Pattern Discovery in Colored Strings

Zsuzsanna Lipták\textsuperscript{1}, Simon J. Puglisi\textsuperscript{2}, and Massimiliano Rossi\textsuperscript{3}

\textsuperscript{1}University of Verona, Department of Computer Science.
\textsuperscript{2}University of Helsinki, Department of Computer Science.
\textsuperscript{3}University of Florida, Department of Computer & Information Science & Engineering.

SEA 2020
16 Jun 2020, Catania (online)
Motivations – Assertion mining

Embedded Systems are everywhere

The design of embedded systems requires to evaluate the correctness of its functionalities. Usually done using assertions (Logic formulae).

Typically written by hand by the designers. It might take months to find a small and effective set of assertions.

Automatic extraction of assertions from simulation traces.
Motivations – Assertion mining

Simulation trace

<table>
<thead>
<tr>
<th>T</th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
<th>$o_1$</th>
<th>$o_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Simulation trace

<table>
<thead>
<tr>
<th>T</th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
<th>$o_1$</th>
<th>$o_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Motivations – Assertion mining

<table>
<thead>
<tr>
<th>Simulation trace</th>
<th>Input alphabet</th>
<th>Output alphabet</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>( i_1 )</td>
<td>( i_2 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Colored string:

\( X \ Y \ X \ Z \ X \ Y \ Z \ Y \ X \ X \ Z \)

A C A C A C B A C A B

1 2 3 4 5 6 7 8 9 10 11
Colored Strings

**Definition**
Colored strings are strings where each character is assigned one of a finite set of colors.

**Objective**
We want to find patterns in the string that always occur with the same color at a certain distance.

We say that ACA is (Y,3)-unique.
Pattern Discovery

Given a colored string $S$ and a color $Y$, report all pairs $(T, d)$ such that $T$ is $(Y, d)$-unique substring of $S$.

Note

Although this problem is simpler than the assertion mining problem, the solution to our problem contains all the information, possibly filtered, to recover the desired set of minimal assertions in a second stage.
Discovery all \((y, d)\)-unique substrings

\[
\begin{align*}
 f &: X\ Y\ X\ Z\ X\ Y\ Z\ Y\ X\ X\ Z \\
 S &: A\ C\ A\ C\ A\ C\ B\ A\ C\ A\ B \\
 S &: 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11
\end{align*}
\]

We need to check all occurrences of a substring of \(S\). To keep the space contained, we use dedicated string data structures, i.e. **Suffix trees**.

Since the delay is measured from the end of the substring, it is convenient to think in terms of prefixes, i.e. **Suffixes of the reverse string**.

\[
\begin{align*}
 f^{rev} &: Z\ X\ X\ Y\ Z\ Y\ X\ Z\ X\ Y\ X \\
 S^{rev} &: B\ A\ C\ A\ B\ C\ A\ C\ A\ C\ A\ B \\
 S^{rev} &: 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12
\end{align*}
\]
Suffix tree

$S$: B A C A B C A C A C A $ $
1 2 3 4 5 6 7 8 9 10 11 12

$P$: A C

- $parent(u)$
- $child(u, \$)$
- Locus of AC
- Implicit suffix link
- Suffix link
- Leaf number
Discovery all \((y, d)\)-unique substrings

\[
f : X \ Y \ X \ Z \ X \ Y \ X \ X \ Z \ f^{rev} : Z \ X \ X \ Y \ Z \ Y \ X \ Z \ X \ Y \ X
\]
\[
S : A \ C \ A \ C \ A \ C \ B \ A \ C \ A \ B \ S^{rev} : B \ A \ C \ A \ B \ C \ A \ C \ A \ C \ A \ C \ A \$
\]

1. Build the **suffix tree** of \(S^{rev}\)
2. Color a leaf if:
   - Either \(ln \leq d\)
   - Or \(f(ln - d) = y\)
3. Color an internal node if:
   - All children are colored.
4. If a node \(u\) is colored, output all strings represented along the incoming edge of \(u\).

**Output:** \(\ldots, \text{CA, ACA, \ldots}\)

**Runs in** \(O(n^3)\) **time.**

\(d = 3\)
\(y = Y\)
Minimal \((y, d)\)-unique substrings

We say that CA is **minimal** \((y,3)\)-unique, because A is **not** \((y,3)\)-unique and C is **not** \((y,4)\)-unique.

**Problem**
Given a colored string \(S\) and a color \(Y\), report all pairs \((T,d)\) such that \(T\) is **minimal** \((Y,d)\)-unique substring of \(S\).
Discovery all minimal \((y, d)\)-unique substrings

\[
f : X Y X Z X \underline{Y} Z \underline{Y} X X Z \quad f^{\text{rev}} : Z X X \underline{Y} Z \underline{Y} X Z X Y X
\]

\[
S : A C A C A C B A C A B \quad S^{\text{rev}} : B A C A B C A C A C A \quad \$
\]

\[
1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11 \quad 12
\]

We say that CA is **minimal** \((Y, 3)\)-unique, because

1. A is **not** \((Y, 3)\)-unique and
2. C is **not** \((Y, 4)\)-unique.

\(\text{(left minimality)}\)

Parent of AC is not colored.

