UNDECIDABILITY AND DECIDABILITY REGARDING SUBSTRING REPLACEMENT

by
Harvey M. Friedman*
Ohio State University
September 9, 2009

There is a vast subject which we call "existential theories of character strings", which is based on the basic operations used in the JAVA string libraries.

In this subject, we obtain undecidability and decidability results for the set of existential sentences with various primitives.

We believe that there many opportunities for obtaining practical decision procedures that can be used in conjunction with tools like JAVA Pathfinder. However, obtaining useful decidability results takes considerable research effort.

As indicated in the report, it seems promising to examine code in order to identify the most relevant combinations of basic operations from the JAVA string libraries - and restrictions on the path conditions (sets of literals).

1. UNDECIDABILITY.

We show that there is no algorithm for determining whether an existential sentence holds in the following interpreted language:

variables over character strings.

= between character strings.

(),0,1,2 for these four character strings.

x[y/z] for the result of successively replacing all occurrences of the character string y as a substring of the character string x, by the character string z, as in Replace in the JAVA library. If y = () then we take x[y/z] to be x.

We quote from
http://java.sun.com/javase/6/docs/api/java/lang/String.html

replace

```
public String replace(CharSequence target,
                      CharSequence replacement)
```

Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence. The replacement proceeds from the beginning of the string to the end, for example, replacing "aa" with "b" in the string "aaa" will result in "ba" rather than "ab".

Parameters:

`target` - The sequence of char values to be replaced

`replacement` - The replacement sequence of char values

Returns:

The resulting string

Throws:

NullPointerException - if `target` or `replacement` is `null`.

Since:

1.5

Thus = is a binary relation symbol, (),0,1,2 are constant symbols, and x[y/z] is a ternary function symbol.

Variables range over character strings, which are finite sequences of bytes. The undecidability proof works for any character set (finite or infinite) with at least three characters.

Obviously, requiring that any specific string that is to be mentioned in the language has to be among (),0,1,2, is rather restrictive. However, we are proving undecidability, and hence being restrictive strengthens the result. In the second section on decidability, we will allow any specific character string to be mentioned.

Below, we will also need to use the character strings 01,02,12.

We would like to define 01,02,12 by a term in this language. This apparently cannot be done. However, we can get close enough.

We define A(x) if and only if x[0/()] = 1 ∧ x[1/()] = 0.

LEMMA 1.1. A(x) if and only if x = 01 ∨ x = 10.

Proof: Since x[0/()] = 1, we see that if we remove 0's in x then we are left with 1. Hence x consists of 0's and exactly one 1. Since x[1/()] = 0, we see that x consists of 1's and exactly one 0. Hence x is 01 or 10. QED

Let B(x) ↔ x[0/()] = (). Let C(x) ↔ B(x[1/0]).

LEMMA 1.2. B(x) if and only if x consists entirely of 0's. C(x) if and only if x is a bit string.

Proof: Suppose B(x). If we remove all 0's from x, we are left with nothing. Hence x consists of all 0's. Suppose C(x). Then x[1/0] consists of all 0's. Hence x must consist entirely of 0's and 1's. QED

Let D(x,y) ↔ y[2/()] = x ∧ y[x/()] = 2.

LEMMA 1.3. Let x be a bit string. Then D(x,y) ↔ (y = 2x ∨ y = x2).

Proof: Let x be a big string, where D(x,y). If x = () then y = 2.

Assume x ≠ 2. By y[x/()] = 2, there is exactly one occurrence of 2 in y.

We claim that 2 occurs at the front or the end of y. To see this, let y = z2w, where z,w ≠ (). Then zw = x. Hence x is not a substring of y.

So write y = 2u or y = u2. Then u = x. QED

LEMMA 1.4. Let x,y,z be bit strings. Then z = xy if and only if y = 2z[2x/()] ∨ y = 2z[x2/()] ∨ y = z2[2x/()] ∨ y = z2[x2/()].

