On Simple Linear String Equations, Sections 2 $\sim$ 5

Chung-Chih Li

LaTeX at 22:28, September 12, 2009

Abstract ....

1 Introduction

2 Motivating Examples

I just rewrite the 1st paragraph

Notation $x^r_{\rightarrow''}[0,16]$... not yet defined at this point.

We use Java Web application `BadLogin` as an example that motivates our work. The same example and pattern are used in our previous work [7] but the proposed informal algorithm in the same paper cannot generate all possible strings, while our new theory to be introduced in this paper can.

....

3 A String Equation System

In this section we define some necessary notations and terminologies that will be used throughout this paper. Let N denote the set of natural numbers and Σ a finite set of alphabets. Let R be the set of regular expressions over Σ. If $r \in R$, let $L(r)$ be the language represented by r. For simplicity, we abuse the notation by writing $\omega \in r$ when the context is clear that r is a regular expression.

If $\omega \in \Sigma^*$, we say that ω is a word (or equivalently, a string constant). We assume that there are infinitely many distinguishable variables for string constants, and let this set of variables be denoted by V. Intuitively, a string expression over Σ is a regular expression over Σ with some occurrences of variables in V and $[i.j]$ and $r \rightarrow \omega$ operators.

3.1 String Equations

We first define string expressions in the following definition.

Definition 1 *Let E denote the set of string expressions over Σ defined as follows:*

1. *If $x \in (V \cup R)$, then $x \in E$.*
2. *If $\mu, \nu \in E$, then $\mu\nu \in E$.*
3. *If $\mu \in E$, then $\mu[i,j] \in E$ for all $i, j \in N$ with $i \leq j$.*
4. *If $\mu \in R$, then $\mu_{r\to\omega}, \mu^{-}_{r\to\omega}, \mu^{+}_{r\to\omega} \in E$ for all $r \in R$ and $\omega \in \Sigma^*$.*
5. *Nothing else except those described above is in E.*

Note that we can extend E by using recursive definitions for 4 above, i.e., let the condition be $\mu \in E$. However, we restrict μ to R for simplicity reason; in particular, we want to keep our notations down when we come to define the semantic. By convention, we use $P_{(x_1/s_1.x_2/s_2,\ldots,x_n/s_n)}$ to denote the object obtained by replacing every occurrence of $x_1, x_2, \ldots x_n$ in P with $s_1, s_2, \ldots s_n$, respectively. However, when P is a string, how to identify the occurrences of x_1 is ambiguous. This motivates us to define a precise operation semantics for string substitution. We first define the mapping function, ϕ_ρ, as follows.

Definition 2 *Suppose $\rho = \{(x_1, s_1), (x_2, s_2), \ldots, (x_n, s_n)\} \subseteq (V \cup R) \times R$ such that, with $1 \leq i, j \leq n$, (i) ρ is single-valued, i.e., $i = j$ iff $x_i = x_j$; and (ii) ρ is consistent, i.e., if $x_i \in R$, then $L(s_i) \subseteq L(x_i)$. Then, we define the mapping function with respect to ρ by $\phi_\rho : E \to E$ such that, for every $\mu \in E$, $\phi_\rho(\mu) = \mu_{(x_1/s_1.x_2/s_2,\ldots,x_n/s_n)}$.*

In the definition above, being single-valued is a standard requirement to have the mapping function well-defined, and being consistent forces the replacement not to replace a regular expression by an arbitrary string but confine the substitute within $L(r)$. Consider the following example. Let $x \in V$, $a, b, c \in \Sigma$, and $\rho = \{(x, ab), ((abc)^*, abcabc)\}$. We have $\phi_\rho(x(abc)^*) = ababcabc$. Note that, if $\rho = \{(x, ab), (abc^*, abcabc)\}$, then, according to the restriction, ρ cannot define a mapping function because $abcabc \notin L(abc*)$.

Definition 3 *A string equation is denoted by $\mu \equiv \nu$ with $\mu, \nu \in E$. We say that $\rho \subseteq (V \cup R) \times R$ is a solution to the equation iff $\phi_\rho(\mu) = \phi_\rho(\nu)$.*

We can directly extend the definition above to a string equation system that is simply a finite set of string equations. We say that ρ is a solution to a system iff ρ is a solution to every string equation in the system.

