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 Σ

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

$$s2 = s1.s1 \wedge \text{len}(s2) = \text{len}(s7) \wedge \ldots$$

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

<u>Many string solvers</u>: 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)

$$s2.a.s3.s4 = s1.s3.s2.b$$

2. Existential theory strings with concat (Buchi&Senger'90)

$$s2 = s1.s1 \wedge s3.s2 \neq s1.s7.s8$$

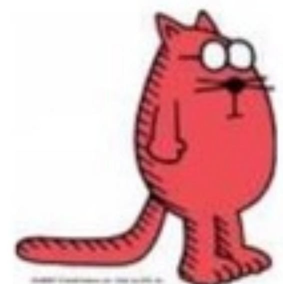
3. Word equations with regex matching (Schulz'90)

$$s2.a.s3.s4 = s1.s3.s2.b \wedge s1 \in (ab)^*$$

Open Problem: Decidability of Word equations with length constraints

The need to add string transductions

Cross-Site Scripting (XSS)



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

send a friend request to Dilbert

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

Sanitising Input Data

- Escape certain characters
- *EVERY* occurrence of < should be changed to <
- *EVERY* occurrence of > should be changed to >

A kind of “replace-all” operation

Adding Sanitisation

Google Closure

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

<script>…</script>

will be converted to

<script>…<script>

The script won’t be executed by Dilbert’s browser

A more tricky example

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

(Adapted from Kern’14)

escapeString “backslash-escape” certain metacharacters

‘ is replaced by ' or \’
“ is replaced by " or \”

Q: Is this code vulnerable to XSS?

Analysis of the code

SWAP

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

INPUT 1: name being Tom & Jerry gives HTML markup

Tom & Jerry

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

');alert(1);//'

innerHTML "mutates" this string to

');alert(1);//' **XSS!**

Detecting XSS via a String Solver

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

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

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

```
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 . y . w2 . x . w3
4. nameElem.innerHTML = R3(z)
5. nameElem.innerHTML matches e1

Step 4: Check for satisfiability

Which String Logic?

1. x = R1(name)
2. y = R2(x)
3. z = w1 . y . w2 . x . w3
4. nameElem.innerHTML = R3(z)
5. nameElem.innerHTML matches e1

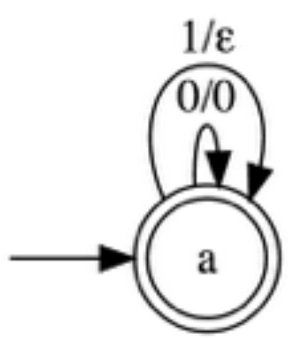
concatenation

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

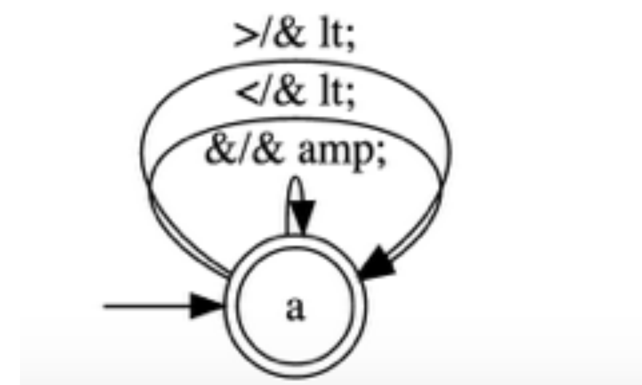
String transductions!

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 $\top$ of conjuncts is in SL

(Inductive) If S is in SL with variables

$$x_1, \ldots, x_m$$

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

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

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

$$y = xx \wedge z = R(y)$$

where R has states $q_0, \ldots, q_n$

$$y_1 = x$$
$$y_2 = x$$

$$\bigvee_{j=0}^{n} (y_1 = R^{q_0, q_j}(x) \wedge y_2 = R^{q_j, q_n}(x))$$

Bound on the size of formula without concatenation

"Doubling" Trick

$$y_1 = y_0 y_0$$

$$y_2 = y_1 y_1$$

$$y_3 = y_2 y_2$$

$$L(y_3)$$

Resulting formula uses

$$2^3 + 2^2 + 2^1 + 3 = 17$$

variables

Can use this trick to encode EXPSPACE Turing machines

Solving the final formula

$$y_1 = x$$
$$y_2 = x$$

$$\bigvee_{j=0}^{n} \left(y_1 = R^{q_0, q_j}(x) \wedge y_2 = R^{q_j, q_n}(x)\right)$$

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 (<u>Bounded Model Property</u>):
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