Proof: Let x,y,z be bit strings. Suppose y = 2z[2x/()]. Then 2x is a substring of 2z. Hence 2x is an initial substring of 2z. Therefore x is an initial substring of z. Hence y is the remaining part of z.

Suppose y = 2z[x2/()]. Then x2 is a substring of 2z. Hence x = z = (), y = ().

Suppose y = z2[2x/()]. Then 2x is a substring of z2. Hence

x = z = (), y = ().

Suppose y = z2[x2/()]. Then x2 is a substring of z2. Therefore x = z. Hence y = (). QED

Let E(x,y,z) ↔ (∃u,v)(D(x,u) ∧ D(z,v) ∧ y = v[u/()].

LEMMA 1.5. Let x,y,z be bit strings. Then z = xy ↔ E(x,y,z).

Proof: By Lemmas 1.3 and 1.4. QED

LEMMA 1.6. $(\exists n \geq 0)(x = 10^n)$ ↔ x is a bit string and x[01/()] = x. $(\exists n \geq 0)(x = 0^n1)$ ↔ x is a bit string and x[10/()] = x.

Proof: By x[01/()] = x, then 01 is not a substring of x. Since x is a bit string, x must be of the required form. By x[01/()] = x, then 01 is not a substring of x. Since x is a bit string, x must be of the required form. QED

Let J(x) ↔ (∃y)(x is a bit string ∧ A(y) ∧ x[y/()] = x).

LEMMA 1.7. $J(x) \leftrightarrow (\exists n \geq 0)(x = 10^n \vee x = 0^n1)$.

Proof: By Lemmas 1.1 and 1.7. QED

We now view each positive integer n as being represented by the strings 10^{n-1}. and $0^{n-1}1$. Thus the "positive integers" are the x with J(x).

Let K(x,y,z) ↔ J(x) ∧ J(y) ∧ J(z) ∧ E(x[1/0],y[1/0],z[1/0]).

LEMMA 1.8. K(x,y,z) if and only if there exists $n,m \geq 1$ such that $x = 10^{n-1} \vee x = 0^{n-1}1$, $y = 10^{n-1} \vee y = 0^{n-1}1$, $z = 10^{n+m-1} \vee z = 0^{n+m-1}1$.

Proof: Let J(x), J(y), J(z), E(x[1/0],y[1/0],z[1/0]). By Lemma 1.8, $x = 10^{n-1} \vee x = 0^{n-1}1$, $y = 10^{n-1} \vee y = 0^{n-1}1$. By Lemma 1.5, z[1/0] = xy[1/0]. Since x,y are bit strings, z and xy have the same length. QED

Thus K is an existential definition of addition for "positive integers".

We now give an existential definition of squaring of

"positive integers".

LEMMA 1.9. Let x be nonempty. Then xy = yx if and only if there exists n ≥ 0 such that $y = x^n$.

Proof: Fix nonempty x. We argue by induction on the length of y. Suppose xy = yx. Then y starts with x. Write y = xy'. Then xxy' = xy'x, and so xy' = y'x. By the induction hypothesis, let $y' = x^m$. Then $y = xx^m = x^{m+1}$. QED

LEMMA 1.10. Let $x = 10^{n-1}$, where n ≥ 1. Then x^n is the unique y such that xy = yx and y[0/()] = x[0/1].

Proof: Let $x = 10^{n-1}$, n ≥ 1. We can clearly set $y = x^n$, since the condition y[0/()] = x[0/1] asserts that the number of 1's in y is the same as the length of x (assuming x,y are bit strings).

Now let xy = yx and y[0/()] = x[0/1]. Then x,y are bit strings, and by Lemma 1.9, y is of the form $(10^{n-1})^m$, m ≥ 0. The number of 1's in y is obviously m. Hence m = n. QED

Let L(x,y) ↔ J(x) ∧ J(y) ∧ xy = yx ∧ y[0/()] = x[0/1].

