Peart

-7

acm PRESS

SIGACT NEWS

PUBLISHED BY THE ACM SPECIAL INTEREST GROUP
ON AUTOMATA AND COMPUTABILITY THEORY

Volume 18 Number 3 Spring 1987 (Whole Number 65)

Table of Contents

REPRESENTING PROBLEMS AS STRING TRANSFORMATIONS*

K.L. Williams, Western Michigan University
M.R. Meybodi, Ohio University

Introduction

A string transformation function operates on an input string to give an output string which is a permutation of the input string. (See [3]). Common examples include string reversal, string rotation, matrix transformation and sorting. We show here that a variety of much more complex problems, including problems which are not computable, problems relating directly to the computing function of digital computers, and classic problems such as SAT (satisfiability of a boolean function) may also be posed as string transformation problems.

Problem Representation

Let I be a string over alphabet Σ. $\forall I \in \Sigma^*$, define $f(I) = \omega$ where ω is the reversal of I. Then f is an example of a string transformation function.

It is easy to see that solutions to problems such as sorting may be represented directly as string transformation functions. It may appear that string transformations represent only a small class of well understood, easily solved problems. This, however, is not the case.

The following string transformation is not computable. Let $w_1, w_2, \ldots$ be an enumeration of all strings from {0, 1} and $T_1, T_2, \ldots$ be an enumeration (encoded into strings of binary digits) of all Turing Machines over input alphabet {0, 1}. For string x composed of the digits {0, 1} consider transformation:

$$f(x01) = \begin{cases} x01 \text{ if } x = w_i \text{ and } w_i \text{ is accepted by } T_i \\ x10 \text{ if } x = w_i \text{ and } w_i \text{ is not accepted by } T_i \end{cases}$$

In order for f to be computable both $S_1 = \{x01 | x = w_i \text{ and } w_i \in L(T_i)\}$ and $S_2 = \{x01 | x = w_i \text{ and } w_i \notin L(T_i)\}$ must be recursively enumerable. (See [2]). S_2, however, is not recursively enumerable therefore f is not computable.

Alternatively, let $x = \langle T_i, w_j \rangle$ (x is an encodement of Turing Machine T_i followed by input string w_j) and define

$$g(x01) = \begin{cases} x01 \text{ if } w_j \text{ is not in } L(T_i) \\ x10 \text{ if } w_j \text{ is not in } L(T_i) \text{ and } T_i \text{ halts on input } w_j \\ \text{undefined otherwise} \end{cases}$$

*This work was partially funded by Western Michigan University Grant FRCAF#86-028.

g, then, is a partial recursive function which can be computed for all values where it is defined.

One may use the above technique to represent any language acceptance problem as a string transformation problem as long as the class of accepting machines is such that a string encodement can be defined to uniquely represent each machine in the class. This is easily done for those classes of machines that are often used as models of computation including dfa, fa, pda, Turing Machines, RAM machines etc. Those strings from the appropriate alphabet that do not represent an actual machine may be treated as representing a "null machine" which accepts precisely language $\varnothing$. In fact, the characteristics of any digital computer can be encoded into a string and the above technique can be applied to create a string transformation reflecting whether or not an actual computer accepts a string.

Of course, for most applications of digital computers the question is not whether the computer accepts a string but rather what output the computer will produce for a given input. Again we observe that a given input sequence and output sequence for a computer can each be encoded into a string. Even in the case of real-time applications, appropriate encodements including timing information, input and output unit characteristics etc. can be developed. A string transformation can then be defined to represent the action of computer C on input I producing output Ω. Let $x = \langle C, I, G \rangle$. Define:

$$T(x01) = \begin{cases} x01 \text{ if C operating on I produces G} \\ x10 \text{ otherwise} \end{cases}$$

Problems such as SAT (see [1]) can be represented directly as string transformations. Let x be a boolean function of several variables. Define:

$$S(x01) = \begin{cases} x01 \text{ if } x \text{ is satisfiable} \\ x10 \text{ otherwise} \end{cases}$$

It is clear that S is an NP-Complete function.

From the above examples it can be seen that string transformations, which are a natural representation for some problems, provide an alternate representation system for a much wider class of problems.

References

[1] Garey, M. R. and Johnson, D. S., Computers and Intractability, Bell Laboratories, 1979.

[2] Hopcroft, J. and Ullman, J., Introduction to Automata Theory, Languages and Computation, Addison-Wesley, 1979.

[3] Williams, K.L. and Meybodi, M.R., "Notes on Parallel Computation for String Transformation Problems", Proceedings, IEEE Phoenix Conference on Computers and Communications, 1986.