A SIZE INDEX FOR MULTITAPE TURING MACHINES

RAINER GLASCHICK

Abstract. In search of a single number like Shannon's state-symbol product to compare the complicacy of Turing Machines including those with multiple tapes and tape heads, a number called TM index is proposed, using a generic definition for single and multi tape machines. Several examples are shown together with their TM indices, including the recently re-discovered physical machine built by G. Hasenjaeger.

Contents

Date: 2012-09-09.

This work was intiated by an invation as a visiting fellow to the Isaac Newton Institute for the Mathematical Sciences in the programme *Semantics & Syntax*, and I got much help from Turlough Neary and Damien Woods.

1. Introduction

When the Heinz Nixdorf MuseumsForum[1] searched for objects to be presented during the Alan Turing year, a physical model of a Turing machine built by Gisbert Hasenjaeger was located by its director Norbert Ryska. Unlike some other physical models, Hasenjaeger's machine is not only a curiosity, but has also an interesting theoretical background. Searching for the relevance of this machine, the starting point was Shannon's state-symbol product, as he wrote in [Shannon1956]: *An interesting unsolved problem is to find the minimum possible state-symbol product for a universal Turing machine.* Having only four states and three binary tapes, giving the very small state-symbol product of eight, this would indicate an unnoticed milestone. But with the use of more than one tape, Shannon's state-symbol product looses its significance, see e.g. [Hooper1969].

Nevertheless, a single number to indicate the complicacy of a Turing Machine with more than one tape or more heads would be nice indicator for a ranking, although it is clear that such a ranking cannot replace an in-depth analysis.

Hooper proposed the number $m \cdot n^p$ for a machine with m states, n symbols (including blank and end marker) and p tapes, which is equal to the state-symbol product for a single tape machine. However, this number implies that the tapes are all of the same kind, i.e. all provide all operations, have the same alphabet, etc. Hooper used further in his article the triple (m, n, p) to characterise the machines, not the single number obtained from his formula.

In his German text, [Priese1979a] uses two-dimensional tapes and gives three complexity types:

- a quadruple (z, b, d, k), where z is the number of states, b the number of symbols, d the number of tape dimensions, and k the number of heads or tapes
- Hooper's number, i.e. $z \cdot b^k$
- the number of possible instructions, basically $(z \cdot b^k \cdot 2d)^{z \cdot b^k}$

In his English paper [Priese1979b], only the third number, called *complexity*, is used. However, this number is not only very large, so the logarithm is used; it more or less characterises a class of machines, and, like Hooper, does not take into account if tapes are non-erasable, cyclic etc.

In their 2009 paper [NearyW2009], Neary and Woods give an excellent overview of small universal TMs in general, and concentrate on weakly universal ones in particular, all as single tape machines.

Margenstern [Margenstern1993] uses *colours* to characterize Turing Machines with a single tape, and cites Pavlotskaia to prove that a minimum number of colours is necessary for a machine to be universal.

This text gives a (not novel) definition for a Turing Machine, here called *generalised TM*, which embraces TMs with one or more tapes in a single model, and defines a size indicator for it, called *TM index*, that is

- equal to Shannon's state-symbol product for standard single-tape TMs,
- is invariant for two transformations intuitively considered irrelevant,
- gives smaller numbers for restricted machines (*non-erasable*, *weakly universalS*)

Comparing the TM index is logical only for machines that serve the same task, and for universal Turing machines in particular.

Multihead and multitape machines that were introduced and used in the discussion of computational complexity by Hartmanis and Stearns [HartmanisS1965] are likewise covered, but not with examples, see also [JiangSV1997].

[1]in Paderborn, Germany: http://www.hnf.de/

It has been observed that reducing the state-symbol product for TMs often

- increases the computational complexity in time and space,
- requires extra effort to encode and decode the tape contents,
- makes it difficult to determine that the machine stopped.

In particular the complexity issues are not contained in this text, for these, see e.g. [WoodsN2009].

Throughout the text, the abbreviation *TM* is used for *Turing Machine*, *UTM* for *Universal Turing Machine*, *gTM* for *generalised Turing Machine*, *sTM* for *standard Turing Machine*, and *xTM* for *extended Turing Machine.*

2. Generalised Turing Machines

2.1. **Motivation.** Looking at various variants of (deterministic) Turing Machines, they all have in common a finite state machine, that sends symbols and action commands to an unlimited deterministic memory, and in return receives a symbol that is used to advance the finite state machine; see also [Fischer1965a][2].

This definition of a generalised TM could be regarded as a template or class of TMs, as it does not require a specific memory.

Note that several definitions of a TM do not formally define the tape memory. The tape is often introduced by its inscriptions, and the changes to the inscriptions are mostly explained just verbally. One could have the impression that TMs are just finite state machines, and miss the specialty of the tape memory.

The attribute *generalised* is used to distinguish from a *standard TM*, that has a single tape without restrictions and in each state writes to the tape and moves the tape during a single state transition[3].

2.2. **Generalised Turing Machines.** An generalised Turing Machine (gTM) is a finite state organ Σ connected to an unlimited memory organ Ψ, both having an input and an output each, such that the output of the memory Ψ is an input to the state organ Σ, and its output is an input to the memory.

2.2.1. *The finite state subsystem.* The finite state subsystem Σ is a deterministic finite state machine described by the sixtuple $(S, I, \Psi, \sigma, s_0, s_1)$ with:

- a finite set of states $S = s_0, s_1, s_2, \ldots,$
- a finite input alphabet $I = i_1, i_2, \ldots,$
- a finite action alphabet $\Psi = \psi_1, \psi_2, \ldots,$
- a state stepping (partial) map $\sigma : (S - \{s_0\}) \times I \to \Psi \times S$

The state s_1 is called the start state, and the state s_0 the stop state, thus there must not be a step defined for it.

The action alphabet embraces the symbols written as well as the movement of the tape, thus it is different from the input alphabet. This does not exclude that the input symbols as well as the actions may be n-tuples; e.g. for a 3-tape machine, the input symbols might be triples and the actions sextuples; see below for the extended TM. The only restriction is that the memory subsystem does understand the action symbols and produces input symbols for the state machine.

Two-dimensional tapes as in [Priese1979a, Priese1979b] fit perfectly in the above definition: if the tapes are binary, the input alphabet has two members, but the action alphabet has eight instead of four (up and down as well as right and left for

[2]Cooper in his *Computability Theory* [Cooper2003] uses this same model in his definition of a TM, i.e. provides separate actions for writing and movement, but does not extend his model to multi-tape machines.

[3]This is what has become standard now; Turing's machine definition allows several actions combined in one state transition to avoid states with less significance to understanding the function of the machine.

each input symbol), probably as pairs with the symbols to be written. Multihead TMs must keep more state in the memory subsystem.

The stepping map need not be a total function; there may be undefined input combinations, often because they will never occur. See the next section on stop states.