LEMMA 1.11. L is an existential definition of the squaring relation SQ(x,y) ↔ J(x) ∧ J(y) ∧ len(y) = $\text{len}(x)^2$.

Proof: By Lemma 1.10. Use Lemma 1.4. QED

According to the solution to Hilbert's 10th problem, it is undecidable whether a polynomial equation with positive integer coefficients, in positive integer unknowns, has a solution.

This undecidability can be put into many well known forms. Here is one.

LEMMA 1.12. There is a fixed set S of equations of the form x+y = z and $x^2 = y$, such that the following holds. The set of all positive integers n such that S has a solution in positive integers, with x = n, is not recursive.

Proof: By the usual method of unraveling polynomials, with extra variables. This does require also equations of the form x•y = z. But we can use

x = y•z if and only if

$x = (y+z)^2 - y^2 - z^2$ if and only if
$x + y^2 + z^2 = (y+z)^2$

and introduce more variables. QED

LEMMA 1.13. There is a fixed existential formula $\varphi(x,...)$ in this language, such that the set of character strings x for which $\varphi(x,...)$ is true, is not recursive.

Proof: By transformation using Lemma 1.12. QED

THEOREM 1.14. There is a fixed set S of literals in this language, with distinguished variable x, such that the set of character strings x for which S is realizable, is not recursive.

Proof: Use Lemma 1.13. Write $\varphi(x,...)$ as a disjunction of existential formulas whose matrix is a conjunction of literals. If we have recursivity for each of this disjuncts, then we would have recursivity for $\varphi(x,...)$, contrary to Lemma 1.13. QED

This negative result is obviously not of direct practical importance. It does tell us that a potentially useful kind of decidability result is impossible.

What happens if we ask for realizability by character strings of limited length, say 2^{64}?

Of course, we then have decidability, with the obvious method of enumerating all character strings of length at most 2^64. Of course, this is entirely unfeasible. Perhaps it can be proved that there is no feasible algorithm for determining realizability by strings of length at most 2^{64}, where the size of the path conditions are reasonable.

2. CORRESPONDING DECIDABILITY.

We discuss the kind of decidability investigations that are suggested by the undecidability result in the previous section. We only establish a weak decidability result. However, we expect much stronger decidability results, which will require considerably more effort.

For these decidability questions, it is not reasonable to restrict the actual strings to be mentioned to merely the strings 0,1,2. We now allow any character strings to be

mentioned.

To avoid any ambiguity, let us carefully define the terms, atomic formulas, literals, and path conditions.

The terms are inductively defined as follows.

1. All variables are terms.
2. All character strings are terms.
3. If r,s,t are terms then r[s/t] is a term.

The atomic formulas are of the form s = t, where s,t are terms.

The literals are the atomic formulas and their negations.

The path conditions are the finite sets of literals.

The degree of a term, a literal, or a path condition, is the total number of left brackets that appear.

A path condition is called realizable if there exists an assignment of character strings to the variables such that the statements in the path condition are all true.

There is a well known decision procedure for determining whether a path condition of degree 0 is satisfiable. The procedure depends only on their being infinitely many character strings.

We now give an efficient decision procedure for determining whether a path condition of degree 1 is realizable.

LEMMA 2.1. It suffices to give an efficient decision procedure for determining whether a path condition of the following form is realizable:

$r[s/t] = p$
$x_1 \neq a_1$
...
$x_n \neq a_n$

where $n \geq 0$, r,s,t,p are variables or character strings, $x_1,...,x_n$ are variables among r,s,t,p, and $a_1,...,a_n$ are variables among r,s,t,p or character strings, and no x_i is a_i.

Proof: If the sole use of brackets is r[s/t] ≠ p, then we replace this by r[s/t] = v, v ≠ p, where v is a new variable. We can obviously remove literals without variables, and literals whose left and right sides are the same. If we have x = a (or a = x), where x is a variable, then we remove this equation and replace all occurrences of x by a. These operations converge in a path condition of the above form, except that some of the xi may not be among r,s,t,p. These inequations can be removed. Note that these operations do not change realizability. QED

Here is the general form of Lemma 2.1 that we will not use, but forms the basis for further, more difficult, decidability results.

