Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 12, December 2015, pg.135 – 138

Complexity Measure of Turing Machine

Rajesh Kumar
CRM Jat College, Hisar, Haryana, India
rajtaya@kuk.ac.in

Abstract— Given a number of tape symbols, the author define the state complexity of a partial-recursive function f as the minimal number of states required for a Turing machine that computes f. Another definition gives the state complexity of recursively enumerable sets and hence of abstract languages. It can be shown that all complexity classes are finite and non-empty. If a certain value of state complexity is surpassed, then every language state complexity class contains at least one language of each of the following types: finite, nonfinite regular, nonregular context-free, context-sensitive but not context-free, recursive but not context- sensitive, enumerable but not recursive. The complexity of a language or a function is closely related to the amount of description or information needed to define the function or language.

I. INTRODUCTION

In this paper the author introduce and investigate via Turing machines the complexity of Turing machines, recursive functions and formal languages. It is start from the fact that every partial recursive function and every recursively- enumerable language is computed (respectively enumerated) by an appropriate Turing machine (T-machine). We then define the complexity of a partial recursive function f as the minimal number of states necessary for a T-machine that computes f. This is well defined if the tape alphabet is restricted in a suitable way. Thus it is obtained an enumerable infinite set of finite complexity classes (S-complexity classes). Some of the properties of this hierarchy are investigated. The S-complexity of a language of a function can be considered to be a measure for the amount of description or information needed to define the language or the function. We suppose that similar considerations on the length of grammars or other descriptive means for languages would yield similar results.

II. DEFINITIONS ON TURING MACHINES

A Turing machine (T-machine) is a 6-tuple $T = (X, Y, Z, r, z_A, z_s)$,where

$X = \{x_0, x_1, x_2, ..., x_m\}$ $(m \geq 1)$ is the tape alphabet,

$Y = X \cup \{R, L\}$ $(R, L \notin X)$ is the operation alphabet (R = right, L= left),

Z is the finite set of states,

$r: Z \times X \rightarrow Z \times Y$ is a mapping,

$z_A \in Z$ the initial state and $z_S \in Z$ the final state.

In addition, $z_A \neq z_s$ must be true. Therefore the initial state and the final state are always different. From the existential point of view, a Turing machine is a finite state automaton that moves(Left and Right) on a tape without ends. The automata picks up its input signals from the tape by a read head and the output may be an elementary move to the right or the left or a change of the scanned symbol under the read head. During this operation, the internal state of the automata is changed too. The set of finite strings over X is denoted by X*, the abbreviations $L_x:=x_0x_0x_0$ and $R_x=x_0x_0x_0$ denote strings of infinite length with the symbol $x_0 \in X$ that are extended to the left side respectively right side. These strings are used to indicate that the left side respectively right side of the tape is filled up to infinity with the blank symbol x_0.The empty string (length O) of X* is denoted by Λ.

We call a triple c= (z, L_xP, xQR_x) configuration of the T-machine if $z \in Z$, p,q$\in$ X* and x$\in$ X. L_xPxQR_x denotes the complete tape inscription, and x is the symbol under the head. A configuration c=(z, L_xP, xQR_x) is called an end configuration if $z=z_s$.

In what follows, we assume that the reader is familiar with the basic facts of the theory of computability and mathematical language theory (e.g. see Davis 1958 and Aho, Ullman 1968)

Definition 1. Each T-machine T =(X, Y, Z, r, z_A ,z_S) defines in the following way a performance function (P-function) $f_T : N \rightarrow N \cup \{\hat{S}\}$, where N = {0, 1, 2, 3,..} and $\hat{S} \notin N$ a special range symbol:

$$f_T(n):=\{//(z_A, L_x, x_1^n R_x)// \text{ if defined }, \hat{S} \text{ else }.$$

x_1^n denotes a string of n consecutive symbols $x_1 \in X$, and we allow n=0. The so-defined functions are in the strict sense not recursive for all T-machines. But if f_T is restricted to the domain $\{n\ f_T(n) \neq \hat{S}\}$, the set of all functions f_T is equal to the set of all partial-recursive functions. With this in mind we call f_T the partial-recursive function computed or defined by T.

