Mathematica Aeterna, Vol. 3, 2013, no. 9, 709 - 738

Methods for Understanding Turing Machine Computations

John Nixon

Brook Cottage
The Forge, Ashburnham
Battle, East Sussex
TN33 9PH, U.K.

Abstract

This paper is an *ab initio* investigation into ways of describing Turing Machine (TM) behaviour. It is shown how results for a TM restricted to a finite length of tape can be used to speed up any TM computation. The non-redundant set of such rules is referred to as the irreducible regular (computation) rules (IRR) and an algorithm is described to generate them for any TM for an arbitrary length of tape. This algorithm has been implemented in C++ as is freely available. The examples studied show that the IRR can be either finite or infinite in number. For several TM's when they are infinite, recursive formulae for them were found and it is expected that this is always possible. These formulae were found by examining the IRR in each example separately and correctly guessing it and proving it by induction. A table showing which IRR follow others dependent on the next symbol read, was found for the examples studied and gives much information on the TM behaviour. It is anticipated that it will be possible in this way to analyse a universal TM to discover how the interpretation cycle works.

Mathematics Subject Classification: 68Q05

Keywords: Turing Machine, Finite tape, short-cut computation rules

1 Introduction

The Turing Machine (TM) is one of the most well known models of computation and is arguably the simplest. The importance of TMs lies in the well known fact that the TM can be programmed to carry out any computation. See for example Minsky's book [1] that initially motivated this study. The general working of a TM is described below and the program that defines a particular example will for simplicity be referred to as a Turing Machine. To define a computation, a particular TM needs to be specified together with the initial state of its "tape", which can be thought of as an linear infinite (in both directions) collection of cells that each can contain one of a finite collection of symbols at any point in time. As defined thus the TM would be a general kind of simulation of a one-dimensional universe that once started could continue indefinitely, but this would not reach an "answer" unless a predefined condition was reached when some aspect of the resulting information on the tape would give the "answer". The simplest way to do this is to declare one of the machine states to be a halt state that when entered immediately stops the TM. The methods in the paper can however be applied to TMs whether or not they have halt states. I think this fact makes my approach different from most other work in this area. I regard the understanding of the continuing computation in terms of the application of short-cut rules an important first step in analysing a computation which is equally or more important than the actual result on halting which can be derived from it.

The generality of the TM concept made a succession of more complex examples fascinating to study and the methods of this paper resulted from a succession of improvements of technique and concepts resulting from these studies. Many times I have given up on a particular approach or example because of the enormous complexity and unsatisfying incompleteness of the results obtained. The resulting methods now form a coherent set of ideas and I expect the uses of these methods will be many and varied and I am publishing this in the hope that my readers will develop the methods here and find many uses for them.

In Section 2, Turing Machines are defined, followed by Configuration Sets (CSs) in Section 3. CSs are the building blocks in terms of which computation rules are defined in Section 4. These are the short-cuts that can be used when carrying out the computation of a particular TM. In Section 5, a class of computation rules (or regular rules RR) is defined recursively by restricting the computation to a finite length of tape, and it is shown how any computation of a TM can be expressed in terms of these. The set of RR so generated has redundancy because rules derived in one step from another rule should not be mentioned separately. Removing these gives the set of Irreducible Regular Rules (IRR) and a definition of the completeness of a set of IRR is given.

The first three examples of TM's were taken from a study of TM's by Machlin and Stout [3] who referred to them as "a simple loop", "a shadow christmas tree" and a "a counter" respectively and it may be instructive to compare my analysis with theirs. In Sections 6 and 7 examples of Turing Machines with a finite and infinite number of IRR respectively are analysed including giving tables which establish the completeness of the set of IRR found and give information on the long-term behaviour of the TM in each case. In Section 8 an example is studied that for the first time has a number of derivation steps in any IRR that is approximately exponentially increasing with the length of the tape. This indicated very complex behaviour and long computation times within short lengths of tape that are effectively summarised by the IRR and is related to the designation "counter".