LEMMA 2.2. Let n ≥ 1. In order to give an efficient decision procedure for determining whether a path condition of degree n is realizable, it suffices to give an efficient decision procedure for determining whether a path condition of the following form is realizable:

$r_1[s_1/t_1] = p_1$
...
$r_n[s_n/t_n] = p_n$
$x_1 \neq a_1$
...
$x_m \neq a_m$

where m ≥ 0, the r's,s's,t's,p's are variables or character strings, $x_1,...,x_m$ are variables among the r's,s's,t's,p's, and $a_1,...,a_m$ are variables among the r's,s's,t's,p's or character strings, and no x_i is a_i.

We define the normal path conditions to be the path conditions of the form given in Lemma 2.2.

According to Lemma 2.1, we will give a decision procedure for realizability of normal path conditions of degree 1.

It will be very convenient to work with what we call rich path conditions, C. We say that C is a rich path condition if and only if C is a normal path condition as in Lemma 2.2, such that for every distinct u,v among the r's,s's,t's,p's, where u is a variable, u ≠ v is present in C.

LEMMA 2.3. It suffices to give a decision procedure for

determining whether a rich path condition of degree 1 is realizable.

Proof: Let $r[s/t] = p$, $x_1 \neq a_1$, ..., $x_n \neq a_n$ be a normal path condition of degree 1. This is realizable if and only if it is realizable if we add some complete combination of $u = v$ or $u \neq v$, for u,v distinct elements of {r,s,t,p}, and u a variable. These equations can be removed so that each of these path conditions become rich path conditions. QED

We say that a rich path condition C is freely realizable if and only if for all finite sets E of character strings, we can realize C by character strings outside E.

The significance of freely realizable rich path conditions C is that every path condition C' obtained by adding inequations $v \neq \alpha$, where v is a variable and α is a character string, is realizable.

We now proceed according to the shape of the leading equation in rich path conditions of degree 1. We use Greek letters for particular character strings used in the leading equation.

$x[x/x] = x$. Freely realizable.
$x[x/x] = y$, x,y distinct. Not realizable.

$x]x/y] = x$, x,y distinct. Not realizable.
$x[x/y] = y$, x,y distinct. Freely realizable.
$x[x/y] = z$, x,y,z distinct. Not realizable.

$x[y/x] = x$, x,y distinct. Freely realizable. Set $x = a^n$, $y = a^{2n}$.
$x[y/x] = y$, x,y distinct. Not realizable.
$x[y/x] = z$, x,y,z distinct. Freely realizable. Set $x = a^{2n}$. $y = a^n$. $z = a^{4n}$.
$x[y/y] = x$, x,y distinct. Freely realizable.
$x[y/y] = y$, x,y distinct. Not realizable.
$x[y/y] = z$, x,y,z distinct. Not realizable.

$x[y/z] = x$, x,y,z distinct. Freely realizable. Set $x = a^n$, $y = a^{2n}$, $z = a^{3n}$.
$x[y/z] = y$, x,yi,z distinct. Freely realizable. $x = a^nb^{2n}$, $y = a^nb^n$, $z = a^n$.
$x[y/z] = z$, x,y,z distinct. Not realizable.
$x[y/z] = w$, x,y,z,w distinct. Freely realizable. Set $x = a^nb^n$, $y = a^n$, $z = a^{2n}$, $w = a^{2n}b^n$.

$\alpha[y/z] = w$, α,y,z,w distinct. See below.
$\alpha[y/z] = z$, α,y,z distinct. Not realizable.
$\alpha[y/z] = y$, α,y,z distinct. See below.
$\alpha[y/y] = w$, α,y,w distinct. Not realizable.
$\alpha[y/y] = y$, α,y distinct. Not realizable.