Definition 2. Each T-machine T =(X, Y, Z, r, z_A ,z_S)defines in the following way a language

$L_T \subset X'^* : = (X-\{x_0\})^*$: $L_T : = \{x_{i1}x_{i2} .....x_{is} \mid x_{ij} \in X'$ and // (z_A , $L_x x_{i1}x_{i2} .....x_{is} R_x$)// defined}

Say L_T is definable by T.

A word w belongs to the language L_T iff the machine T, started with the configuration (z_A, L_x, wR_x) reaches an end configuration, that is, it halts and w$\in$ X'*, where X ' : = X-{x_o}. The so-defined languages correspond exactly to the recursively enumerable languages.

We denote by Z_T the set of states of the T-machine T, therefore $|Z_T|$ gives the number of states of T.

Definition 3. $\mathcal{F}_{X,n} : = \{ f \mid f : N \rightarrow N \cup \{\hat{S}\}$ and $Com_x(f)=n\}$. $\mathcal{F}_{X,n}$ is called the complexity class with index *n* of the partial recursive functions.

$\mathcal{L}_{X,n}: = \{L \mid L \subset X'^*$ and $Com_x(L)= n\}$.

$\mathcal{L}_{X,n}$ is called the complexity class with index n of the recursively enumerable languages.

We have assumed that $|X| \geq 2$, and hence each partial recursive function occurs in one of the classes $F_{X,n}$. It is seen immediately that $\mathcal{F}_{X,n} \cap \mathcal{F}_{X,m} = \square$ and $\mathcal{L}_{X,n} \cap \mathcal{L}_{X,m} = \square$ if $n \neq m$. By simple combinatorial methods (calculation of an upper bound for the number of all T-machines with fixed X and Z)it is possible to prove

$$|\mathcal{F}_{X,n}|, |\mathcal{L}_{X,n}| \leq n(n-1)[n(m+3)]^{n(m+1)}$$

The complexity classes $\mathcal{F}_{X,n}$ and $\mathcal{L}_{X,n}$ are finite. Now we are able to state the main theorem on the complexity classes:

Theorem 1. If n≥2, then $\mathcal{F}_{X,n} \neq \square$, all proper complexity classes are nonempty.

Proof. We define the sequence of functions $f_j : N \rightarrow N$ by $f_j(k) := k+ j$ where j = 0, 1, 2,.... .

(1) We have $Com_x(f_0)=2$ by the T-machine $T_0=(X, Y, Z, r, z_A, z_s)$ with the structure $Z_0:=\{z_A, z_s)$, r(z, x):= (z_s, x) for all $z \in Z_0$ and x$\in$X. There are according to our definition no machines with fewer than two states. Therefore T_0 is f-minimal and has the property $f_0 = f_{T0}$.

(2) Now we assume that T_j = (X,(X, Y, Z, r, z_A, z_s)is an f-minimal machine that defines f_j. With the help of this machine we can build up a machine T' which has exactly one state more than T_j and defines f_{J+l}. T' is defined by

T'=(X, Y, Z$\cup$ {z_s' } r', z_A ,z_s'), z_s' is an additional state ($z_s' \notin$ Z),

$$r'(z,x)= \begin{cases} r(z,x) & \text{if } z \in Z-\{z_s, z_s'\} \text{ and } x \in X \\ (z_s,L) & \text{if } z=z_s \text{ and } x \neq x_0 \\ (z_s',x_1) & \text{if } z=z_s \text{ and } x= x_0 \\ (z_s',x) & \text{if } z=z_s' \end{cases}$$

T' behaves almost like T_j with one slight variant at the end of the computation:

If T_j halts, T' runs on the tape to the left seeking a blank symbol x_0. If such a symbol is found, the machine changes it to x_l and stops. Therefore $f_{T'} = f_j+1$, and we conclude that an f-minimal machine defining f_j+l has at most one state more than any minimal machine for f_j.

