String Solving with Word Equations and Transducers: Towards a Logic for Analysing Mutation XSS

Anthony W. Lin (Yale-NUS), Pablo Barcelo (Univ. of Chile)
String Solving: A View on the Landscape
What are String Solvers?

Solvers for Satisfiability Modulo Theory (SMT) over strings

**Domain**: the set of all words over \( \Sigma \)

**Operations**: concatenation, regex matching, length constraints, replace, replace-all, string transductions, ...

\[ s_2 = s_1.s_1 \land \text{len}(s_2) = \text{len}(s_7) \land \ldots \]

A different combination of operations gives rise to a different theory over strings!! (Just as for integer domain)

Many string solvers: CVC, HAMPI, Kaluza, Kudzu, Norn, Pex/Z3, PISA, S3, Saner, Stranger, StrSolve, SUSHI, Z3-str, ...
Why Develop String Solvers?

• Static analysis of security vulnerabilities in web applications against code injection and XSS
  
  • Caused by improper handling of untrusted strings

• Automatic test case generation for scripting languages

• Path query languages for graph databases
String Solving: Theory vs. Practice

• Faster heuristics each year

• Much less progress on theory

Which SMT over strings is decidable?
1. Word equations (Makanin’77)
   \[ s_2.a.s_3.s_4 = s_1.s_3.s_2.b \]
2. Existential theory strings with concat (Buchi&Senger’90)
   \[ s_2 = s_1.s_1 \land s_3.s_2 \neq s_1.s_7.s_8 \]
3. Word equations with regex matching (Schulz’90)
   \[ s_2.a.s_3.s_4 = s_1.s_3.s_2.b \land s_1 \in (ab)^* \]

Open Problem: Decidability of Word equations with length constraints
The need to add string transductions
Cross-Site Scripting (XSS)

// Obtain friend request from server
element.innerHTML = friendName;
...

Created a user profile with name:
"<script>
window.open('http://evilsite.com')
</script>"

send a friend request to Dilbert
Sanitising Input Data

• Escape certain characters

• *EVERY* occurrence of `<` should be changed to `&lt;`

• *EVERY* occurrence of `>` should be changed to `&gt;`

* A kind of “replace-all” operation
Adding Sanitisation

Google Closure

```javascript
var x = goog.string.htmlEscape(friendName);

element.innerHTML = x
```

```html
<script>…</script>

will be converted to

```html
&lt;script&gt;…&lt;/script&gt;
```

The script won’t be executed by Dilbert’s browser
A more tricky example

escapeString “backslash-escape” certain metacharacters

' is replaced by \"39; or \'
" is replaced by \"34; or \"

Q: Is this code vulnerable to XSS?
Analysis of the code

INPUT 1: name being **Tom & Jerry** gives HTML markup

```
<a onclick="viewPerson('Tom &amp; Jerry')">Tom &amp; Jerry</a>
```

INPUT 2: name being `');alert(1);//` gives HTML markup

```
<a onclick="viewPerson('');alert(1);//'">');alert(1);//</a>
```

innerHTML “mutates” this string to

```
<a onclick="viewPerson('');alert(1);//'">');alert(1);//</a>
```
Detecting XSS via a String Solver

**Step 1:** Identify “sink variables” (innerHTML, document.write)

```javascript
var x = goog.string.htmlEscape(name);
var y = goog.string.escapeString(x);
nameElem.innerHTML = '<a onclick="viewPerson(' + x + '");' + x + '</a>'
```

**Step 2:** Find “attack patterns” from known vulnerabilities (eg, OWASP)

