities (see e.g. [1] – [4]). In particular, the identity checking problem for finite semigroups is extensively studied (see e.g. [5] and references in [5]). The identity checking problem in semigroup $A$ is the following combinatorial decision problem.

**CHECK-Id($A$):**

**Instance:** Words of variables $u$ and $v$.

**Question:** Whether or not the identity $u = v$ holds in $A$?

Usually, a semigroup for an instance of the identity checking problem for finite semigroups is given by a semigroup multiplication table. There is a finite semigroup $A$ such that **CHECK-Id($A$)** is $\text{co-NP}$-complete (see e.g. [5]).

Note that a large number of algorithmic problems of robotics received a lot of attention recently (see e.g. [6] – [15]). The representation of robotic systems plays an important role in solutions of robotic tasks (see e.g. [16]). There is a natural way to represent a robotic system by elements of some semigroup (see e.g. [17]). But in this case, we need to consider exponentially
large or infinite semigroups. Therefore, for robotic systems, a representation by semigroup relations is preferred. It should be noted that in some cases a representation by relations and identities may be considerably shorter than any representation by relations. So, for robotic systems, it is interesting to consider the problem $\text{CHECK-ID}(\mathcal{A})$ where $\mathcal{A}$ is a description of some robotic system. In this paper, we assume that a semigroup is given by a set of semigroup relations. In this case, also there is a finite semigroup $\mathcal{A}$ such that $\text{CHECK-ID}(\mathcal{A})$ is $\text{co-NP}$-complete. There is an infinite semigroup $\mathcal{A}$ such that $\text{CHECK-ID}(\mathcal{A})$ is undecidable (see e.g. [18]). In this paper, we consider a genetic algorithm to solve the problem.

Let $\Sigma$ be a finite system of semigroup relations. Let

$$\mathcal{A} = \langle a[1], \ldots, a[n] \mid \Sigma \rangle$$

be a semigroup where $\{a[1], \ldots, a[n]\}$ is a set of generators of $\mathcal{A}$. Let

$$u(x[1], \ldots, x[m]) = v(x[1], \ldots, x[m])$$

be an identity.

To solve $\text{CHECK-ID}(\mathcal{A})$, we can either derive the identity or to prove its falsity. Therefore, we use a parallel run of two sequences of genetic algorithms.

**Assumption 0:** $\mathcal{A} \not\models u(x[1], \ldots, x[m]) = v(x[1], \ldots, x[m])$.

- A genetic algorithm GA[0,1] for selection a set $S$ of elements of $\mathcal{A}$.
- A genetic algorithm GA[0,2] for construction of homomorphism $H : \mathcal{A} \rightarrow S = \langle S \mid \Sigma \rangle$

where $S$ is the subsemigroup of $\mathcal{A}$ generated by $S$.

- A genetic algorithm GA[0,3] for construction of multiplication table of $S$.
- A genetic algorithm GA[0,4] for selection a set $F$ of elements of $S$.

Consecutive run of genetic algorithms GA[0,1], GA[0,2], GA[0,3], GA[0,4] allows us to select values of $x[1], \ldots, x[m]$ that can potentially falsify the identity.

**Assumption 1:** $\mathcal{A} \models u(x[1], \ldots, x[m]) = v(x[1], \ldots, x[m])$.

- A genetic algorithm GA[1,1] for selection a set of templates $W = \{w \mid w \in (\{a[1], \ldots, a[n]\} \cup \{y[i] \mid i \in N\})^+\}$. 

A genetic algorithm GA[1,2] for discovery a set of equalities


where we consider

\[ \{ y[i] \mid i \in N \} \]

as the set of variables.

A genetic algorithm GA[1,3] for deduction a set of identities \( T \).

A genetic algorithm GA[1,4] for deduction

\[ u(x[1], \ldots, x[m]) = v(x[1], \ldots, x[m]) \].

At first, we run GA[1,1] and create a set \( W \). After this, we run GA[1,2]. GA[1,2] uses auxiliary genetic algorithm for initial prediction of elements of \( E \). For initial value of \( E \), we use a recursive parallel run of a genetic algorithm GA[1,4] for deduction of equalities and consecutive run of genetic algorithms GA[0,1], GA[0,2], GA[0,3], GA[0,4]. GA[1,4] allows us to prove some equalities from initial set \( E \). Consecutive run of genetic algorithms GA[0,1], GA[0,2], GA[0,3], GA[0,4] we use to falsify some elements of \( E \). GA[1,3] uses auxiliary genetic algorithm for initial prediction of elements of \( T \). For initial value of \( T \), we use a recursive run of GA[1,4] for deduction of identities. GA[1,4] uses four additional operators.

• Union of constants: if

\[ A \models w[1](x) = w[2](x), \]

for any value of a constant \( x \), then we can consider \( x \) as a variable.

• Separation of variables: if \( w \) is a some word, \( x \) is a variable, and \( x \in w \), then we can replace \( x \) by any element of \( A \).


• Substitution: for any variable \( x \) and for any \( w(x) \) and \( u \), we can consider \( w(u) \).

It is easy to see that we can use only GA[1,4] instead of usage of GA[1,1], GA[1,2], GA[1,3], and GA[1,4]. Selected experimental results are given in Table 1.

In Table 1, we assume equal allocation of computing resources for the consideration of assumptions 0 and 1. A[1] is an algorithm that uses only GA[1,4] after \( 10^5 \) generations. A[2] is an algorithm that uses GA[1,1], GA[1,2], GA[1,3],
and GA[1,4] after $10^3$ generations. A[3] is an algorithm that uses GA[1,1], GA[1,2], GA[1,3], and GA[1,4] after $10^4$ generations. A[4] is an algorithm that uses GA[1,1], GA[1,2], GA[1,3], and GA[1,4] after $10^5$ generations.

Since we use a parallel run of two sequences of genetic algorithms for the consideration of assumptions 0 and 1, we need some procedure to divide computing resources. In our computational experiments, we consider assumptions 0 and 1 on equal computing resources. Also, we consider a genetic algorithm for dynamic allocation of computing resources. Selected experimental results are given in Table 2.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>average time</td>
<td>94.17 h</td>
<td>72.89 h</td>
<td>61.75 h</td>
<td>42.3 h</td>
</tr>
</tbody>
</table>

Table 1: Experimental results for different genetic algorithms.

<table>
<thead>
<tr>
<th></th>
<th>ER</th>
<th>DA($10^3$)</th>
<th>DA($10^4$)</th>
<th>DA($10^5$)</th>
<th>DA($10^6$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>average time</td>
<td>42.3 h</td>
<td>39.57 h</td>
<td>33.71 h</td>
<td>11.84 h</td>
<td>11.36 h</td>
</tr>
</tbody>
</table>

Table 2: Experimental results for resource allocation where ER denotes equal computing resources and DA($g$) denotes dynamic allocation of computing resources with a genetic algorithm after $g$ generations.

ACKNOWLEDGEMENTS. The work was partially supported by Analytical Departmental Program “Developing the scientific potential of high school” 8.1616.2011.

References


The identity checking problem for semigroups


Received: December 3, 2012