Being defined on finite sets, the stepping map is normally described by some sort of table or matrix; here, a table with four columns will be used: The first two give the current state and input element, the last two the action and the next state.

2.2.2. *The stop state.* In this definition, a stop state is stipulated[4].

Several TM definitions declare the stop of the machine when σ is undefined for the current state and input symbol, or they do not show the stop state in the state table, but have a remark in the text. In these cases, often only the missing line in the state table has to be added, as has been done in some of the examples below. When the stop state is reached, the action output symbol is arbitrary and not sent to the memory, thus any one could be used; this may be indicated by dashes in the state tables, that are not counted as actions.

For other TMs, the stop state is equivalent to a certain cycle of states. This cycle should be at least detectable by a FSM. Then, its states have to be counted too, unless a simple extension of the state machine would do.

2.2.3. *The memory subsystem.* The memory is an unlimited deterministic subsystem using the same sets as the state machine for S, I and Ψ, described by the sixtuple (T, r, w, S, I, Ψ) with:

- a (unlimited) set of states $T = t_1, t_2, \ldots$
- a read function $r : T \to I$ to read the current symbol of the tape state
- a write function $w : T \times \Psi \to T$ to write, i.e. execute an action and return a new tape state

The first memory state t_1 is called the initial memory contents.

2.2.4. *The combination is the gSM.* Combining the state machine and the memory, a generalised Turing Machine is thus described by the tuple:

$$(S, I, \Psi, \sigma, s_0, s_1, T, r, w)$$

Note that the initial memory contents t_1 is not included, in order to allow the application of a single machine to different initial memory contents.

So the application of a gTM to an initial memory contents t_1 generates a list of quadruples (s_j, t_j, i_j, ψ_j) with (for $j > 0$ until $s_{j+1} = s_0$):

$$i_j = r(t_j)$$

$$(s_{j+1}, \psi_j) = \sigma(s_j, i_j)$$

$$i_{j+1} = w(t_j, \psi_j)$$

The symbols i_j and ψ_j are redundant and were used for clarity.

[4]Alan Turing's definition for computable numbers uses cycle-free machines, that are running forever without going into cycles, i.e. repeating a pattern periodically on the tape, as compared to machines used for the solution of the *Entscheidungsproblem.*

2.2.5. *The TM index.* The *generalised TM state table complicacy index*, or *TM index* for short, is defined as a single number by

$$|S| \cdot \sqrt{\frac{|I| \cdot |\Psi|}{2}}$$

For a standard TM with one erasable tape, it is equal to the state-symbol product, see below. For the product $|I| \cdot |\Psi|$, the term *symbol state area size* or simply *IO size* might be used, and for the whole term after the state count the term *symbol equivalent.*

This index, as it uses the cardinality of Ψ, thus depends on which actions are possible on the memory (i.e. if tapes move in only one direction) and the number of symbols that could be written (i.e. for non-erasable or read-only memory), and thus reflects characteristics from the memory subsystem.

This, however, does not honor every quirk in the memory subsystem, see 2.6.6 on penalties. In this case, we call the TM index without penalties the basic TM index.

2.3. Extended Turing Machines.

A extended Turing Machine (xTM) is a special case of the generalised TM, which differentiates the output symbols and the tape movements, using a subset of the input symbols as output symbols.

Here, the action symbols are pairs of elements from a set O of state output symbols and a set Ξ of memory instructions. Naturally, the memory subsystem now must accept these pairs.

The set O of symbols output to the memory is taken from the input symbols, but needs not to cover them all. This allows to express directly that symbols on the tape are only present in the initial tape inscription, and are never written. A new set of memory instructions Ξ is introduced, and the extended TM is a gTM with the tuple $(S, I, \Psi, \sigma, s_0, s_1, T, r, w)$, having:

- the output symbol set $O \subseteq I$
- the memory instructions $\Xi = \{\xi_1, \xi_2, \ldots\}$
- where the memory actions are $\Psi \subseteq O \times \Xi$
- and the memory write function is $w : T \times (O \times \Xi) \to T$

Note that the expression

$$|S| \cdot \sqrt{\frac{|I| \cdot |O| \cdot |\Xi|}{2}}$$

is only equal to the gTM index, if $\Psi = O \times \Xi$, i.e. if all combinations of output symbols and instructions are used as memory actions.

Note that because the set of output symbols O may be a subset of the input symbols I, this allows to express characteristics of non-erasing machines and weak machines already in the state machine, and not only as (mostly verbal) restrictions of the memory subsystem.

Note also, that the input and output symbols as well as the memory instructions might still be (sub-) tuples, (e.g pairs for two tapes).

The extended TM will not be used in the examples, because it looks not suitable to define a useful index.

2.4. Standard Turing Machines.

The standard TM is an xTM, where:

- the set of output symbols is equal to the input symbols: $I = O$
- the set of movement operations has two members $\Xi = \{R, L\}$

Thus, the TM index becomes, because $|\Xi| = 2$:

$$|S| \cdot \sqrt{\frac{|Y| \cdot |Y| \cdot |\Xi|}{2}}$$

$$= |S| \cdot \sqrt{\frac{|Y| \cdot |Y| \cdot 2}{2}}$$
$$= |S| \cdot |Y|$$

which is equal to Shannon's state-symbol product.

For binary machines, there are only 4 action symbols possible, and these are normally all used, if it is a standard TM, even though only two of them can be used per state. This allows all 4 action symbols if the machine has more than 2 states, which has been proved to be necessary anyhow. For a non-erasing binary machine, there may be still 4 actions, so there are $4 \cdot 2$ possible combinations of actions and input symbols per state.

As the examples below show, rather often single tape TMs with many symbols do not use all combinations, so that in these cases the TM index is smaller than the state-symbol product, which might be surprising first, but indicates that the memory has to be less complex than for a full TM.

2.5. **Margenstern's colours.** For a standard TM, Margenstern introduces in [Margenstern1993] the colours of an instruction as the combination of the input symbol and action, i.e. output and movement operation for a gTM. It is possible to look at the set of colours per state and per machine. The maximum numbers of colours per state is $|I| \cdot |O| \cdot 2 = |I|^2 \cdot 2$, as any input can be combined with any output with one out of two movements, and input symbols are the same as output symbols for a standard TM. The set of colors per machine then contains the colours of all instructions of all states. However, it seems to be of limited value for an index number, because it is limited by the above number, even for many states. (If a single state uses up all colours, no other state can add new colours). If follows that the number of colours of a binary machine is 8 at most. (Astonishingly, the set of colours for a machine is often less than the maximum.)

Instead, the sum of the number of colours per state could be used, which is not greater than the product $|I| \times |\Psi|$, the IO size of an gTM. So it might be more specific, but might not be invariant to some transformations, as it heavily depends on the distribution of colours versus states.

It would be interesting to analyze the Theorems given by [Margenstern1993] (some of which were found by Pavlotskaia), if they could be applied to the TM index, but this is outside the scope of this work.