$x[\alpha/z] = w$, x,α,z,w distinct. Freely realizable. Set $x = a^n\alpha$, $z = b^n$, $w = a^nb^n$.
$x[\alpha/z] = z$, x,α,z distinct. Not realizable.
$x[\alpha/z] = x$, x,α,z distinct. Not realizable.
$x[\alpha/x] = w$, x,α,w distinct. If $\alpha \neq ()$ then freely realizable. Let $x = \alpha^n$, $w = \alpha^{n\hat{}2}$. If $\alpha = ()$ then not realizable.
$x[\alpha/x] = x$, x,α distinct. Freely realizable. If $\alpha = ()$ then obviously free realizable. If $\alpha \neq ()$ then let α not begin with c. Set $x = c^n$.

$x[y/\alpha] = w$, x,y,α,w distinct. Freely realizable. Set $x = a^nb^n$, $y = a^n$, $w = \alpha b^n$.
$x[y/\alpha] = y$, x,y,α distinct. Freely realizable. Set $x = \alpha^{n\hat{}2}$, $y = \alpha\hat{}n$.
$x[y/\alpha] = x$, x,y,α distinct. Set $x = a^n$, $y = b^n$.
$x[x/\alpha] = w$, x,α,w distinct. Not realizable.
$x[x/\alpha] = x$, x,α distinct. Not realizable.

$x[y/z] = \alpha$, x,y,z,α distinct. See below.
$x[y/y] = \alpha$, x,y,α distinct. Not realizable.
$x[y/x] = \alpha$, x,y,α distinct. See below.
$x[x/y] = \alpha$, x,y,α distinct. Not realizable.
$x[x/x] = \alpha$, x,α distinct. Not realizable.

$x[y/\alpha] = \beta$, x,y,α distinct, x,y,β distinct. See below.
$x[x/\alpha] = \beta$, x,α distinct, x,β distinct. If $\alpha = \beta$ then freely realizable, by setting $x = \alpha^n$. If $\alpha \neq \beta$ then not realizable.
$x[\alpha/z] = \beta$, x,z,α distinct, x,z,β distinct. See below.
$x[\alpha/x] = \beta$, x,α distinct, x,β distinct. See below.
$x[\alpha/\beta] = w$, x,w,α distinct, x,w,β distinct. If $\alpha = ()$ or $\alpha = \beta$, then not realizable. Suppose $\alpha \neq ()$, $\alpha \neq \beta$. Let α not begin with c. Then freely realizable by setting $x = \alpha c^n$, $w = \beta c^n$.
$x[\alpha/\beta] = x$, x,α distinct, x,β distinct. If $\alpha = ()$ then freely realizable. Suppose $\alpha \neq ()$, and let α not begin with c. Freely realizable. Set $x = c^n$.
$\alpha[y/z] = \beta$, y,z,α distinct, y,z,β distinct. See below.
$\alpha[y/y] = \beta$, y,α distinct, y,β distinct. If $\alpha = \beta$ then freely realizable. If $\alpha \neq \beta$ then not realizable.

$\alpha[y/\beta] = w$, y,w,α distinct, y,w,β distinct. See below.
$\alpha[y/\beta] = y$, y,α distinct, y,β distinct. See below.
$\alpha[\beta/z] = w$, z,w,α distinct, z,w,β distinct. See below.
$\alpha[\beta/z] = z$, z,α distinct, z,β distinct. If β is not a proper substring of α, or $\beta = ()$, then not realizable. If β is a nonempty proper substring of α, then not realizable.
$x[\alpha/\beta] = \gamma$, $x \neq \alpha,\beta,\gamma$. See below.
$\alpha[y/\beta] = \gamma$, $y \neq \alpha,\beta,\gamma$. See below.
$\alpha[\beta/z] = \gamma$, $z \neq \alpha,\beta,\gamma$. See below.
$\alpha[\beta/\gamma] = w$, $w \neq \alpha,\beta,\gamma$. See below.