Let $x, y \in V$, $a, b, c \in \Sigma$, and $\rho = \{(x, c), (ab^*, ab), (y, b)\}$. One can verify that ρ is a solution to string equation $xab^* \equiv cay$, since $\phi_\rho(xab^*) = \phi_\rho(cay) = cab$.

Definition 3 seems intuitively understandable. The problem is, however, do we have a clear semantic to say that $r_1 \equiv r_2$ iff $\phi_\rho(r_1) = \phi_\rho(r_2)$? For example, consider $a^* \equiv a^*a^*$ and $\rho = \{(a^*, b)\}$. Do we consider $\phi_\rho(a^*) = \phi_\rho(a^*a^*)$? Using the definitions above directly, we have $\phi_\rho(a^*) = b \neq bb = \phi_\rho(a^*a^*)$. This discrepancy shall be resolved by precise semantics of string substitutions.

3.2 Semantics

In this section we define the semantics for some notations we just introduced. For $s \in \Sigma^*$, let $s[i,j]$ denote the substrings of r starting from index i up to index $j-1$. Note that if $j-1$ is bigger than the last index of s, then $s[i,j]$ is the substring from index i to the end of s. If $r \in R$, then let $r[i,j] = \{s[i,j] : s \in r\}$. For $s, \omega \in \Sigma^*$ and $r \in R$, $S_{r\to\omega}$ denotes the set of all possible strings that can be obtained from s by substituting ω for every occurrence of substring $s \in L(r)$. Moreover, let $s^-_{r\to\omega}$ denote the string in $s_{r\to\omega}$ obtained with *pure reluctant left-most* pattern matching procedure, and $s^+_{r\to\omega}$ with *pure greedy left-most* pattern matching procedure. Formally, we define the three sets as follows:

Definition 4 *Let $s, \omega \in \Sigma^*$ and $r \in R$.*

$$s_{r\to\omega} = \begin{cases} \{s\} & \text{if } s \notin \Sigma^* r \Sigma^*; \\ \{\nu_{r\to\omega}\omega\mu_{r\to\omega} \mid s = \nu\beta\mu, \beta \in r\} & \text{otherwise.} \end{cases}$$

If $s_{r\to\omega} = \{s\}$, then let $s^-_{r\to\omega} = s^+_{r\to\omega} = s$; otherwise, define

- $s^-_{r\to\omega} = \nu\omega\mu^-_{r\to\omega}$ *where $s = \nu\beta\mu$ such that, $\nu \notin \Sigma^* r \Sigma^*$, $\beta \in r$, and for every x, y, z, t with $\nu = xy$ and $\beta = zt$, if $y \neq \epsilon$ then $yz \notin r$ and if $t \neq \epsilon$ then $z \notin r$.*
- $s^+_{r\to\omega} = \nu\omega\mu^+_{r\to\omega}$ *where $s = \nu\beta\mu$ such that, $\nu \notin \Sigma^* r \Sigma^*$, $\beta \in r$, and for every x, y, z, t, m, n with $\nu = xy$, $\beta = zt$, and $\mu = mn$, if $y \neq \epsilon$ then $yz \notin r$ and if $m \neq \epsilon$ then $y\beta m \notin r$.*

Note that $s_{r\to\omega}$ is a set of words but $s^-_{r\to\omega}$ and $s^+_{r\to\omega}$ are words. Moreover, $s_{r\to\omega}$ is declarative while $s^-_{r\to\omega}$ and $s^+_{r\to\omega}$ are procedural as they are defined based on a reluctant and a greedy procedures, respectively, with left-most pattern matching. One can verify that $s^-_{r\to\omega}$ and $s^+_{r\to\omega}$ are uniquely defined for any $s, \omega \in \Sigma^*$ and $r \in R$. Consider the following two examples. (i) If $s = aaab$, $r = (aa|ab)$, and $\omega = c$, then $s_{r\to\omega} = \{cc, acb\}$, and $s^-_{r\to\omega} = s^+_{r\to\omega} = cc$. (ii) If $s = aaa$, $r = a^+$, and $\omega = b$, then $s_{r\to\omega} = \{b, bb, bbb\}$, $s^-_{r\to\omega} = bbb$, and $s^+_{r\to\omega} = b$.

We can easily extend s to regular expressions as follows.