2.6. **Tape Memory.** No attempt will be made to expand the examples by proper tape memory definitions, only some illustrations are shown for standard TMs. Note that the position of the read-write head must be recorded in the state of the memory. It would be desirable to have something like the TM index for the state machine for the memory system, in order to reflect the complicacy of the memory. However, the tape memory is regarded not only as fairly simple in its structure, but there are only a few varieties, that can serve a much larger number of state machines. Also, several of the specialties of the tape memory can be cared for in the gTM definition and state table analysis, except e.g. cyclic tapes. Thus, such an attempt is not tried here.

2.6.1. *Tape memory as a n-tuple.* A fairly simple and flexible memory for a standard TM uses a n-tuple for the tape contents (except the still blank part), and an integral number as an index into to the n-tuple as location for the tape head.

Using $n = |(x_1, \ldots, x_n)|$ for the number of elements in the n-tuple $(x_1, \ldots, x_n)$, and defining the auxiliary functions κ, λ and ζ, we could model the tape state as a set (or pair) of a natural number and an n-tuple of input symbols:

$$t_j = \{k_j, (i_{j,1}, i_{j,2}, \ldots i_{j,k_j-1}, i_{j,k_j}, \ldots i_{j,l_j})\}$$

$$1 \leq k_j \leq l_j$$
$$\kappa(t_j) = k_j$$
$$\lambda(t_j) = l_j = |(i_{j,1}, i_{j,2}, \dots i_{j,k_j-1}, i_{j,k_j}, \dots i_{j,l_j})|$$
$$\forall x \leq l_j : \zeta(t_j, x) = i_{j,x}$$

The read function is just:

$$r(t_j) = \zeta(t_j, \kappa(t_j))$$

For the write function, the action symbol is a pair (y, z) where $y \in O \subseteq I$ and $z \in \{R, L\}$ and the new state depends on the values of z and $k_j = \kappa(t_j)$ as follows, where $l_j = \lambda(t_j)$:

- $z = R, k_j \neq l_j$: $w(t_j, (y, z)) = \{k_j{+}1, (i_{j,1}, i_{j,2}, \dots i_{j,k_j-1}, y, i_{j,k_j+1}, \dots i_{j,l_j})\}$
- $z = R, k_j = l_j$: $w(t_j, (y, z)) = \{Sk_j + 1, (i_{j,1}, i_{j,2}, \dots i_{j,k_j-1}, y, y_0)\}$
- $z = L, k_j \neq 1$: $w(t_j, (y, z)) = \{k_j - 1, (i_{j,1}, \dots, i_{j,k_j-1}, y, \dots, i_{j,l_j})\}$
- $z = L, k_j = 1$: $w(t_j, (y, z)) = \{1, (y_0, y, i_{j,2}, \dots, i_{j,l_j})\}$

where y_0 is the blank symbol. A blank tape is then $t_1 = \{1, (y_0)\}$ with $\kappa(t_1) = 1$, so the first operation extends it to two elements.

This scheme extends easily to multi-tape machines and multi-dimensional tapes. Cyclic tapes are also not difficult to integrate, as they have a fixed size and thus the n-tuple with the contents is initially predefined and does not change, and the tape position is taken modulo this size. Non-erasing tapes just require more conditions on the function, and are sufficiently characterized by less actions in the gTM definition.

2.6.2. *Tape memory as string.* The single tape memory used in the standard TM could be described as as a symbol chain or string, using an extra symbol not in the input-output-symbol to note the position of the read-write-head. This is often the state number, in which case the set of symbols and states must be distinct.

2.6.3. *Tape memory as two pushdown stores.* Tape memory is a pair of symbol strings, i.e. n-tuples of input symbols, plus the current symbol, the strings constituting the parts left and right to the head.

2.6.4. *Tape memory as fixed mapping from integer numbers to symbols.* The tape contents is a mapping $m : \mathbb{Z} \to S$ from integer numbers to symbols, and the state is the pair (s_j, m_j), see [Boerger1985]. It is presumed that the domain of m_j, i.e. the smallest and largest number for the memory state used so far, needs not be tracked explicitly, but can be efficiently determined from the mapping itself.

2.6.5. *Tape memory as gliding mapping integer numbers.* The tape state is a mapping $m(i, y) : \mathbb{Z} \to Y$ from integer numbers to symbols, where the symbol for zero is the current position of the head. As the tape head is always mapped from zero, there is no need to keep it besides the mapping. However, realisation of a mapping requires a table for each state, and thus doubles the size to save a single index.

2.6.6. *Restricted tapes.* In many cases, tapes are restricted:

- they can only move in one direction
- they do not move at all
- the tapes are read-only, i.e. the initial inscription is never changed
- a symbol may not be erased
- the symbols constitute an ordered graph with respect to writing, (which is a generalisation of a non-erasable tape)
- the tapes are cyclic (if read-only)
- the tapes are infinitely prefilled with patterns

Except the last ones, these conditions have an influence in the TM index because fewer operations are provided.

2.7. **Penalties.** In case the TMs have some quirks that allow fewer states by leaving out features that the others have, e.g. a missing stop state or requiring a cyclic tape, penalties may correct for these cases. However, penalties depend on human judgement and thus are estimates. The following rules are applied in the examples:

If one of the tapes is cyclic and read-only, i.e. has a single infinitely repeated pattern, this saves at least one action and thus reduces the TM index. Maybe it also increases the computational complexity, but this is not taken into account in the TM index. Thus, a penalty of one extra operation to compensate for the saved move in the other direction could be applied.

If the tape is infinite, but not filled with blanks, but with a repeating pattern outside the encoding of the guest machine, this is regarded more complicated than a cyclic tape, and thus an action penalty of two is used.

If the tape is not only infinite, but also filled with a pattern that is not repeating, an action penalty of at least three is used.

If there is no stop state easily added, some extra states may be added in the calculation. The number used should estimate the necessary effort to determine the stop situation. If the stop is a cycle of n states, a state penalty of $n-1$ is used, as a coupled FSM to detect the cylcle would need n states. If the stop is not detected by a defined cycle, the state penalty becomes very large, if not infinite. At least, the number of states plus 1 will be used.

Thus, using the state penalty P_S and the operation penalty P_Ψ, the formula for the estimated TM index becomes:

$$(|S| + P_S) \cdot \sqrt{\frac{|I| \cdot (|\Psi| + P_\Psi)}{2}}$$

3. Invariants

Transformations on standard TMs and their influence on the state symbols product have been studied since Shannon, and as the TM index is the same, these need not to be evaluated here.

So here the emphasis is on changes in the number and characteristics of more than one tape. For these, some machine transformations can be given that do not alter the TM index.

3.1. **Moving state to an extra tape.** Shannon proved that with only one tape, a 1-state standard UTM is not possible.