Sufffix link of AC is not colored for \(d = 4\).  \(\text{(right minimality)}\)

Process \(d\) from 11 downto 0

**Output:** ..., CA, ...

Runs in \(O(n^2)\) time.

\(d = 3\)  
\(y = Y\)
Skipping Algorithm
Skipping algorithm

Given a node \( u \) and an integer \( \ell \), \( h(u, \ell) \) is the largest delay \( d < \ell \) such that the corresponding string can be \((y, d)\)-unique.

\[
\ell = 7 \\
y = Y
\]

\( u \) can be \((Y, 3)\)-unique.

\[
\ell = 4 \\
y = Y
\]

\( u \) is \((Y, 3)\)-unique.
Skipping algorithm

• We discover the strings for \( d \) from \( n \) downto 0. (right minimality)
• For each node \( u \), we keep the value \( h(u, d + 1) \) updated.
• We find a node \( u \) such that:
  1. \( u \) has the largest value \( h(u, d + 1) \);
  2. \( u \) has priority on its children. (left minimality)
• We check if \( u \) is right minimal, and if so, we report it.
• We update:
  1. the value of all nodes \( v \) in the subtree rooted on \( u \) to \( h(v, d) \)
  2. the value of all ancestors \( v \) of \( u \) to \( h(v, d) \)

Maximum-oriented indexed priority queue

Runs in \( O(n^2 \log n) \) time.
Output restrictions
Output restrictions

We restrict the output to \((y, d)\)-unique substrings with at least two occurrences followed by \(y\).

\[
f : X \ Y \ X \ Z \ X \ \boxed{Y} \ Z \ \boxed{Y} \ X \ X \ Z
\]
\[
S: \ A \ C \ A \ C \ A \ C \ B \ A \ C \ A \ B
\]
\[
1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11
\]
\[
d = 3
\]

\[
f : X \ Y \ X \ Z \ X \ Y \ Z \ Y \ X \ \boxed{X} \ Z
\]
\[
S: \ A \ C \ A \ C \ A \ C \ B \ A \ C \ A \ B
\]
\[
1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11
\]
\[
d = 7
\]

Considering this consideration as part of the problem, we can modify the computation of \(h(u, d)\), when all children of \(u\) are leaves.
Experimental results
Experimental results

**Algorithms:**
- **Baseline:** Suffix tree based algorithm.
- **Skipping:** Skipping algorithm.
- **Real:** Skipping with output restrictions as part of the problem.

**Data:**
1. **Synthetic data:** Randomly generated data varied:
   - string length \( n = 100, 1000, 10000, 100000, \)
   - alphabet size = 2, 4, 8, 16, 32,
   - number of colors = 2, 4, 8, 16, 32.
2. **Real data:** Simulation on a set of established benchmarks in embedded systems verification.

<table>
<thead>
<tr>
<th>Design</th>
<th>Description</th>
<th>PIs</th>
<th>POs</th>
<th>( n )</th>
<th>( \sigma )</th>
<th>( \gamma )</th>
<th>( n_{\gamma} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>b03</td>
<td>Resource arbiter</td>
<td>6</td>
<td>4</td>
<td>100000</td>
<td>17</td>
<td>5</td>
<td>3210</td>
</tr>
<tr>
<td>b06</td>
<td>Interrupt handler</td>
<td>4</td>
<td>6</td>
<td>100000</td>
<td>5</td>
<td>4</td>
<td>44259</td>
</tr>
<tr>
<td>s386</td>
<td>Synthesized controller</td>
<td>9</td>
<td>7</td>
<td>100000</td>
<td>129</td>
<td>2</td>
<td>8290</td>
</tr>
<tr>
<td>camelia</td>
<td>Symmetric key block cypher</td>
<td>262</td>
<td>131</td>
<td>103615</td>
<td>70</td>
<td>224</td>
<td>2292</td>
</tr>
<tr>
<td>serial</td>
<td>Serial data transmitter</td>
<td>11</td>
<td>2</td>
<td>100000</td>
<td>118</td>
<td>2</td>
<td>16353</td>
</tr>
<tr>
<td>master</td>
<td>Wishbone bus master</td>
<td>134</td>
<td>135</td>
<td>100000</td>
<td>417</td>
<td>80</td>
<td>759</td>
</tr>
</tbody>
</table>
Synthetic data

Legend
- base - $\gamma = 2$
- base - $\gamma = 8$
- base - $\gamma = 32$
- skip - $\gamma = 2$
- skip - $\gamma = 8$
- skip - $\gamma = 32$
- real - $\gamma = 2$
- real - $\gamma = 8$
- real - $\gamma = 32$
Real data

Design

Speedup

Algorithms
- base
- skip
- real

b03 b06 s386 camellia master serial
Thank you for your attention!