So it remains to consider only the following lead equations.

1. $\alpha[y/z] = w$, α,y,z,w distinct.
2. $\alpha[y/z] = y$, α,y,z distinct.
3. $x[y/z] = \alpha$, x,y,z,α distinct.
4. $x[y/x] = \alpha$, x,y,α distinct.
5. $x[y/\alpha] = \beta$, x,y,α distinct, x,y,β distinct.
6. $x[\alpha/z] = \beta$, x,z,α distinct, x,z,β distinct.
7. $x[\alpha/x] = \beta$, $x \neq \alpha,\beta$.
8. $\alpha[y/z] = \beta$, y,z,α distinct, y,z,β distinct.
9. $\alpha[y/\beta] = w$, y,w,α distinct, y,w,β distinct.
10. $\alpha[y/\beta] = y$, $y \neq \alpha,\beta$.
11. $\alpha[\beta/z] = w$, z,w,α distinct, z,w,β distinct.
12. $x[\alpha/\beta] = \gamma$, $x \neq \alpha,\beta,\gamma$.
13. $\alpha[y/\beta] = \gamma$, $y \neq \alpha,\beta,\gamma$.
14. $\alpha[\beta/z] = \gamma$, $z \neq \alpha,\beta,\gamma$.
15. $\alpha[\beta/\gamma] = w$, $w \neq \alpha,\beta,\gamma$.

In each of these 15 cases, we must give a decision procedure for determining whether any extension by $v_1 \neq a_1$, ..., $v_n \neq a_n$, where $v_1,...,v_n$ are variables among those appearing in the case, and the a's are character strings, is realizable.

In some of these cases, we give very crude algorithms, a few of which are even exponential. We have no doubt that efficient linear time algorithms can be given with much more careful analyses.

LEMMA 2.4. Let $n \geq 0$. No nonempty proper initial substring of a^nb^n is a tail of a^nb^n.

Proof: Any nonempty proper initial segment has more a's than b's. Any nonempty proper tail has more b's than a's. QED

1. $\alpha[y/z] = w$, α,y,z,w distinct. Then y is a nonempty proper substring of α. There are only finitely many relevant y. Thus it suffices to determine realizability of extensions of

$\alpha[\beta/z] = w$, α,β,z,w distinct.

This is handled in 11.

2. $\alpha[y/z] = y$, α,y,z distinct. Then y is a nonempty proper substring of α. There are only finitely many relevant y. Thus it suffices to determine realizability of extensions of

$\alpha[\beta/z] = \beta$, α,β,z distinct.

This is handled in 14.

3. $x[y/z] = \alpha$, x,y,z,α distinct. Then y is a nonempty proper substring of x, and so z is a proper substring of α. There are only finitely many relevant y,z. Thus it suffices to determine realizability of extensions of

$x[\beta/\gamma] = \alpha$, x,β,γ,α distinct.

This is handled in 12.

4. $x[y/x] = \alpha$, x,y,α distinct. Then y is a nonempty proper substring of x, and so x is a proper substring of α. There are only finitely many relevant x,y. Thus it suffices to determine realizability of extensions of

$\beta[\gamma/\beta] = \alpha$.

Since there are no variables left, there are no proper extensions, in which case realizability is the same as truth.

5. $x[y/\alpha] = \beta$, x,y,α distinct, x,y,β distinct. Then y is a nonempty proper substring of x.

case 1. $\alpha = \beta = ()$. Freely realizable. Set $x = a^{2n}$, $y = a^n$.

case 2. $\alpha = ()$, $\beta \neq ()$. Freely realizable. Let β not begin with c. Set $x = c^n\beta$, $y = c^n$.

case 3. $\alpha \neq ()$. For realizability, it is necessary to be able to write β as $\gamma_1\alpha\gamma_2\alpha...\gamma_k$, $k \geq 2$. Let $\beta = \gamma_1\alpha\gamma_2\alpha...\gamma_k$, $k \geq 2$. We have free realizability, since we can set $x = \gamma_1 a^n b^n \gamma_2 a^n b^n ... \gamma_k$, $y = a^n b^n$. Apply Lemma 2.4, to see that the successive copies of y in x are as displayed.