If a second tape is allowed, a machine with only 1 state is fairly simple, using the technique used by Hooper in appendix II of [Hooper1969]. For a standard Turing machine, just a second (changeable) tape is added, the number of symbols on this tape being equal to the number of states. This second tape is never moved, it just serves to replace the states.

The transformation of any n-state m-symbol 1-tape machine with the common two operations is simple: Just use the pairs of the m symbols and n states as inputs, giving $m \cdot n$ new input symbols, and write the new state to second tape, i.e. use pairs of new state and new symbol as actions, i.e. $m \cdot 2 \cdot n$ actions. The TM index was $n \cdot \sqrt{\frac{m \cdot m \cdot 2}{2}}$ before and is $1 \cdot \sqrt{\frac{(n \cdot m) \cdot (m \cdot 2 \cdot n)}{2}}$ thereafter, thus is equal under this transformation. Hence, using the square root is essential.

3.2. **Many synchronous tapes.** Any n-state, m-symbol, 1-tape TM can be simply transformed to a n-state, 2-symbol, multitape machine, using either unary or binary representation. All tapes are moved the same each time.

If unary representation is used, m tapes are required, the symbols are represented by a bit on one of the m tapes only.

Although m tapes with 2 symbols each span an input space of size 2^m, only m of these are used. Similarily, there are still only $2 \cdot m$ actions, thus the symbol state area size is still $n \cdot m \cdot 2 \cdot m \cdot n$, and the TM index is unchanged.

If binary representation is chosen, k tapes are required such that $2^k > m$ The m symbols are binary coded. As only the used symbols and actions are counted, the TM index is still the same.

3.3. **Splitting (or joining) symbol and move action.** A case that might at the first glance be counter intuitive is the splitting of actions, because one might demand that this operation, that does not change the net behavior, should keep the TM index invariant.

For a standard TM, it is redundant to have a *no move, just change* action, because this action could then be replaced by that of the target state[5]. An engineer might argue technically, that the symbol change and the tape movement have to be done sequentially anyhow, as the tape can only be moved once the symbol write has finished. So he might propose a TM, where the action set does not contain the four members: $\{0R, 0L, 1R, 1L\}$, but instead use the action set $\{0, 1, R, L\}$ i.e. either move or change symbol. Because every state of the original TM would have to be split into two states, there would be twice the number of states and thus the TM index double. The engineer should, however, come to the conclusion that it would normally be less effort to have a three-phase clock instead of a two-phase clock, speeding up the machine by the factor 4/3 and using half the number of memory cells for the state memory. Thus, the increase in the TM index is finally sensible, at least if used as an indicator for the technical effort.

With multitape TMs, the picture is different. For the human it is much easier to understand a state table, when just one tape is moved at a time. Whether e.g. the examples below can be optimized (regarded as gTMs), and if this can be done by a general algorithm, has not yet been explored.

4. Examples

Several TMs are discussed, and the results summarized in Table 1. Some standard TMs are included for comparison.

4.1. **Notational remarks.** State tables are written down using four columns, as indicated above:

- actual state number
- input symbols
- action symbols
- next state number

It is assumed that input symbols are mapped to a single character per tape, which is possible in the cases given. If the machine has more than one tape, the input symbol column thus has two or more characters that represent the symbols on the tapes.

For the action symbols, it depends on the machine if there is a single character used for writing symbol or moving the tape, or two characters for writing a symbols, and moving the tape afterwords. In the latter case, five columns may be used, where the colums for the output symbol and the tape movement could be seen as joined to a single column if seen as an gTM. The hyphen character is used as symbol for *no action.*

In order to increase human readability, dots are used in the tables as follows:

[5] except for the stop state, see examples

In the input column, they denote that the input symbol may be any of those on the tape, and the state table line in fact represents as many lines as there are symbols for the dot. This implies that for each dot, all the symbols to be generated are explicitly used in at least one other line.

In the output column, the meaning of the dot is *same symbol as input.*

E.g., Hasenjaeger's machine could either move or punch on the R-tape, because the Wang instructions are like this, including the option not to move or punch, which is represented by `-`. On Hooper's machine, a pair is given of the new symbol — which may be the same — and the tape move including no move, explicitly denoted by `-`.

Dots in the last column simply mean *keep that state.*

From a mathematical point of view, there is no difference whether a symbol on a tape is replaced by itself (output) or nothing done; from a technical point of view the latter mostly means significantly less circuitry, but it can be easily detected and is noted here by dots just for readability. For counting the numbers to calculate the TM index, dots are resolved to the respective values.

The condition that tapes are presumed to be cyclic is not noted in the state tables directly, as are other restrictions.

Note that for calculating the TM index, it is irrelevant if e.g. the operations are noted with two characters, like `0R`, `0L`, `1R` and `1L` or four different single characters, as long as it is used consistently. We just need to expand the dots and count the number of different input conditions and the number of different actions.

It would have been a nice exercise to transform each non-binary TM to a binary TM and calculate the TM index, optionally doing some optimizations.

Please keep in mind that the issues of computational complexity and the effort of encoding and decoding are not covered by the TM index.

4.2. **A very simple non-UTM.** Just to have a very simple TM, we give a (non-optimal) state table for a non-universal binary TM that appends a mark at the end of a chain of marks:

```
s   in  out  s'
1   M   R    1
1   b   -    2
2   b   M    2
2   M   -    0
```

It has two states, two input symbols `b,M` and three output actions `R,M,-` so the state-symbol product is 4, but the TM index is $2 \cdot \sqrt{3} = 3.46$, because the `L` action is not used. The colour count is 4 instead of 6 for the same reason.

4.3. **Moore's machine.** This machine, published in [Moore1952], was perhaps the first multitape machine with a small number of states with a fully published state table. Also, Moore discussed the conditions to physically build such a machine and found it feasible, the largest problem being the erasable tape, as also observerd by Hasenjaeger, see [Hasenjaeger1987].

The machine has 15 states and 3 tapes, one for the simulated machine, one non-erasable tape for remembering the next state, and one read-only unidirectional cyclic tape for the coded state table. The coding just writes four numbers for each state, using alternately inverted unary numbers[6].