```regex
e1 = /<a onclick="viewPerson\(' \[\^']\[\^'\] ' \)\); \[\^']\[\^'\]\)">.*<\/a>/
```

**Step 3:** Express the program logic in a string logic:

1. $x = R1(name)$
2. $y = R2(x)$
3. $z = w1 \cdot y \cdot w2 \cdot x \cdot w3$
4. $\text{nameElem.innerHTML} = R3(z)$
5. $\text{nameElem.innerHTML}$ matches $e1$

**Step 4:** Check for satisfiability
Which String Logic?

1. $x = R_1(name)$
2. $y = R_2(x)$
3. $z = w_1 \cdot y \cdot w_2 \cdot x \cdot w_3$
4. `nameElem.innerHTML = R_3(z)`
5. `nameElem.innerHTML matches e_1`

concatenation

R1, R2, R3 - replace-all kind of operations

String transductions!
Finite-state I/O Transducers

Just like finite-state automaton, but the transition label is a pair of words: $v/w$

Erases 1

Replaces some reserved characters by HTML entity names

Relation recognised by $A$ is

$\{ (v_1 \cdots v_n, w_1 \cdots w_n) : q_0 \xrightarrow{v_1/w_1} \cdots \xrightarrow{v_n/w_n} q_n \in F \}$
Modelling sanitisation functions and implicit browser transductions

Lots of works modelling these as FST or extensions thereof:

- Saxena et al., S&P’10
- D’Antoni&Veanes, VMCAI’13
- Hooimejer et al., USENIX Security’11
- Veanes et al., POPL’11
- …
Is theory of strings with concatenation and FST decidable?
Undecidability

**Proposition** (BFL’13): Checking if the constraint
\[ x = y.z \& x = R(z) \]
for a transduction R, is satisfiable is undecidable

**Proposition**: Undecidability still holds when only allowing “erasing” transducers (i.e. replace A with an empty string)
The Straight-Line Fragment (SSA Form)

Inductive Definition:

(Base) An empty set $\emptyset$ of conjuncts is in SL

(Inductive) If $S$ is in SL with variables $x_1, \ldots, x_m$

then $S \land x_{m+1} = P_{m+1}$ is in SL, where

$P_{m+1} = R(y)$ OR $P_{m+1} = y_1 \cdots y_n$

where the $y$'s are variables in $S$ or new variables

Regex matching: a boolean combination of
Decidability of SL

**Theorem:** SATISFIABILITY for the class SL is decidable in exponential space (double-exponential-time)

In fact, EXPSPACE-complete

**Theorem (Bounded Model Property):**
Every satisfiable constraint in SL has a solution of double-exponential size

Provides some completeness guarantee of several existing string solvers

Under a reasonable assumption, we get a single-exponential bound
Proof idea for decidability (without regex matching)

Step 1: Remove concatenation from the formula

\[ y = xx \land z = R(y) \]

where \( R \) has states \( q_0, \ldots, q_n \)

\[ y_1 = x \]
\[ y_2 = x \]

\[ \bigvee_{j=0}^{n} (y_1 = R^{q_0, q_j}(x) \land y_2 = R^{q_j, q_n}(x)) \]
Bound on the size of formula without concatenation

“Doubling” Trick

\[
\begin{align*}
    y_1 &= y_0y_0 \\
    y_2 &= y_1y_1 \\
    y_3 &= y_2y_2 \\
    L(y_3) &= \text{variables}
\end{align*}
\]

Resulting formula uses

\[2^3 + 2^2 + 2^1 + 3 = 17\]

Can use this trick to encode EXPSPACE Turing machines
Solving the final formula

\[ y_1 = x \]
\[ y_2 = x \]
\[ \bigvee_{j=0}^{n} (y_1 = R^{q_0, q_j}(x) \land y_2 = R^{q_j, q_n}(x)) \]

Acyclic (straight-line)

Satisfiability for this kind of formulas is decidable

Post/pre images of regular languages under FST are regular
Improving the upper bound

The doubling tricks are artificial

Limiting them into a bounded height is reasonable in practice

All the examples we’ve seen in practice are of height at most 4

**Theorem**: SATISFIABILITY for the restricted SL is decidable in polynomial space (exponential-time)

**Theorem** (*Bounded Model Property*): Every satisfiable constraint in restricted SL has a solution of exponential size
Extending the logic
Adding integer constraints

Constraints of the form

\[ a_1 t_1 + \cdots + a_n t_n \leq d \]

where

- \( a_i \) is a constant integer
- \( t_i \) is either:
  1) an integer variable,
  2) \(|x|\) for some string variable \( x \)
  3) \(|x|_a\) for some string variable \( x \)
Decidability

**Theorem:** SATISFIABILITY for the class SL with integer constraints is decidable in exponential space

In fact, EXPSPACE-complete

**Theorem (Bounded Model Property):** Every satisfiable constraint in SL with integer constraints has a solution of double-exponential size
Conclusion and Future Work

• Concatenation and string transductions are both important for XSS applications

• Straight-line fragment of string logic with concatenation and transductions (and even with integer constraints) is decidable

• **Future work 1**: an algorithm for computing a better estimate of the maximum size of solutions

• **Future work 2**: study the extension with symbolic transducers

• **Future work 3**: A more precise model of sanitisation functions and implicit browser transductions as transducers