In Section 9 it is shown that the IRR in general have a tree structure. Theorems are given that allow recursive construction of the IRR and a general algorithm applicable to any TM is given to generate them for an arbitrary length of tape. Another example of a TM is studied in detail that required a much more complex induction proof for the set of IRR. The algorithm to generate the IRR was used to generate the IRR for the earlier examples which suggested their general formulae which were then verified by induction as in the earlier sections of the paper. Section 10 has some concluding remarks.

2 Definition of Turing Machines (TMs)

There are many slightly different but equivalent versions and notations for TM's. A TM here is defined to interact with a one-dimensional tape that is potentially infinite in both directions and may be thought of as marked off into squares, each having one symbol on it at any point in time. The interaction is by reading, writing (i.e. erasing the previous symbol written in that square) to one square of the tape at a time at the position where the read/write head is at that time, and moving one step to the right or left.

The machine table that completely describes a particular TM can be specified by a set of quintuples. This is also the notation I used to represent TM's for input to the computer program [2] for TM analysis. In each quintuple, the first two elements are respectively the machine state and the symbol that has been read. These pairs uniquely determine three corresponding actions to be taken by the TM in one cycle i.e. (1) print a new symbol at the current position of the read/write head (pointer), (2) move the pointer left (L) or right(R) by one square, (3) enter a new machine state. These actions are carried out in this order and are represented respectively by the remaining three symbols of the quintuple. Halting condition(s) may be included by having 'H' as the new state reached for a subset of the quintuples. The TM continues to repeat the basic cycle indefinitely unless a halting condition is reached.

3 Configuration Sets

A configuration of a TM is the complete description of the state of the computation at one instant i.e. the contents of the tape, the machine state, and the position of the pointer on the tape. In the following analysis, extensive use will be made of configurations sets (CSs), which are sets of configurations defined by specifying the machine state and a few symbols on the tape relative to the position of the pointer. The notation for CSs is as follows: the machine state, then the specified symbols on the tape in the same order, with the pointer indicated by an underscore or an underline. I will use capital letters for machine states and non-negative integers for symbols on the tape.

In this paper, CSs will determine the symbols in a single range on the tape without any gaps. A subset of a CS can be formed by specifying extra symbols for it. For example, $A0\_$ that could be read is the CS defined by state A and symbol 0 immediately to the left of the pointer and is a superset of $A10\_$. A superscript indicates a repeat count for example $B\underline{0}2^31$ is $B\underline{0}2221$ and $C\{01^23\}^3300\_$ is $C011301130113300\_$. All parameter values are non-negative integers unless otherwise stated. This convention saves specifying conditions explicitly in each formula.

4 Computation Rules

A computation rule (abbreviated to 'rule') for a TM has a pair of CSs that are related by a set of TM steps, such that every TM configuration in the left hand side (LHS) of the rule is mapped by the TM steps to a configuration in the right hand side (RHS) of the rule, and the RHS is the exact image of the LHS under the mapping. Using such derived rules for computing with a TM speeds up the computation (in terms of substitution steps) by a factor which is a weighted average of the number of TM steps needed to derive a rule. The length of a rule is the number of symbols in the neighbourhood of the pointer that are specified in its LHS and RHS, which must be the same. The instructions of the TM (the quintuples) are the rules of length 1 and can be expressed in the form $q\underline{s_1} \to q'\_s'_1$ or $q\underline{s_1} \to q's'_1\_$ for a left or right moving machine step respectively, where an underline or underscore represents the pointer position. The s's represent symbols on the tape, and q's represent the machine states.

For the regular rules (RR) to be defined recursively below of length n, the RHS is the result of computing on a piece of tape of length n with specified symbols for as many TM steps as possible unless the 'C' (or cycle) condition occurs. The 'C' condition means that the computation enters a stationary cycle of repeating configurations. This is in contrast to an endless repeating cycle that progresses along the tape when for example the tape has all 1's at the pointer and to the right of it and the instruction $A\underline{1} \to A0\_$ repeats indefinitely replacing the 1's by 0's as the pointer moves right. This is not a 'C' condition.