The state table is (in the order used by Moore, without his extensive comments):

```
S PQR PQR S'
1 11. -R- 2
```

[6] $2, 1, 3, 4$ is encoded $1^2, 0^1, 1^3, 0^4$, i.e. `1101110000`

```
 2 1.. R-- 1
 1 10. -L- 3
 3 1.. R-- .
 3 01. -L- 4
 1 01. -L- 4
 4 01. -L- .
 4 00. R-- 5
 5 00. R-- .
 5 10. -R- 1
 1 00. -R- 6
 6 00. R-- 7
 7 10. --0 8
 7 00. R-- 9
 9 01. --1 8
10 10. --L 8
10 00. R-- 11
11 10. --R 8
 8 100 -1- 13
 8 101 -1- 12
12 111 -R- 15
15 .0. -1- 13
13 11. -R- 14
14 10. R-- 15
15 .0. -1- 13
13 01. -L- 4
```

As it has 3 tapes, it has $2^3 = 8$ input symbols, which are covered by the state table. Only 8 actions are used for the tape, as only one tape is operated at a state transition. So the symbol equivalent is $\sqrt{32} = 5.7$, giving a basic TM index of

$$15 \cdot \sqrt{\frac{8 \cdot 8}{2}} = 84.9$$

For the cyclic tape, an action penalty of 1 used. There is no stop state; perhaps because the state number zero cannot be coded. If in the encoding of the state table a state number is used for which not definition exists, the machine will endlessly search that state number on the program tape. According to the rules given above, a state penalty of 14 has to be used, although a smaller solution might be possible. The estimated TM index is thus:

$$(15 + 14) \cdot \sqrt{\frac{8 \cdot (8 + 1)}{2}} = 174$$

4.4. **Hasenjaeger's machines.** Hasenjaeger never published details himself, just some global remarks in [Hasenjaeger1987]; he just did build a physical machine he called the *Mini-Wang.*[7] According to his hand-written notes, he started the design in 1962 and did the schematics in 1963, thus his machine could be dated as of 1964. No evidence is known today when the machine was ready for and used in demonstrations in seminars and lectures in Münster or Bonn.

The binary encoding for Wang's instructions used by Hasenjaeger is:

- 1: mark tape
- 01: move right
- 001: move left

[7]He had build a larger machine of similar design before and called it the *Wang machine*, because Wang in [Wang1957] showed that a universal machine could be non-erasing.

- 00 0^n 1: conditional skip n instructions

Note that each instruction has exactly one marked bit, so the number of ones is the number of instructions to skip.

4.4.1. *The original machine with a bi-directional progamme tape.* The (real) Hasenjaeger machine in its original form for a bi-directional cyclic programme tape[8] and without stop state is:

```
S PQR PQR S'
              I: P=1 is punch, P=0 other instruction
1 1.0 R-M .      mark if not marked, next instruction
1 1.1 R-- .      do not mark if marked, next instruction
1 0.. R-- 2      other instruction, take the 0

             II: R, L or other; Q is zero on entry
2 10. R-R 1      go right, next instruction
2 00. RR- .      save 0 in Q, check next P bit
2 11. RLL 1      next P bit is 1, go left, clear Q, next inst.
2 01. RL- 3      next P bit is 0, this is a skip

            III: skip part 1: count zeroes to Q, if mark
3 0.0 R-- .      R has space, skip zeroes until P=1
3 1.0 R-- 1      R has space, end found: next instruction
3 0.1 RR- .      R has mark, count zeroes until P=1
3 1.1 LR- 4      R has mark, end found: exec skip

             IV: skip part 2: execute
4 01. L-- .      while Q>0, skip zeroes on P, leave Q
4 11. LL- .      while Q>0, skip a one, decrement Q
4 .0. R-- 1      Q=0, next instruction (mark prepended)
```

It has four distinct states and three binary tapes, `P`, `Q` and `R`. Tape `P` is the programme tape with a defined initial inscription encoded as above, (using relative backward jumps only and a cyclic tape to achieve forward jumps), and it is unchangeable. Tape `Q` is also unchangeable and has just a single mark, which is at the tape head position when the machine starts. Thus, tape `Q` can be used as a counter. Tape `R` is the working tape[9] for the simulated (guest) machine and not erasable.

As there are three binary tapes, so it has $2^3 = 8$ input combinations, which are all used and defined. The actions are:

- `-`, `R` and `L` for tape `P` and tape `Q`
- plus `M` for tape `R`.

Thus, the maximum number of possible output actions is $3*3*4 = 36$, but only 9 of these are used, e.g. tape `R` is only modified in three lines of the state table.

Thus, the symbol equivalent is $\sqrt{\frac{8\cdot 9}{2}} = \sqrt{36} = 6.0$ (incidently $3 \cdot 2$ for 3 tapes with 2 symbols), and the basic TM index is:

$$4 \cdot \sqrt{\frac{8 \cdot 9}{2}} = 24.0$$

ignoring that there is no stop and that tape `P` is necessarily cyclic to allow conditional transfers to any instruction on tape `P`.

[8]build only recently from a bidirectional *uniselector* found in his legacy

[9]*Rechenband* in German, means *calculating tape*

To replace tape `P` by a not-cyclic one is not trivial, thus a penalty of one action is used, giving an estimated TM index of:

$$4 \cdot \sqrt{\frac{8 \cdot (10+1)}{2}} = 26.5$$

As regards the missing stop state, Hasenjaeger in his private notes refers to what he calls a *dynamic stop*, i.e. using a conditional `skip 0`, i.e. jump back just to the same instruction. This means that tape `R` does not act, and tape `P` stays within a small section of the tape, and the relay pattern repeats each 10 clocks, which is not easy to detect. Using the above penalty rules, that would mean a state penalty of 3, giving an estimated TM index of:

$$(4+3) \cdot \sqrt{\frac{8 \cdot (9+1)}{2}} = 44.27$$

It is, however, easy to detect a `skip 0` by splitting the last line of state 3 as follows:

```
3 111 LR- 4     end found; R has mark, need to skip
3 101 --- 0     skip 0 is stop
```

in which case no state penalty is necessary. However, the current hardware can sense tape `R` only in state 1 and 3, and tape `Q` only in state 2 and 4, so this requires a changed hardware. Another option could be to treat marking a marked state as a stop. But the conditional backward jump spaces one too far and lands on the mark that terminates an instruction, which has no effect, as tape `R` is on a mark anyhow, but requires that the mark on a marked tape does not stop.

Thus, excluding the engineering aspect, only the penalty for the cyclic tape `P` is needed, giving an estimated TM index of 26.5.

4.4.2. *The modified machine with a uni-directional progamme tape.* When the machine was found, there was no documentation available and a unidirectional programme tape enclosed, which did not fit to the state table. Looking for a way to have this configuration running, using forward jumps seemed to be the solution, requiring only a small change in the state table, namely inverting a relay contact:

```
S PQR PQR S'
             I: P=1 is punch, P=0 other instruction
1 1.0 R-M .     mark if not marked, next instruction
1 1.1 R-- .     no need to mark if marked, next instruction
1 0.. R-- 2     other instruction, take the 0

            II: R, L or other; Q is zero on entry
2 10. R-R 1     go right, next instruction
2 00. RR- .     save 0 in Q, check next P bit
2 11. RLL 1     next P bit is 1, go left, clear Q, next inst.
2 01. RL- 3     next P bit is 0, this is a skip

           III: skip part 1: count zeroes to Q, if mark
3 0.0 R-- .     R has space, skip zeroes until P=1
3 1.0 R-- 1     R has space, end found: next instruction
3 0.1 RR- .     R has mark, count zeroes until P=1
3 1.1 RR- 4     R has mark, end found: need to skip

            IV: skip part 2: execute
4 01. R-- .     while Q>0, skip zeroes on P, leave Q
4 11. RL- .     while Q>0, skip a one, decrement Q
```

```
        4 .0. --- 1     Q=0, next instruction
```