Definition 5 *Let $S, r \in R$ and $\omega \in \Sigma^*$. Define (i) $S_{r\to\omega} = \bigcup_{s\in S} s_{r\to\omega}$, (ii) $S^-_{r\to\omega} = \{s^-_{r\to\omega} : s \in S\}$, and (iii) $S^+_{r\to\omega} = \{s^+_{r\to\omega} : s \in S\}$.*

4 Modeling String Replacements

In this section we provide a model for solving string expressions.

Finite State Transducers (FTS)

We first introduce a modified finite state machine called *finite state transducer* used in [**?**] to recognize the regular relation for a phonological rule system, which we found also very useful for our string replacements.

Definition 6 *Let Σ^ϵ denote $\Sigma \cup \{\epsilon\}$. A finite state transducer (FST) is an enhanced two-taped nondeterministic finite state machine described by a quintuple $(\Sigma, Q, q_0, F, \delta)$, where Σ is the alphabet set, Q the set of states, $q_0 \in Q$ the initial state, $F \subseteq Q$ the set of final states, and δ is the transition function, which is a total function of type $Q \times \Sigma^\epsilon \times \Sigma^\epsilon \to 2^Q$.*

By convention, let the second and third arguments of the transition function come from the current symbols on first and second tapes, respectively. It is easy to argue that with an appropriate coding method, adding an additional input tape to the standard finite state machine does not enhance the power of the machine to accept more languages. What makes FST more powerful is to allow a transition based on only one symbol from either one of the two tapes. For example, $q_j \in \delta(q_i, a, \epsilon)$ means that if the current symbol in the first input tape is a, the machine can transfer from state q_i to q_j without reading (i.e., consuming) the current symbol in second tape. Let L_1 and L_2 be two languages. If an FST, M, accepts (ω_1, ω_2) iff $(\omega_1, \omega_2) \in L_1 \times L_2$, we say that M recognizes the language pair (L_1, L_2), and is denoted by $M_{L_1 \times L_2}$. It is straightforward to argue that, L_1 and L_2 are regular iff $M_{L_1 \times L_2}$ exists. For convenience, we can extend symbols to regular expressions for transitions to obtained augmented AFST, denoted by AFST.

Definition 7 *An augmented finite state transducer (AFST) is an FST $(\Sigma, Q, q_0, F, \delta)$ with the transition function augmented to type $Q \times R \times R \to 2^Q$, where R is the set of regular expressions over Σ.*

Note that, while we have tried to keep our setup as general as possible, we would restrict the transition function of an AFST to the type of $Q \times R \times \Sigma^* \to 2^Q$. In a transition diagram, we label the arch from q_i to q_j for transition $q_j \in \delta(q_i, r, \omega)$ by "$r : \omega$". Now, we can use an AFST to model our declarative string replacement $s_{r \to \omega}$ for any $s \in \Sigma^*$, $r \in R$, and $\omega \in \Sigma^*$. Figure 1 shows the construction, which presents an AFST that accepts $(s : \eta)$ iff $\eta \in s_{r \to \omega}$. In other words, given any two $s, \eta \in \Sigma^*$, we can use the AFST to check if η is a string obtained from s by replacing every occurrence of patterns in r with ω. We alternatively use FST and AFST for the time being without loss of generality since it is clear that every AFST has an corresponding FST to recognize the same language pair.

The AFST in Figure 1 uses nondeterminism to handle the declarative nature of $S_{r \to \omega}$. It is known that the nondeterministic transducer (NFST) is more powerful than the deterministic one (DFST), which differs with the well known fact that DFA

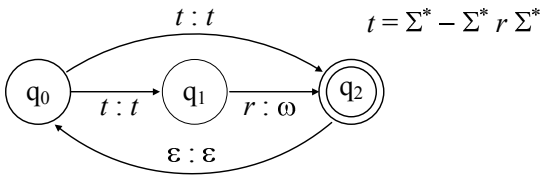

Figure 1: An AFST for $s_{r\to\omega}$

and NFA are equivalent. However, since we restrict the languages for the second tape to constant strings (words), we can manage our model to a deterministic one. Due to the space constrains, we omit the detailed construction. Clearly, a deterministic machine is much more convenient for implementation.

5 Solving Simple Linear String Equations

In this section we narrow down our scope to a kind of simplified liner string equations called *Simple Linear String Equations* (SLSE). Compared to the general string equations given in Definition 3, SLSE is easy to solve and yet it will suffice to formulate, for example, command injection security problems raised in many web applications. Our approach is to break SLSE into several basic cases and then combine their solutions to obtain the general solutions. We first define SLSE as follows.