Useful computation rules frequently fall into the following 8 forms, the first

4 of which represent cases where the computation does not halt or cycle:

$$q\underline{s_1}s_2\dots s_n \rightarrow q'\_s_1's_2'\dots s_n'$$

$$q\underline{s_1}s_2\dots s_n \rightarrow q's_1's_2'\dots s_n'\_$$

$$qs_1s_2\dots \underline{s_n} \rightarrow q'\_s_1's_2'\dots s_n'$$

$$qs_1s_2\dots \underline{s_n} \rightarrow q's_1's_2'\dots s_n'\_$$

which are defined to have types LL, LR, RL, RR respectively, and the other 4 forms are defined to have the types LH, LC, RH, RC where the first symbol (L or R) of the 'type' represents the pointer being at the left or right respectively in the LHS, and second symbol of the 'type' (H or C) represents the computation ending in a halt state H or a static endless cycle (condition 'C') respectively. The RHS of the regular rules of types LH and RH is the final halting CS reached, and the RHS of regular rules of types LC and RC is one of the CSs that is repeated in the computation.

5 Regular rules: representing TM computations in terms of computation rules

The following algorithm generates a subset of the set of all rules of length n for any TM, which is defined to be the set of all regular rules (RR) of length n for the TM. Note that unfortunately 'RR' could stand for 'regular rule' or one of the 'types' of a rule but there should be no confusion because of the context.

Algorithm 5.1. *The regular rules of length 1 are simply the instructions of the TM. Add each symbol in turn at the pointer position in the RHS of each regular rule of length n with one of the following types: LL, LR, RL, RR that has a distinct RHS, to generate the LHS of a new regular rule of length* $n + 1$. *For each of these, run the TM until either (1) a halting state H is reached, or (2) a reentered CS is reached indicating and endless static cycle, or (3) the pointer reaches just past one of the ends of the part of the tape with specified symbols. In cases (1) and (3) the RHS of the regular rule is the final CS reached, and in case (2) there is not really a RHS, but any RHS in the repeating loop could be used. The 'type' C indicates this exceptional circumstance so this will not be a problem. Repeat the whole procedure starting with* $n = 1$ *and adding 1 to n in each cycle until the desired value of n is reached.*

It is easily verified that the regular rules generated by this algorithm have one of the 8 types defined above. The importance of this algorithm is the following obvious theorem:

Theorem 5.2. *Every computation of a TM that does not halt or cycle, using only a portion of the tape of length n and continuing until the pointer reaches just past where the specified symbols are given and passing every square in that portion, can be represented by a unique sequence of n applications of the regular rules from Algorithm 5.1, where the ith regular rule in the sequence has length i. The application of each regular rule involves reading a symbol on the part of the tape not yet passed by the pointer. If the computation halts, the last regular rule applied is a halting regular rule i.e. the last regular rule's type has H as its second symbol. Likewise if the computation goes in an endless static cycle, the repetition is within the last applied regular rule (having type LC or RC).*

Proof. Divide the sequence of computation steps into a sequence of subsequences of steps, such that each subsequence ends and a new one starts when the pointer moves to each position on the tape that has not been passed before. The first subsequence of TM steps is the single TM step which is first to be executed in the computation (length 1). The ith subsequence of TM steps operates on a string of symbols on the TM's tape of length i though not all these symbols in this string have to be read during its execution. The effect of this subsequence of TM steps is a RR of length i of the type generated by algorithm 5.1 because the LHS of this RR has one symbol added at the pointer to a RHS of an RR of length $i-1$, as in Algorithm 5.1. The remaining statements of the theorem are now obvious. □

A faster but equivalent algorithm to Algorithm 5.1 can be obtained by making maximum use of the previously found regular rules instead of running the TM from scratch each time.

