Path Measures of Turing Machine Computations

(preliminary report)

Joachim Biskup

Lehrstuhl für Angewandte Mathematik, insbesondere Informatik
RWTH Aachen
D-5100 Aachen

Germany

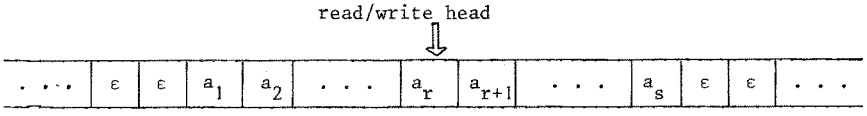
It is well-known that M.BLUM's axioms for computational complexity measures [5] admit very "unnatural" models (cf. [9]). In this note we give a preliminary report on our attempt to define a class of "natural" computational complexity measures. We restrict the discussion to Turing machine computations, however we shall also indicate how to generalize our approach to arbitrary "machine-like" computations.

0. Some notations

$f:A \nrightarrow B$	f is a (possibly) partial function from A into B
$\downarrow$	defined
$\uparrow$	undefined
dom f	domain of f
range f	range of f
$f \circ g$	composition of functions: $f \circ g(x) = f(g(x))$
$\mathbb{N} = \{0,1,2,\ldots\}$	set of natural numbers
$(\psi_i)_{i \in \mathbb{N}}$	acceptable numbering of all partial recursive functions
$(\Phi_i)_{i \in \mathbb{N}}$	associated complexity measure (in the sense of M. BLUM)
card A	cardinality of the set A

1. Fundamental properties of path measures

We consider a Turing machine as an instance of a class of machines as they are studied in the books of S. EILENBERG [8] and W.S. BRAINERD / L.H. LANDWEBER [6].
Let Σ be a fixed finite alphabet and $\Gamma := \Sigma^* \times \Sigma^*$ the set of configurations. A configuration $(a_1 \ldots a_r, a_{r+1} \ldots a_s) \in \Gamma$ is intended to represent the following tape configuration:



A Turing machine can perform conditional instructions of the following form:

if the symbol under the read/write head is equal to σ, then

{ move the read/write head one tape square to the right resp. left
{ print a new symbol

Formally the set of instructions $\mathfrak{F}$ is a finite set of partial functions on Γ. For instance we can formalize the instruction "under condition σ move left" by the following partial function:

$$\sigma\text{-left} : \Gamma \overset{p}{\to} \Gamma,\ \sigma\text{-left}(u,v) := \begin{cases} (\tilde{u},\sigma v) & \text{if } u = \tilde{u}\sigma \\ \uparrow & \text{otherwise} \end{cases}$$

A (deterministic) Turing program $\mathfrak{P}$ is given by a finite set of statements of the form (label, instruction, goto-label), an initial label and a set of terminal labels such that

the label of a statement is not a terminal label, and whenever (l, φ_1, l_1) and (l, φ_2, l_2) are distinct statements of $\mathfrak{P}$, then $\operatorname{dom} \varphi_1 \cap \operatorname{dom} \varphi_2 = \emptyset$.

A path in $\mathfrak{P}$ is a finite sequence of statements of $\mathfrak{P}$ such that for every statement (except the last one) the goto-label equals the label of the next statement. A path P in $\mathfrak{P}$ is called terminating if the goto-label of its last statement is a terminal label of $\mathfrak{P}$; P is called (formally) successful if it is terminating and if the label of its first statement is the initial label of $\mathfrak{P}$. If $P = (l_0,\varphi_1,l_1),\ldots,(l_{n-1},\varphi_n,l_n)$ is a path then the associated (order reversed) instruction sequence $\varphi_n \ldots \varphi_1$ determines a (partial) function $|P| : \Gamma \to \Gamma$, $|P| := \varphi_n \circ \ldots \circ \varphi_1$, which is called the computation of P.

It can easily be shown (cf. [6], lemma 4.1) that for every program $\mathfrak{P}$ and for every configuration $w = (u,v) \in \Gamma$ there exists at most one successful path P in $\mathfrak{P}$ such that $w \in \operatorname{dom} |P|$. A Turing program $\mathfrak{P}$ computes the following function:

$$|\mathfrak{P}| : \Gamma \overset{p}{\to} \Gamma$$

$$|\mathfrak{P}|(w) := \begin{cases} |P|(w) & \text{if P is the uniquely determined successful path in } \mathfrak{P} \text{ such that } w \in \operatorname{dom} |P| \\ \uparrow & \text{if such a path does not exist} \end{cases}$$

The standard measures of time and space are already completely determined by their definitions on paths respectively on instruction sequences associated with paths (cf. [4], § 3). Thus the standard measures are definable along the path structure of programs in an analogous way as we defined above the notion of computation. This observation leeds us to the following attempt to define a class of "natural" computational complexity measures.

First we introduce some further notations. We consider the instruction set $\mathfrak{F}$ as a finite alphabet and denote the free monoid generated by $\mathfrak{F}$ without the empty word by $\mathfrak{F}^*$, that means $\mathfrak{F}^*$ is the set of all finite nonempty instruction sequences. On the other hand each element $\phi = \varphi_n \ldots \varphi_1 \in \mathfrak{F}^*$ determines a (possibly partial) func-