Just the direction of skipping is changed from backwards to forwards, and using the cyclicity of the tape for backward jumps. However, this slows down the execution, as actions like skipping over a string of marks each time cycle the P-tape, making demonstrations nearly incomprehensible. Also, the *dynamic stop* is hard to see, but could be done as before, though not in the physical machine:

```
        3 111 RR- 4     R has mark, end found: need to skip
        3 101 --- 0     R has mark, end found, stop
```

Because tape `P` has one action less, the machine has two less, thus the basic TM index is

$$4 \cdot \sqrt{\frac{6 \cdot 7}{2}} = 21.2$$

Adding the penalty for the cyclicity of tape `P`, gives an estimated TM index of:

$$4 \cdot \sqrt{\frac{8 \cdot (7+1)}{2}} = 22.6$$

4.4.3. *The compact machine.* As I observed only recently, in state 2 of the above machine tables, tape `Q` is used as auxiliary state memory (similar to Hooper). Taking this further on, one state may be saved:

```
S PQR PQR S'
           I: P=1 is punch, Q is zero on entry
1 100 R-M .     mark while not marked, next instruction
1 101 R-- 0     if already marked, stop
1 00. RR- .     save 0 in Q, change to alternate state
1 11. RLR .     was 01: R, next instruction
1 01. RL- 2     consumed two zeros, continue decoding

          II: L or Jump; Q is zero on entry
2 10. R-L 1     was 001: L, next instruction
2 00. RR- .     bias Q, change to alternate state
2 010 R-- .     R has space, no jump, skip zeroes
2 110 RL- 1     R has space, no jump, done
2 011 RR- .     R has mark, jump, count number of zeros
2 111 L-- 3     end of jump instruction, execute jump

         III: execute jump
3 01. L-- .     skip zeros backwards
3 11. LL- .     skip back and count instruction
3 00. R-- .     Q zero, before end of next instruction
3 10. R-- 1     Q zero, continue with next instruction
```

It uses tape `P` bidirectional, and there are 8 input symbols and 8 output actions, giving a symbol equivalent of 5.7. With with only 3 states the basic TM index is:

$$3 \cdot \sqrt{\frac{8 \cdot 8}{2}} = 17.0$$

This machine stops if an already marked place has to be marked, which is possible here. Although this increases the effort for some transformations from standard TMs to Wang-Hasenjaeger TMs, here the goal is to have a low TM index. Just one action penalty for the cyclic tape gives an estimated TM index of:

$$3 \cdot \sqrt{\frac{8 \cdot (8+1)}{2}} = 18.0$$

If a unidirectional tape `P` is used, this saves two actions:

```
S PQR PQR S'
             I: P=1 is punch, Q is zero on entry
1 100 R-M .     mark while not marked, next instruction
1 101 R-- 0     if already marked, stop
1 00. RR- .     save 0 in Q, change to alternate state
1 11. RLR .     was 01: R, next instruction
1 01. RL- 2     consumed two zeros, continue decoding

            II: L or Jump; Q is zero on entry
2 10. R-L 1     was 001: L, next instruction
2 00. RR- .     bias Q, change to alternate state
2 010 R-- .     R has space, no jump, skip zeroes
2 110 RL- 1     R has space, no jump, done
2 011 RR- .     R has mark: count number of zeros
2 111 R-- 3     end of jump instruction, execute jump

           III: execute jump
3 01. R-- .     skip zeros
3 11. RL- .     skip mark and count instruction
3 .0. R-- 1     Q zero, continue with next instruction
```

Thus, the basic TM index is

$$3 \cdot \sqrt{\frac{8 \cdot 6}{2}} = 14.7$$

For the cyclic tape, there is still an action penalty of `1` to be added:

$$3 \cdot \sqrt{\frac{8 \cdot (6+1)}{2}} = 15.87$$

The computational complexity in time and space is not significantly changed, as all forward jumps are shorter by a constant and all backward jumps are longer by a constant.

However, this machine is again not compatible to the physical one, as the mutually exclusive reading of tapes `Q` and `R` is no longer satisfied. The now free fourth state (as we have to use two flipflops anyhow) can be used to implement a stop if the jump distance is zero, which here is a no-operation.

4.5. **Hooper's machines.**

Hooper published in [Hooper1969] two multi-tape machines, a binary one with one state, four tapes and two symbols, that uses Wang's instructions (like Hasenjaeger with relative jumps and a cyclic tape); and another one with two tapes, two states and three symbols, that uses cyclic tag systems.

4.5.1. *Hooper's 1-state Wang style UTM.*

This machine is remarkable in several ways: It uses Wang's instructions, and it uses the two additional tapes not only to count the skip for Wang's conditional transfer, but also to replace states by tape contents.

The encoding used is:

- 01: mark tape
- 11: move right
- 10: move left
- $(00)^n$: conditional skip n one-bits

The table reads in common format:

S	PTRS	PTRS	PTRS	S’	
1	1.10	..0.	L-LL	1	(1, 2)
1	0.01	..1.	LL-R	1	(3, 4)
1	1.01	..1.	LR-R	1	(5, 6)
1	0010	....	L-R-	1	(7)
1	1000	.1..	L-L-	1	(8)
1	0110	...1	L-RR	1	(9)
1	0100	..1.	L---	1	(10)
1	0111	....	L---	1	(11)
1	1111	..00	L-LL	1	(12)
1	1100	....	--LL	1	(13)
1	0000	....	L-L-	1	(14)
1	.011	....	----	0	(see p. 215)

`P` is the progamme tape, `T` the target tape, and `R` and `S` are auxiliary tapes with a predefined inscription. There are $2^4 = 16$ possible inputs, that are all used (after the stop condition added that is mentioned in the text). The number of possible actions is very high (16 output symbols, and 3 tape actions each, including the neutral one, i.e. $2^4 \cdot 3^4 = 1296$). But as only 13 actions are used, the basic TM index is very low:

$$1 \cdot \sqrt{\frac{16 * 13}{2}} = 10.2$$

Applying a penalty of one action for the cyclic tape `P`, and one for the predefined inscription of tape `R` and `S` increases the estimated TM index slightly to

$$1 \cdot \sqrt{\frac{16 * 15}{2}} = 10.95$$

4.5.2. *Hooper’s 2-state UTM.* The machine has 2 states, 2 tapes, and 3 symbols (including blank) on each, and uses a tag system to encode the guest TM.