Algorithm 5.3. *The regular rules of length 1 are simply the instructions of the TM. Add each symbol in turn at the pointer position in the RHS of each regular rule of length n of type LL, LR, RL, or RR that has a unique RHS to generate the LHS of a new regular rule of length $n+1$. For each of these LHS's, repeatedly match it with the left hand side of one of the regular rules of length $n-1$ and use that regular rule to make the substitution it indicates. Do this until either (1) a reentered CS is reached among the results of the substitution steps indicating and endless static cycle, or (2) a regular rule being applied itself ended in a halt state H or a static endless cycle i.e. the applied regular rule has type LH, LC, RH, or RC, or (3) the pointer reaches just past one of the ends of the part of the tape with specified symbols. Repeat the whole procedure from $n = 1$ adding 1 to n in each cycle until the desired value of n is reached.*

The validity of this algorithm and its equivalence with Algorithm 5.1 is the essence of the next theorem. It is one of the most important results of the paper, and the proof has a lot of tedious detail but there is nothing difficult there. First I need to set up some notation. For each $k \geq 1$, let L_k and R'_k be the sets of all LHS's and RHS's of regular rules of length k respectively for the TM, except that in the stationary cycling case an element of R'_k will not be assigned. For a regular rule of length k, the RHS is (1) empty in the static cycling case 'C' or (2) is a halting CS i.e. a CS having state 'H' or (3) is a CS having the pointer immediately to the right or left of the string of known symbols. I define H_k and R_k to be the subsets of R'_k corresponding to cases (2) and (3) respectively, so that $R'_k = H_k \cup R_k$. By definition, each member of L_{k+1} is obtained by adding an extra symbol at the pointer to a member of R_k.

Theorem 5.4. *For each LHS of a regular rule of length* $n+1$, *and for any stage in the sequence of substitutions starting from it, one of the regular rules of length* n *will match until the algorithm is terminated by one of the conditions (1), (2) or (3) above. The result of algorithm 5.3 is the same as the result of Algorithm 5.1 for any TM.*

Each of the regular rules obtained from Algorithm 5.3 is of one of the 8 types mentioned above i.e. LL, LR, RL, RR, LH, LC, RH, RC. The derivation has an alternating sequence of regular rules of types LR, and RL (that can have zero or more members) terminated by a regular rule of any of the types LL or RR, LC, LH, RC, RH. The second symbol of the type of a regular rule must match the first symbol of the type of the following regular rule in the derivation, so for example a regular rule of type LL can only follow a regular rule of type RL, and a regular rule of type RR can only follow a regular rule of type LR. The type of the derived regular rule is given by the first symbol of the type of the first derivation regular rule followed by the second symbol of the type of the last derivation regular rule. If the derivation becomes a static repeating cycle or halts, the last symbol of the type of the derived regular rule is C or H respectively.

Proof. The first part of the theorem needs a proof that each member of L_{n+1} is a subset of a member of L_n determined by the addition of a single extra symbol to that member of L_n at the opposite end of the string of symbols to the pointer. Using the relationship between R_n and L_{n+1} defined in Algorithms 1 and 2 this is equivalent to the statement that each member of R_n is a subset of a member of R_{n-1}. It is this statement that will be proved by induction. This is obviously true for $n = 2$ because the substitution steps are then just single TM steps and the end result of computing on a 2-square length of tape (if a halting condition is not reached) is a CS of length 2 that must be a subset of the CS which is the RHS of the last regular rule applied.