(3) Let T_0, T_1, T_2 ,... be a sequence of f-minimal machines with $f_{Tj} = f_j$.
We then have $| Z_{Tj}|+1 \geq |Z_{Tj+1}|$ in other words, upward increments of the sequence $n_j=Comf_x(T_j)$ of natural numbers are at most one. Each number $n \geq 2$ can occur only finitely often in the sequence because we have $\mathcal{F}_{X,n}$ finite and $f_i \neq f_j$ if $i \neq j$. Therefore the sequence must grow to infinity and each natural number $n \geq 2$ must occur. This completes the proof of the theorem.

III. THE STATE COMPLEXITY CLASSES OF LANGUAGES AND THE CHOMSKY HIERARCH

The basic idea of the last proof can help us to get further results on the S-complexity of languages.
Theorem 2. If $x \in X'$, L a language over X ' and $Com_x(L)=n$, then the language

$$xL = \begin{cases} |\{xw|\ w \in L\} & \text{if } \square\square L, \\ |\{xw|\ w \in L\} \cup \{\square\} & \text{if } \square\square L, \end{cases}$$

has complexity $Com_x(xL) \leq n+1$.
Proof. Let $T=(X, Y, Z, r, z_A, z_s)$ be a machine that defines L. With the help of this machine we can build up a machine $T_{x'}$ with the property $L_{Tx''}$ where x' is an element of X'. Define

$$T_{x'}: =(X, Y, Z \cup \{z_A'\}, r', z_A', z_s) \text{ where } z_A' \notin Z \text{ is a new initial state,}$$

$$r'(z,x)= \begin{cases} |r(z,x) & \text{if } z \in Z \text{ and } x \in X \\ |(z_A',x) & \text{if } x \neq x', x_0, \quad z=z_A' \\ |(z_A',x_0) & \text{if } x=x' \quad z=z_A' \\ |(z_A',R) & \text{if } x=x_0, \quad z=z_A' \end{cases}$$

The machine $T_{x'}$ acts as follows: If the first symbol on the tape (at the beginning under the read head) is x', $T_{x'}$ rewrites x' as x_0 (blank symbol), goes one step to the right and activates machine T (initial state za). If the first symbol is not x' or x_0, then $T_{x'}$ never halts (loop in z_A'). If the first symbol is the blank symbol x_0 ,then the machine T is activated without any change of the tape inscription, therefore $\square\square L_{Tx'}$ iff $\square\square L_T$. Thus the machine $T_{x'}$ defines the language

$$L_{Tx'}= x'L_T$$

Since $T_{x'}$ has exactly one state more than T, the theorem is proved.
Theorem 3. Let X be a tape alphabet.
(1) $\mathcal{L}_{X,n}$ contains for $n \geq 2$ a finite language and a nonfinite regular language.

(2) If $|X'| \geq 2$, then there exists a constant C_1^x such that for all $n \geq C_1^x$ $\mathcal{L}_{X,n}$ contains a context-free nonregular language and no other classes contain such languages.

(3) There exists a constant C_2^x with the following property: Each class $\mathcal{L}_{X,n}$ with $n \geq C_2^x$ contains a context-sensitive non context-free language and no other class contains such a language.

(4) There exists a constant C_3^x with the following property: Each class $\mathcal{L}_{X,n}$ with $n \geq C_3^x$ contains a recursive and not context-sensitive language and no other class cantains such a language.

(5) There exists a constant C_4^x with the following property: Each class $\mathcal{L}_{X,n}$ with $n \geq C_4^x$ contains a non-recursive language and recursively enumerable and no other class contains such a language.

Proof. Let L_0 be an arbitrary language with $L_0 \subset X'^*$, $L_0 \neq \square$ and $\square\square L_0$.
The languages L_i defined by $L_i : = x_k L_{i-1} = x_k^i L_0$ ($x_k \in X'$) have the property
$L_i \neq L_j$ if $i \neq j$, since the shortest word of L_i is longer than the shortest word of L_i if $i > j$. By the method used in the proof of Theorem 1 and by Theorem 2 we have the following assertion: If $Com_x(L_0) = C \in N$, then at
least one of the languages L_i occurs in each complexity class $\mathcal{L}_{X,n}$ with $n \geq C$ because $Com_x(L_i)$ $Com_x(L_{i-1}) + 1$ and the sequence of numbers
ni := Com_x(Li) must goes to infinity. Now we can attack the Theorem.