The state table is in compact form (stop state added as mentioned in the text):

S	TU	TU	op	S’	
1	00	..	R-	.	
1	10	bb	RL	.	
1	11	.b	LL	2	
1	01	.0	LL	2	
1	b1	1.	L-	.	
1	0b	..	L-	.	
1	bb	1.	L-	.	
1	1b	0.	LR	2	
1	b0	..	--	0	(see p. 211)
2	b1	.0	-L	.	
2	b0	.1	-L	.	
2	bb	11	L-	1	
2	0b	.1	LL	.	
2	10	.1	-R	.	
2	11	.0	-R	.	
2	1b	..	RL	.	
2	00	.b	-L	.	
2	01	1b	RL	1	

All 9 possible inputs are used (stop state added). Out of the $3^2 \cdot 3^2 = 81$ possible output actions, only 15 are used, giving a TM index of

$$2 \cdot \sqrt{\frac{9 \cdot 15}{2}} = 16.4$$

4.6. **Rogozhin's machines.** Rogozhin published in [Rogozhin1996] seven universal standard TMs using tag systems for encoding, of which the one with the lowest state-symbol product of 24 and the one using a binary tape are shown. Both machines have a stop already incorporated into the state table, and use standard tapes which are blank outside the encoded part.

4.6.1. *The 4-state 6-symbol machine.* The machine has 4 states and uses 6 symbols, giving a state-symbol product of 24. The symbols with the arrows above the characters are replaced by more common characters (`e` for east and `w` for west) in the following state table:

S	I	O	S'
1	1	wL	.
1	b	eR	.
1	e	bL	.
1	w	0R	.
1	0	wL	.
1	c	0R	4
2	1	0R	.
2	b	eL	3
2	e	wR	.
2	w	eL	.
2	0	1L	.
2	c	bR	.
3	1	1R	.
3	b	wR	4
3	e	bR	.
3	e	--	0
3	0	cR	1
3	c	1R	1
4	1	0R	.
4	b	cL	2
4	e	wR	.
4	w	--	0
4	0	cL	2
4	c	bR	.

Out of maximal $6 * 2 = 12$ actions, just 11 are used (unused is `0L`), so the TM index is slightly less the state symbol product:

$$4 \cdot \sqrt{\frac{6 \cdot 11}{2}} = 23.98$$

4.6.2. *The 24-state 2-symbol machine.* The machine has 24 states and 2 symbols according to the following state table:

S	I	O	S’
1	0	.R	5
1	1	.R	2
2	0	1R	1
2	1	.L	3
3	0	.L	4
3	1	0L	2
4	0	1L	12
4	1	0L	9
5	0	1R	1
5	1	0L	6
6	0	.L	7
6	1	.L	7
7	0	.L	8
7	1	0L	6
8	0	.L	7
8	1	1R	2
9	0	.R	19
9	1	.L	4
10	0	1L	4
10	1	0R	13
11	0	.L	4
11	1	--	0
12	0	.R	19
12	1	.L	14
13	0	.R	10
13	1	.R	24
14	0	.L	15
14	1	.L	11
15	0	.R	16
15	1	.R	17
16	0	.R	15
16	1	.R	10
17	0	.R	16
17	1	.R	21
18	0	.R	19
18	1	.R	20
19	0	1L	3
19	1	.R	18
20	0	1R	18
20	1	0R	18
21	0	.R	22
21	1	.R	23
22	0	1L	10
22	1	.R	21
23	0	1R	21
23	1	0R	21
24	0	.R	13
24	1	0L	3

As all input symbols and output actions are used, so the TM index is equal to the state-symbol product:

$$24 \cdot \sqrt{\frac{2 \cdot 4}{2}} = 48$$

4.7. **Wolfram's machine.** Wolfram published 2002 in [Wolfram2002] a very small machine with only 2 states and 3 symbols, having a state symbol product of 6. A proof for its universality was announced 2007 to be published in *Complex Systems.* Whether it is really universal is not of full importance here, as it just serves to illustrate the TM index.

The state table is:

S	I	O	S'
1	0	1R	2
1	1	2L	.
1	2	1L	.
2	0	2L	1
2	1	2R	.
2	2	0R	1

All 3 input symbols are used, but only 5 output actions are used (the `2L` is duplicate and `0L` not used), thus the basic TM index is

$$2 \cdot \sqrt{\frac{3 \cdot 5}{2}} = 5.48$$

As it seems that the machine does not only not stop, but infinitely write to the tape, according to the above rule, a state penalty of 3 is used, which seems to be very generous here. As the initial inscription of the tape is said to be infinite non-repeating, an action penalty of three is used, resulting in an estimated TM index of:

$$(2 + 3) \cdot \sqrt{\frac{3 \cdot (5 + 3)}{2}} = 17.32$$

The question of encoding as well as decoding as well as the computational complexity are at least difficult to determine for a machine that never stops, but is, as always here, not included in the above estimated TM index.

4.8. **Woods' and Neary's machines.** Woods and Neary have published a fairly large number of machines, mostly for cyclic tag systems and often what they called *weakly universal*, where the tape is infinitely prescribed.

4.8.1. *The 4 state 5 symbol machine.* Published in [WoodsN2009], it is a standard TM with 5 symbols and 4 states, thus the state-symbol product is 20. The state table (λ replaced by %):

S	I	O	S'	
1	0	μL	.	
1	1	BR	2	
1	μ	0R	2	
1	B	1R	0	(see p. 171)
1	%	0L	2	
2	0	μL	.	
2	1	1L	.	
2	μ	0R	1	
2	B	0L	.	
2	%	μL	3	

3	0	BR	.
3	1	1R	.
3	μ	0R	4
3	B	0L	2
4	0	BR	.
4	1	1R	.
4	μ	0R	3
4	B	1L	2

All 5 input symbols are used, but out of the 10 possible actions, only 6 are used, because the delimiter `%` is never written to the tape. Thus, the basic TM index is smaller than the state-symbol product:

$$4 \cdot \sqrt{\frac{5 \cdot 6}{2}} = 15.49$$

The encoding fills the tape to the left with a repeating pattern, but not to the right, thus an action penalty of two is used for the estimatd TM index:

$$4 \cdot \sqrt{\frac{5 \cdot 8}{2}} = 17.89$$

4.8.2. *The 2 state 4 symbol machine.* Published in [NearyW2009] as $U_{2,4}$, it has 2 states and 4 symbols and uses *Rule 110* encoding. It is characterized as *weakly universal* because a predefined pattern is repeatedly duplicated on the tape outside the encodeded part.

Its state table is:

S	I	O	S'
1	0	∅L	.
1	1	lL	2
1	∅	lL	.
1	l	lL	.
2	0	lR	1
2	1	∅L	.
2	∅	0R	.
2	l	1R	.

All 4 input symbols are used, but out of the 8 possible actions, only 5 are used (`0L`, `1L` and `∅R` are not used), so the basic TM index is:

$$2 \cdot \sqrt{\frac{4 \cdot 5}{2}} = 6.32$$

The predefinded repeated pattern is similar to a cyclic tape, but slightly more complicated, so 2 action penalties might be justified, giving an estimate TM index of:

$$2 \cdot \sqrt{\frac{4 \cdot (5 + 2)}{2}} = 7.48$$

5. Overview of TM indices and colours

Numbers are given in Table 1, some shown graphically in Fig. 1. The y-axis is logarithmic as to better distinguish the points at small index numbers.