Let $x \in L_{n+1}$. It has an extra symbol say s_1 at the pointer added to a member y of R_n, which is itself by assumption a subset of a member say z of R_{n-1} determined by a single extra symbol say s_2 at the opposite end of the string of symbols from the pointer. Therefore s_1 and s_2 are at opposite ends of the string of symbols in L_{n+1}. Adding back the symbol s_1 at the pointer to z gives $t \in L_n$ that contains x. Therefore a regular rule of length n can be applied to x. The result, say w, is a subset of a member of R'_n. The subset is formed by adding the extra symbol s_2 to the member of R'_n provided it is actually a member of R_n. If this is true i.e. the applied regular rule does not have a type where its second symbol is 'C' or 'H', w either has the pointer at the same end of the string of symbols as x with new symbol not yet known and the derivation terminates, or with the pointer at symbol s_2. Which is the case depends on whether the type of the applied regular rule (1) does not send the pointer to the opposite end of the string (type RR or LL) or (2) does so (type RL or LR) respectively. In case (2) the result is a member of L_{n+1} and the argument can be repeated, now with the pointer at the opposite end of the string of symbols to where it started. In case (1) the sequence of substitutions terminates and the result is the end point of the computation on the $n+1$ square length of tape (because the next symbol to be read is not yet known), and is obviously the same endpoint that would be reached by using single TM steps, and the result is by definition in R_{n+1} and is a subset of the member of R_n which is the RHS of the last applied regular rule. By definition, every member of R_{n+1} can be obtained by this argument, therefore every member of R_{n+1} is a subset of a member of R_n. If the last applied regular rule is such that the second symbol of its type is 'C' or 'H', this terminates the derivation and then this type obviously applies to the derived regular rule also. If the argument has to be repeated, either an endless stationary cycle (condition 'C') occurs or the computation ends in an applied rule of type 'H' or 'C', or it terminates with an applied rule of type 'LL' or 'RR', when the above argument holds. This completes the proof by induction. The remaining statements of the theorem are now obvious. $\square$

Consider the following example of a TM (TM1).

$$\begin{array}{ll} (a1) & A\underline{0} \rightarrow B1\_ \\ (a2) & A\underline{1} \rightarrow C\_1 \\ (a3) & B\underline{0} \rightarrow C1\_ \\ (a4) & B\underline{1} \rightarrow H1\_ \\ (a5) & C\underline{0} \rightarrow A\_0 \\ (a6) & C\underline{1} \rightarrow A0\_ \end{array} \tag{1}$$

I find it convenient and rapid to do the computations proceeding down the page because most of the symbols don't need to be changed in one substitution step and they are merely copied from one line to the next when a computation rule is applied. R_1 is the set $\{B1\_, C\_1, C1\_, A\_0, A0\_\}$. Note that the halting CS $H1\_$ was omitted. The set L_2 is therefore

$$\{B1\underline{0},\ B1\underline{1},\ C\underline{0}1,\ C\underline{1}1,\ C1\underline{0},\ C1\underline{1},\ A\underline{0}0,\ A\underline{1}0,\ A0\underline{0},\ A0\underline{1}\}\,.$$

The notation

$$\begin{array}{c} B1\underline{0} \\ C11\_ \end{array} \tag{2}$$

represents a computation of 1 step using the computation rule $B\underline{0} \to C1\_$ and shows that $B1\underline{0} \to C11\_$. The following computation of 3 steps shows that $C\underline{1}1 \to A\_01$:

$$\begin{array}{r} C\underline{1}1 \\ A0\underline{1} \\ C\underline{0}1 \\ A\_01 \end{array} \tag{3}$$

Likewise all the computations starting from L_2 are

$$\begin{array}{llll} B1\underline{0} \to C11\_ & B1\underline{1} \to H11\_ & C\underline{0}1 \to A\_01 & C\underline{1}1 \to A\_01 \\ C1\underline{0} \to C\_10 & C1\underline{1} \to A10\_ & A\underline{0}0 \to C11\_ & A\underline{1}0 \to C\_10 \\ A0\underline{0} \to B01\_ & A0\underline{1} \to A\_01 & & \end{array} \tag{4}$$

This shows that

$$R_2' = \{C11\_,\ H11\_,\ A\_01,\ B01\_,\ C\_10,\ A10\_\}$$

and

$$R_2 = \{C11\_,\ A\_01,\ B01\_,\ C\_10,\ A10\_\}\,.$$

Similarly

$$L_3 = \{C11\underline{0}, C11\underline{1}, A\underline{0}01, A\underline{1}01, B01\underline{0}, B01\underline{1}, C\underline{0}10, C\underline{1}10, A10\underline{0}, A10\underline{1}\}\,. \tag{5}$$

In the sense defined by Theorem 1, the regular rules for a TM completely describe its behaviour. This however is not a satisfactory description because there are always an infinite number of regular rules for a TM except in trivial cases.