Definition 8 *A Simple Linear String Equations (SLSE) $\mu \equiv r$ is a string equation such that $\mu \in E, r \in R$ provided that every string variable occurs at most once in μ.*

Recall Definitions 2 and 3 and consider the following definition.

Definition 9 *Let $\mu \equiv r$ be an SLSE. We say that ρ is a variable solution to $\mu \equiv r$ iff $\rho = \tau \cap (V \times R)$ and $\phi_\rho(\mu) = \phi_\tau(r)$ where τ is some solution to $\mu \equiv r$.*

Definition 9 rules out replacing a string/regular expression and restrict the replacement to variables only. This is a reasonable setup for some application such as injection attack since the hacker can only alter the values for variables but never be able to modify the code.

Definition 10 *Let $\mu \equiv r$ be an SLSE and suppose string variable v occurs in μ. The solution pool for v, denoted by $sp(v)$, is defined as follows.*

$$sp(v) = \{\omega \mid (v, \omega) \in \rho \text{ where } \rho \text{ is a variable solution to } \mu \equiv r.\}$$

It is clear that $sp(v)$ must be a regular language. In the following discussion, we will describe an algorithm that takes an SLSE as input and constructs as output regular expressions that represent the solution pools for all string variables in the equation.

5.1 Solving Basic Cases of SLSE

According to Definition 1, E is constructed recursively based on the atomic case (rule 1) and three operations: concatenation (rule 2), substring (rule 3), and string replacement (rule 4). Thus, solving an SLSE can be reduced to solving the four basic cases. The atomic case is trivial. That is, for $E \equiv r$, if $E \in R$ then the equation has no variable solution (a general solution may exist but we do not discuss the case in the present paper); if $E = x$ and $x \in V$, then the solution pool of x is simply $L(r)$.

Substring case: $\mu[i, j] \equiv r$, where $\mu \in E$ and $i, j \in N$. The following equivalence is straightforward by which we can remove a substring operator.

Equivalence 1 *For any SLSE of the form $\mu[i, j] \equiv r$ where $\mu \in E$, $r \in R$, and $i, j \in N$, ρ is a variable solution to $\mu[i, j] \equiv r$ iff ρ is a variable solution to $\mu \equiv \Sigma^i r[i, j]\Sigma^*$.*

In the following easy example, we will see how to put Definitions 2, 3, 9, and 10 into the picture. Consider SLSE $x[2, 4] \equiv ab^*$ where $x \in V$ and $a, b \in \Sigma$. Using Equivalence 1, we obtain $x = \Sigma^2 r[2, 4]\Sigma^*$ and hence $sp(x) = \Sigma^2 r[2, 4]\Sigma^*$. Consider an arbitrary word in $sp(x)$, e.g., $x = cc(ccabcc)[2, 4]ccc = ccabccc$. Let $\tau = \{(x, ccabccc), (ab^*, ab)\}$. According to Definitions 2 and 3, the mapping function, ϕ_τ, is well-defined and τ is a solution to $x[2, 4] \equiv ab^*$, since $\phi_\tau(x[2, 4]) = ccabccc[2, 4] = ab = \phi_\tau(ab^*)$. According to Definition 9, $\rho = \tau \cap (V \times R) = \{(x, ccabccc)\}$ is a variable solution to the equation. Finally, according to Definition 10, $ccabccc \in sp(x)$.

Concatenation case: $\mu\nu \equiv r$, where $\mu, \nu \in E$. The equivalence is obvious when $\nu \in R$. Consider $xr_1 \equiv r_2$ where $x \in V$ and $r_1, r_2 \in R$. In this trivial case, x is simply the (right) quotient of r_2 with respect to r_1, i.e., $sp(x) = \{\omega \mid \omega\eta \in L(r_2), \eta \in L(r_1)\}$. By convention, we denote the quotient of r_2 with respect to r_1 by r_2/r_1. We know that if r_1 and r_2 are regular, so is r_2/r_1. Thus, we have the following equivalence.