(1) The finite language $\{\square\}$ has complexity 2, the language $L_0:=\{x_l\}$ consists of exactly one string and has $Com_x(L_0) \leq 3$. All languages $x_l^i L_0 = L_i$ are therefore finite too and each complexity class $\mathcal{L}_{X,n}$ must contain at least one finite language. Now define $L_0 := X'^*$ and $L_i:=x_l^i L_0$. We have $Com_x(L_0) = 2$ by an easy construction. All languages L_i are regular and infinite.

(2) Let $L_0 \subset X'^*$ be a context-free nonregular language. Then $x_i^*L_0$ is context-free nonregular for all i = 1, 2, 3,.... Now, let L_0 be a context-free nonregular language with the lowest complexity possible. Define $C_1^x := Com_x(L_0)$ and each class $\mathcal{L}_{X,n}$ $\mathcal{L}_{X,n}$ with $n \geq C_1^x$ contains a context-free non-regular language. By the definition of C_1^x, lower complexity classes cannot contain context-free and nonregular languages. If □ □L_0, an easy modification of this proof holds.

(3, 4, 5) The method used to proof (2) can be used to prove (3), (4) and (5) too. The details are not tricky.an be used to prove (3), (4) and (5) too. The details are not tricky.
Clearly it would be interesting to get the constants C_1^x ,..., C_4^x for a small X, but this seems to be not a simple problem.

IV. RESULTS REGARDING UNDECIDABILITY

The following argument is the key to all further undecidability results:

Theorem 4. It is recursively undecidable whether an arbitrary given T-machine is f-minimal.
Proof. Let T_1, T_2 , T_3 ,... be an effective enumeration of all T-machines with a fixed X, $|X| \geq 2$, such that $|Z_T^i| \leq |Z_{T+1}^i|$ (i = 1, 2, 3,...). The partial recursive function computed by T_i is denoted by f_i. Define $U_k := \{f_i \mid |Z_T^i| \leq k\}$, U_k is the (finite) set of all partial recursive functions computable by machines with no more than k states. It is easy to see that for all i = 1, 2, 3,... $f_i \in U_{i+2}$. Now define a function g according to

$$g(i) := Min_j[f_j \notin U_{i+2}].$$

If it is decidable whether T-machines are f-minimal, then g(i) is computable, i.e. recursive. Then the recursion theorem (Davis 1958) guarantees the existence of at least one index i_0 with the property $f_{i0} = f_{g(i0)}$. That leads to the contradiction (1) $f_{i0} \in U_{i+2}$,(2) $f_{i0} = f_{g(i0)} \notin U_{i+2}$]. Therefore g cannot be recursive, the f-minimality of T-machines cannot be decidable.
Theorem 5. The functions $Comf_x(T)$ and $ComL_x(T)$ are not computable. There is no such minimization algorithm exist for T-machines, i.e. an algorithm that computes for each given T-machine T a second machine T' such that T' is f-minimal and $f_T = f_{T'}$
.
Proof. We have $Comf_x(T)$ $=|Z_T| \Leftrightarrow$T f-minimal. If $Comf_x(T)$ is computable, the f-minimality of T-machines would be decidable. A similar argument holds for $ComL_x(T)$. The existence of a minimization algorithm imply the computability of $Comf_x(T)$.

REFERENCES

[1] Aho, A. V., and Ullman, J. D. (1968), The Theory of Languages, Math. Systems Theory 2, 97-125.
[2] Blum, M.(1967), On the Size of Machine, Information and Control 11, 257-265.
[3] Davis, M. (1958), " Computability and Unsolvability." McGraw-Hill, New York, Toronto, London.
[4] Schmitt, A. (1968), Uber Die Berechnung minimaler Turinfmaschinen and anderer minimaler System, Elektron. Informationsverarbeit. Kybernetik 4, 313-326.
[5] Shannon, C. E. (1956) , A Universal Turing Machine with two Internal State Automata Studies(Ann of Math, studies 34), Princeton.