There is a redundancy in the output of Algorithm 5.3 that generates the regular rules (RR). For example the rule $B1\underline{0} \to C11\_$ when interpreted as a string substitution is redundant because the rule $B\underline{0} \to C1\_$ from which it is derived in 1 step has exactly the same effect in the subset of cases where there is the symbol 1 immediately to the left of the pointer at the beginning. Therefore $B1\underline{0} \to C11\_$ is clearly redundant and does not allow any speed up of the computation. However for example $C\underline{1}1 \to A\_01$ in three steps is clearly not redundant. In many cases Algorithm 5.3 will generate RR of length $n+1$ that are derived in one step from a RR of length n. In that case the RR of length $n+1$ is redundant and should not be recorded. This RR has the same redundant symbol on its LHS and RHS and clearly cannot have type

LR or RL. In this case I will refer to the regular rule as being *reducible*. For example $B1\underline{0} \to C11\_$ is reducible to $B\underline{0} \to C1\_$. Rules derived in more than one step or have length 1 are *irreducible*. Every reducible regular rule (RRR) r has a unique irreducible regular rule (IRR) $I(r)$ which derives it. Only IRR should be recorded. $r = I(r)$ if and only if r is irreducible so for example $I(B1\underline{0} \to C11\_) = B\underline{0} \to C1\_$. But note that $I(r)$ cannot necessarily be obtained from r by removing all the symbols that appear to be redundant on both sides, For example $C\underline{1}1 \to A\_01$ does not imply the rule $C\underline{1} \to A\_0$ which is actually not true for this TM.

The final result of computing on a segment of tape of length n as in Theorem 5.2 is, if a halt or a cycle has not occurred, in R_n i.e. it is in R_i^+ for some $i \leq n$, where I introduce the notations R_i^+ is the set of RHS's of IRR of length i and R_i^0 is the set of RHS's of RRR of length i so that $R_i = R_i^+ \cup R_i^0$.

These ideas suggest a faster method than using Algorithm 5.3 for obtaining the IRR which is described later together with a computer implementation of it [2]. The rough idea is to proceed similarly to Algorithm 5.3 but omit recording RRR i.e. rules resulting from derivations with only one step. To find the matching regular rule, search backwards through the current set of IRR with lengths less than that of the one being derived. This will ensure that the longest applicable IRR will be used. Stop if there is some positive integer l such that there are no IRR of length l, because then there are no IRR of length $\geq l$.

There is a version of Theorem 5.2 that applies to the IRR. Because the RRR have been eliminated, instead of applying a regular rule r of length n, $I(r)$ must be used instead in any derivation.

Theorem 5.5. *Every computation of the TM that does not halt or enter a stationary cycle, using only a portion of the tape of length n and passing every position in that length and continues until the pointer reaches just past where the symbols are given, can be obtained by a unique sequence of n applications of the IRR. The ith rule in the sequence is a rule of length less than or equal to i. The application of each rule involves reading a symbol on the part of the tape not yet passed by the pointer.*

At each stage a RR r of length n can be applied to continue the computation, which can be replaced by its irreducible form $I(r)$ of length say $l \leq n$, so the result of applying the rule is a subset of the RHS of $I(r)$.

If the computation halts or cycles, this sequence is truncated where the last applied rule is respectively an irreducible halting rule (type LH or RH) or an irreducible cycling rule (type LC or RC).

It seems obvious that if a proper subset of the set of irreducible rules is used in place of the complete set, then Theorem 5.5 would not hold. Therefore a set of irreducible rules will be defined as complete if and only if it satisfies Theorem 5.5.

If a proper subset of the IRR, S considered, then there is some computation of length n that cannot be obtained by a sequence of n applications of rules in S where the ith rule in the sequence has length less than or equal to i and involves reading a symbol on the part of the tape not yet passed by the pointer. Suppose from the complete set of irreducible rules for a TM, rule r was omitted to get the set S. Now consider a computation where r is needed to make a substitution step. This obviously exists by Theorem 5.4 for any regular rule r. Now consider the representation of it by irreducible rules without r, i.e. the set S. At the point where r would be needed to make the substitution step, one must fall back on a shorter irreducible rule which will necessarily lead to more substitutions being needed than if r was allowed to be used, and more than one substitution step would correspond to the addition of the single symbol where rule r was applicable. These arguments establish the following characterisation of completeness of the set of IRR.