Equivalence 2 *For any SLSE of the form $\mu r_1 \equiv r_2$ where $\mu \in E$ and $r_1, r_2 \in R$, ρ is a variable solution to $\mu r_1 \equiv r_2$ iff ρ is a variable solution to $\mu \equiv r_2/r_1$.*

For the general concatenation case, we can reduce it to the case in Equivalence 2 by using other equivalences first.

Replacement case: $\mu_{r_1\to\omega} \equiv r_2$ where $\mu \in E$, $r_1, r_2 \in R$, and $\omega \in \Sigma^*$.

Consider the case $\mu = x$ with $x \in V$. According to the definition, a possible solution to $x_{r_1\to\omega} \equiv r_2$ is a word $s \in \Sigma^*$ such that $s_{r_1\to\omega} \subseteq L(r_2)$. Thus, $sp(x) = \{s | s_{r_1\to\omega} \subseteq L(r_2)\}$. Our goal is to construct an FST that accepts only $(s : \eta)$ such that $\eta \in L(r_2)$ and η is obtained from s by replacing every occurrence of patterns in r_1 with ω. In other words, we want an FST, denoted by $\mathcal{M}^{r_2}_{r_1\to\omega}$, such that,

$$(s : \eta) \in L(\mathcal{M}^{r_2}_{r_1\to\omega}) \iff \eta \in s_{r_1\to\omega} \cap L(r_2).$$

It is clear that, if such FST exists, we have the following lemma to reduce our SLSE.

Lemma 1 *Let $x \in V$. For any $x_{r_1\to\omega} \equiv r_2$ where $r_1, r_2 \in R$ and $\omega \in \Sigma^*$, we have $sp(x) = \{s | s_{r_1\to\omega} \subseteq L(r_2)\} = \{s \mid (s : \eta) \in L(\mathcal{M}^{r_2}_{r_1\to\omega})$ for some $\eta\}$.*

We fix some natation first. Given two FST's $\mathcal{M}_1$ and $\mathcal{M}_2$, let $\mathcal{M}_1||\mathcal{M}_2$ denote a FST such that,

$$L(\mathcal{M}_1||\mathcal{M}_2) = \{(s : \omega) \mid (s : \eta) \in \mathcal{M}_1 \text{ and } (\eta : \omega) \in \mathcal{M}_2 \text{ for some } \eta \in \Sigma^*\}$$

Intuitively, $\mathcal{M}_1||\mathcal{M}_2$ pipes the contents of the second tape of $\mathcal{M}_1$ into the first tape of $\mathcal{M}_2$, and simulates $\mathcal{M}_1$ and $\mathcal{M}_2$ in parallel. Thus, if we consider each FST representing a regular relation [**?**], $\mathcal{M}_1||\mathcal{M}_2$ represents a relation composition, i.e., $L_1(\mathcal{M}_1||\mathcal{M}_2)L_2$ iff there is L' such that $L_1\mathcal{M}_1L'$ and $L'\mathcal{M}_2L_2$. The construction of $\mathcal{M}_1||\mathcal{M}_2$ from given $\mathcal{M}_1$ and $\mathcal{M}_2$ can be found in [**?**, **?**], which is similar to the standard construction of a DFA that accepts the intersection of two regular languages. Due to the space constraints, we omit the details.

$\mathcal{M}^{r_2}_{r_1\to\omega}$ now can be constructed as follows. Let $\mathcal{M}^{r_2}$ denote the FST such that $(s : s) \in L(\mathcal{M}^{r_2})$ iff $s \in L(r_2)$. Also, let $\mathcal{M}_{r_1\to\omega}$ be the FST that models $s_{r_1\to\omega}$ as the one shown in Figure 1, i.e., $(s : \eta) \in L(\mathcal{M}_{r_1\to\omega})$ iff $\eta \in s_{r_1\to\omega}$. It is clear that $\mathcal{M}^{r_2}_{r_1\to\omega} = \mathcal{M}^{r_2}||\mathcal{M}_{r_1\to\omega}$. The equivalence below can be verified by Lemma 1 and a straightforward substitution.

Equivalence 3 *For any SLSE of the form $\mu_{r_1\to\omega} \equiv r_2$ where $\mu \in E$, $r_1, r_2 \in R$, and $\omega \in \Sigma^*$, ρ is a variable solution to $\mu_{r_1\to\omega} \equiv r_2$ iff ρ is a variable solution to $\mu \equiv r$ where r is $sp(x)$ to $x_{r_1\to\omega} \equiv r_2$.*