6. $x[\alpha/z] = \beta$, x,z,α distinct, x,z,β distinct. Then α is a nonempty proper substring of x, and z is a nonempty substring of β. There are only finitely many relevant z. Thus it suffices to determine realizability of extensions of

$x[\alpha/\gamma] = \beta$, $x \neq \alpha,\beta,\gamma$.

This is handled in 12.

7. $x[\alpha/x] = \beta$, $x \neq \alpha,\beta$. Then α is a nonempty proper substring of x, and so x is a substring of β. There are only finitely many relevant x. Thus it suffices to determine realizability of extensions of

$\gamma[\alpha/\gamma] = \beta$.

Since there are no variables left, there are no proper extensions, in which case realizability is the same as truth.

8. $\alpha[y/z] = \beta$, y,z,α distinct, y,z,β distinct.

case 1. $\alpha = \beta$. Then we have free realizability, by setting $y = a^n$, $z = b^n$.

case 2. $\alpha \neq \beta$. Then y is a nonempty proper substring of α. There are only finitely many relevant y. Thus it suffices to determine realizabilty of extensions of

$\alpha[\gamma/z] = \beta$, $z \neq \alpha,\gamma,\beta$.

This is handled in 14.

9. $\alpha[y/\beta] = w$, y,w,α distinct, y,w,β distinct. Then y is a nonempty proper substring of α. There are only finitely many relevant y. Thus it suffices to determine realizability of extensions of

$\alpha[\gamma/\beta] = w$.

This is handled in 15.

10. $\alpha[y/\beta] = y$, $y \neq \alpha,\beta$. Then y is a nonempty proper substring of α. There are only finitely many relevant y. Thus it suffices to determine realizability of extensions of

$\alpha[\gamma/\beta] = \gamma$.

Since there are no variables, there are no proper extensions, in which case realizability is equivalent to truth.

11. $\alpha[\beta/z] = w$, z,w,α distinct, z,w,β distinct.

case 1. β is a nonempty proper substring of α. Then freely realizable, by setting $z = a^n$, $w = \alpha[\beta/a^n]$.

case 2. Otherwise. Then $\alpha = w$, and hence not realizable.

12. $x[\alpha/\beta] = \gamma$, $x \neq \alpha,\beta,\gamma$. Then α is a nonempty proper substring of x.

case 1. $\beta = ()$. If we have realizability, then we have free realizability, since we add any number of α's at the end of x. Note that if the number of α's that are being replaced exceeds the length of γ, then there are two adjacent α's that are being replaced. One of these can be removed. This establishes a bound on the x of least length.

case 2. $\beta \neq ()$. Note that the number of copies of α that are being replaced must be at most $\text{len}(\gamma)$, and what is left over in α must have length at most $\text{len}(\gamma)$. So the length of x must be at most $\text{len}(\alpha)(\text{len}(\gamma)+1)$. This bounds the lengths of the relevant x.

13. $\alpha[y/\beta] = \gamma$, $y \neq \alpha,\beta,\gamma$. Then y is a nonempty substring of α. There are only finitely many relevant y.

14. $\alpha[\beta/z] = \gamma$, $z \neq \alpha,\beta,\gamma$.

case 1. β is a nonempty substring of α. The relevant z are of length at most that of γ.

case 2. Otherwise. We have free realizability, since z can be arbitrary.

15. $\alpha[\beta/\gamma] = w$, $w \neq \alpha,\beta,\gamma$. The only w to be considered is $\alpha[\beta/\gamma]$.

* This material is based upon work supported by the National Science Foundation under Grants No. DMS-0701260 and CCF-0811737. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.