STRING REPLACEMENT SYSTEMS

by
Harvey M. Friedman
Distinguished University Professor
of Mathematics, Philosophy, Computer Science Emeritus
Ohio State University
Columbus, Ohio
https://u.osu.edu/friedman.8/foundational-adventures/downloadable-manuscripts/

CIE 2022
July 13, 2022

https://u.osu.edu/friedman.8/foundational-adventures/downloadable-manuscripts/
118. String Replacement Systems, May 6, 2022, 26 pages.

It is commonplace that the usual definitions of the basic notions of recursion theory and computational complexity are based on computational models with ad hoc features. Famously, in the early period of development, there was much work done on variant computational models, proving that all kinds of reasonable variations lead to equivalence.

Here we go back to basics, and develop particularly simple models of computation with arguably no ad hoc features, and prove that it does lead to the same classes.

The basic idea is to use what we call string replacement systems. A String Replacement System, or SRS, consists of a list of finitely many rules

$x_1 \to y_1$
...
$x_r \to y_r$

where $x_1,\dots,x_r,y_1,\dots,y_r$ are nonempty finite strings of nonnegative integers (N).

For greater control, we define an SRS(n) to be a finite list of rules

$x_1 \to y_1$

...

$x_r \to y_r$

where $x_1,\ldots,x_r,y_1,\ldots,y_r \in \{0,\ldots,n\}^*$. Here n is a nonnegative integer and A* is the set of all nonempty finite strings from A.

SRS can be viewed as a simplified kind of formal grammar (Post, Chomsky). This has the conceptual advantage of not relying on ad hoc choices in the specific models of computation. This purity also suggests possible theoretical interactions with chemistry and molecular biology.

Of particular importance are bit strings, or the elements of SRS(1). Some of our results use bit strings only.

Let α be a SRS(n). The execution sequences are nonempty finite or infinite sequences of strings $w_1,w_2,\ldots$ from $\{0,\ldots,n\}^*$, where each w_{i+1} is obtained from w_i by choosing a rule $x \to y$ and replacing some consecutive substring (which is a copy of) x in w_i by y. Thus there is nondeterminism in two ways. First in the choice of rule and second in the choice of position of the consecutive substring.

A terminal execution sequence is a finite execution sequence which cannot be properly extended to an execution sequence. I.e., no rule can be applied to the last term.

The acceptance set of α is the set of first strings of the various terminal execution sequences.

The SRS(n) acceptance sets are the acceptance sets for the various SRS(n).

SRS(0) involves a single letter alphabet (unary strings), and SRS(1) involves a two letter alphabet (bit strings).

THEOREM 1. (obvious). Every SRS(n) acceptance set is an r.e. subset of $\{0,\ldots,n\}^*$.

THEOREM 2. The SRS(0) acceptance sets are exactly the sets $\{0^{(i)}: i < k\}$, $0 \le k \le \infty$.

THEOREM 3. Let $0 \le n \le m$. If $A \subseteq \{0,\ldots,n\}$ then A* is a SRS(n) acceptance set. Every SRS(n) acceptance set is an SRS(m) acceptance set.

THEOREM 4. There is a complete r.e. set among the SRS(1) acceptance sets.

Ideally, we would like to have

1) Let $n \geq 1$. The r.e. subsets of $\{0,...,n\}^*$ are exactly the SRS(n) acceptance sets.

THEOREM 5. Let $n \geq 0$. The SRS(n) acceptance sets are a proper subset of the r.e. subsets of $\{0,...,n\}^*$. In fact, every nonempty SRS(n) acceptance set has an element of length 1.

To see this, assume SRS(n) accepts $x \in \{0,...,n\}^*$ and let $y \in \{0,...,n\}^*$ be terminal. Then every letter in y must itself be terminal since otherwise y would not be terminal.

We have proved modified forms of 1), not only with r.e., but with P, NP, PSPACE.

The development starts with the following.

THEOREM 6. For all $n \geq 0$ there exists $m > n$ such that the r.e. subsets of $\{0,...,n\}^*$ are exactly the intersections of the SRS(m) acceptance sets with the SRS(m) acceptance set $\{0,...,n\}^*$.