The abbreviations used are:

Author	Year	Tp	St	Sy	SSP	In	Ops	SyE	TMi	eTMi	Mc	HN	Enc
Moore	1952	3	15	2	30	8	8	5.7	84.9	174	28	120	Moore
Hasenjaeger	(1964)	3	4	2	8	8	9	6.0	24.0	26.5	29	32	Wang
Hooper	1969	4	1	2	2	16	13	10.2	10.2	10.9	14	16	Wang
Hooper	1969	2	2	3	6	9	15	8.2	16.4		17	18	cytag
Rogozhin	1996	1	4	6	24	6	11	5.7	23.9		19	24	cytag
Rogozhin	1996	1	24	2	48	2	4	2.0	48.0		8	48	cytag
Wolfram	2007	1	2	3	6	3	5	2.7	5.5	17.3	6	6	r110
Neary&Woods	2009	1	4	5	20	5	6	3.9	15.5	17.9	11	20	cytag
Neary&Woods	2009	1	2	4	8	4	5	3.2	6.3	7.5	8	8	r110
Hasenjaeger*	2012	3	3	2	6	8	6	5.7	14.7	15.9	21	18	Wang

TABLE 1. Summary of Numbers

Tp:	tapes
St:	states
Sy:	number of symbols
SSP:	state symbol product
SyE:	symbol equivalent
TMi:	basic TM index
Mc:	union of Margenstern's colours
eTMi:	estimated TM index (if larger)
Enc:	encoding for programme
cytag:	cyclic tags
r110:	rule 110

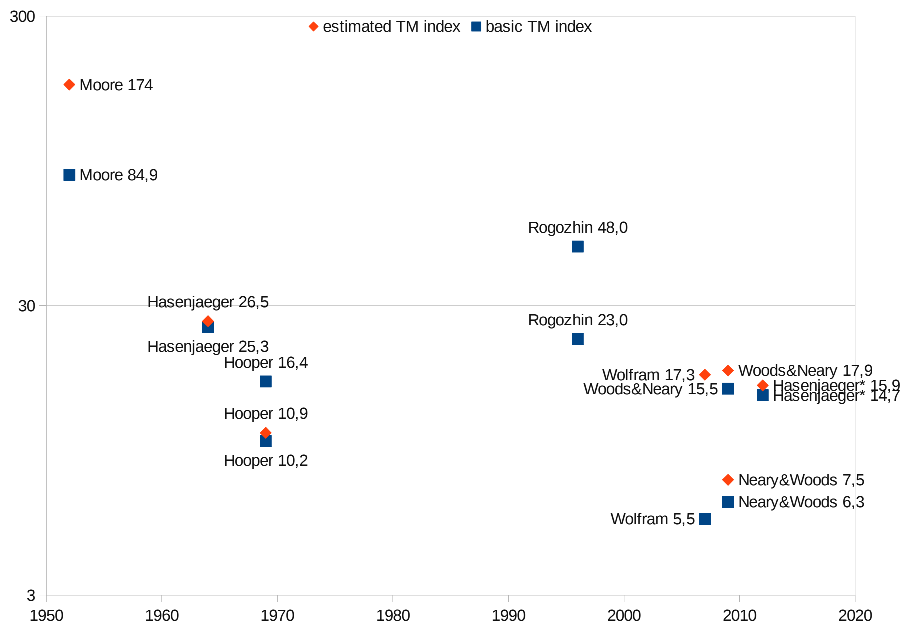

FIGURE 1. TM Index vs. Year

References

[Boerger1985]	Egon Börger: *Berechenbarkeit — Komplexität — Logik.* Vieweg Braunschweig 1985. In English: *Computability, Complexity, Logic.* Studies in Logic and the Foundations of Mathematics, Vol. 128, North-Holland, 1989.
[Cooper2003]	S. Barry Cooper: *Computability Theory.* John Wiley & Sons, 2008.
[Fischer1965a]	Patrick C. Fischer: *On Formalisms for Turing Machines.* Journal of the ACM 12 (4) p.570-580, 1965.
[Hasenjaeger1987]	Gisbert Hasenjaeger: *On the early history of register machines.* LNCS 270 p.181-188, Springer, 1987.
[HartmanisS1965]	J. Hartmanis, R. E. Stearns: *On the Computational Complexity of Algorithms.* Tran. Am. Math. Soc. 117, p. 285-306, 1965.
[Hooper1969]	Philip K. Hooper: *Some small, multitape universal Turing machines.* Information Sciences 1 (2) p.205-215, 1969
[JiangSV1997]	Tao Jiang, Joel I. Seiferas, Paul M. B. Vitányi: *Two heads are better than two tapes.* J. ACM 44 (2), p.237-256, 1997.
[Margenstern1993]	Maurice Margenstern: *Non-erasing turing machines: A (new) frontier between a decidable halting problem and universality.* LNCS 710, p. 375-385, 1993.
[Moore1952]	E. F. Moore: *A simplified universal Turing machine.* Proceedings of the 1952 ACM national meeting (Toronto) p. 50-54, 1952.
[NearyW2009]	Turlough Neary, Damien Woods: *Four Small Universal Turing Machines.* Fundam. Inform, 91(1), pp. 179-195, 2009.
[Priese1979a]	Lutz Priese: *Über eine minimale universelle Turing-Maschine.* LNCS 67, p. 244-259, 1979.
[Priese1979b]	Lutz Priese: *Towards a Precise Characterization of the Complexity of Universal and Nonuniversal Turing Machines.* SIAM Journal on Computing, 8(4), pp. 508-523, 1979.
[Rogozhin1996]	Yurii Rogozhin: *Small universal Turing machines.* Theoretical Computer Science, 168(2), pp. 215-240, 1996.
[Shannon1956]	Claude E. Shannon: *A Universal Turing Machine with Two Internal States.* Automata Studies, Princeton University Press, 1956.
[Turing1936]	Alan M. Turing: *On Computable Numbers, with an Application to the Entscheidungsproblem.* Procedings of the London Mathematical Society, 42(2), pp. 230-265, 1936.
[Wang1957]	Hao Wang: *A Variant to Turing's Theory of Computing Machines.* J. ACM, 4(1), pp. 63-92, 1957.
[Wolfram2002]	Stephen Wolfram: *A New Kind of Science.* Wolfram Media, Champaign, 2002.
[WoodsN2009]	Damien Woods, Turlough Neary: *The complexity of small universal Turing machines: A survey.* Theoretical Computer Science 410 (4-5) p. 443-450", 2009.

MENTROPSTR. 84, 33106 PADERBORN, GERMANY
E-mail address: `rainer@glaschick-pb.de`