This is proved straightforwardly using a standard nondeterministic Turing machine model that yields the r.e. subsets of $\{0,...,n\}^*$. The $m > n$ corresponds to the number of auxiliary tape and state symbols used in the nondeterministic Turing machine model.

This is improved to the following and is less straightforward.

THEOREM 7. For all $n \geq 0$, the r.e. subsets of $\{0,...,n\}^*$ are exactly the intersections of the SRS(n+1) acceptance sets with the SRS(n+1) acceptance set $\{0,...,n\}^*$.

Note that this is particularly interesting with $n = 0$.

THEOREM 8. The r.e. subsets of $\{0\}^*$ are exactly the intersections of the SRS(1) acceptance sets with $\{0\}^*$.

It remains to characterize the r.e. subsets of $\{0,...,n\}^*$ using just some variant of SRS(n) acceptance sets. We have already

seen from Theorem 5 that we have to modify the SRS(n) acceptance sets to achieve this goal.

We achieve this goal by introducing premier execution sequences. These have an added feature that they are deterministic. Here at each stage we must apply some rule at the first possible position, and then choose the first such rule. The premier acceptance set of SRS(n) is the set of first terms of its terminal premier execution sequences. The premier SRS(n) acceptance sets are the terminal premier execution sequences of the SRS(n).

Using more combinatorial ideas, we prove the following.

THEOREM 9. For all n ≥ 1, the r.e. subsets of {0,...,n}* are exactly the positive Boolean combinations of the premier SRS(n) acceptance sets.

And the case n = 1 is of special interest since now we only use bit strings.

THEOREM 10. The r.e. subsets of {0,1}* are exactly the positive Boolean combinations of the premier SRS(1) acceptance sets.

So for just bit strings, we have Theorem 10 and

THEOREM 8. The r.e. subsets of {0}* are exactly the intersections of the SRS(1) acceptance sets with {0}*.

THEOREM 4. There is a complete r.e. set among the SRS(1) acceptance sets.

We now adapt this development to NP and PSPACE instead of just r.e.

We use a polynomial bound on the length of terminal execution sequences of SRS(n+1). The polynomial is applied to the initial string. Thus we speak of the SRS(n+1)/PL acceptance sets. For fixed n, these are obviously in NP.

We also uae a polynomial bound on the length of the strings used in the terminal execution sequences of SRS(n+1). The polynomial is applied to the initial string. Thus we speak of the SRS(n+1)/PS acceptance sets. For fixed n, these are obviously in PSPACE.

THEOREM 11. Let $n \geq 1$. $S \subseteq \{0,...,n\}^*$ is NP if and only if S is the intersection of an SRS(n+1)/PL acceptance set with $\{0,...,n\}^*$.

THEOREM 12. Let $n \geq 1$. $S \subseteq \{0,...,n\}^*$ is PSPACE if and only if S is the intersection of an SRS(n+1)/PS acceptance set with $\{0,...,n\}^*$.

We can also handle PTIME. Here we use a polynomial bound on the length of *premiere* terminal execution sequences. Thus we speak of the premiere SRS(n+1)/PL acceptance sets.

THEOREM 13. Let $n \geq 1$. $S \subseteq \{0,...,n\}^*$ is P if and only if S is a positive Boolean combination of premier SRS(n)/PL acceptance sets.

FURTHER RESEARCH

We don't have a good understanding of what the SRS(n) acceptance sets are, $n \geq 1$. We also don't have a good understanding of what the premier SRS(n) acceptance sets are.

What can we say about the positive Boolean combinations that suffice for Theorems 9,10,13?

How can we characterize the r.e., P, NP, and PSPACE subsets of $\{0,...,n\}^*$ in terms of the SRS(n) acceptance sets instead of going to SRS(n+1) or premiere SRS(n).

How can we likewise treat the more refined complexity classes used in computational complexity theory, instead of the relatively crude r.e., P, NP, PSPACE?

The purity of the string replacement systems, especially bit string replacement systems, and perhaps even with premier execution, suggests possible theoretical interactions with chemistry and molecular biology. For instance, those sciences may suggest additional models of processing related to the present string replacement systems, worthy of analogous mathematical investigation.