Theorem 5.6. *A set of IRR for a TM is complete if and only if for every distinct RHS of an IRR that is non-halting and is not a stationary cycling rule, and every possible new symbol that could be read, the CS that combines this RHS with the new symbol read, matches the LHS of one of the set of IRR (the longest one possible must be taken) such that after this rule is applied there are only three possible conditions that can arise: (1) new symbol needs to be read to continue the computation further (2) the last IRR applied is a stationary cycling rule (3) the last IRR applied is a halting rule*

6 Analysis of an example of a Turing Machine with a finite number of irreducible regular rules

Returning to example TM1, the result (6) contains the list of all the IRR derivable from it together with the number of TM steps needed for each. This was verified by the computer program [2]. Note that not all of these rules can be obtained by generating the list of new LHS's for the IRR from the previous RHS's of the IRR of length shorter by 1. This is because sometimes an RRR has a RHS that can lead to a LHS of a new IRR. There are three examples of this in this list, (b7), (b10) and (b11). These are also the set of unreachable CSs in Table (1) below. (It is a good exercise to verify this while generating all the RR up to length 6 because in this example there are not many computations to be carried out.) This unfortunately means that the algorithm for generating the IRR is more complex than might naively have been expected. The algorithm employed in the program [2] is based on Theorem 10 described later.

$$\begin{array}{lrr}
(b1) & C\underline{1}1 \to A\_01 & 3 \\
(b2) & C1\underline{0} \to C\_10 & 2 \\
(b3) & A\underline{0}0 \to C11\_ & 2 \\
(b4) & A0\underline{1} \to A\_01 & 2 \\
(b5) & C11\underline{0} \to A\_010 & 5 \\
(b6) & A\underline{0}01 \to A110\_ & 3 \\
(b7) & A10\underline{1} \to C\_101 & 3 \\
(b8) & A\underline{0}010 \to B1101\_ & 4 \\
(b9) & A110\underline{1} \to A\_0101 & 6 \\
(b10) & C011\underline{0} \to B1101\_ & 9 \\
(b11) & A0110\underline{1} \to H11011\_ & 11 \\
(b12) & A\underline{0}0101 \to H11011\_ & 5
\end{array} \tag{6}$$

In any computation of this TM, Table 1 shows all the CSs and the corresponding IRR that must be applied to continue the computation as defined in Theorem 5.6. For each case there is at least one possible new rule that can be applied depending on the context of nearby symbols, and just one longest IRR, which is the one stated, after which a new symbol must be read.

This TM is simple enough to have a finite number of IRR. A consequence of this is that Table 1, that was constructed using all the (IRR, new symbol) combinations, has a finite number of rows. For each of the distinct results in the last column of Table 1, being the result of the application of an IRR, again one of the IRR in lists (1) or (6) can be applied to continue the computation to the point where the next symbol must be read etc.. For example rule $a5$ with a 0 at the pointer could be followed by IRR $b3$, $b6$, $b8$, or $b12$, but following $a5$ with $b6$ would imply knowing about the 1 on the right, so $b6$ would have been preceded by $b1$ or $b4$ if using the IRR optimally in the sense of Theorem 5.5. Table 1 shows that the TM can be effectively speeded up by replacing it by the set of IRR and at each step using the table to determine the next step and its effect depending on the new symbol read.

Table 1 with Theorem 5.6 shows the completeness of the combined list (1) and (6) of IRR. In this example Theorem 6.1 also applies demonstrating that its set of IRR is finite and is therefore complete.

It is interesting to note that if the IRR whose LHSs are obtained from the RHSs of RRR are omitted from Table (1), the set of rules appearing in the first and third columns would be the same. This suggests that this subset of the IRR has some other significance.

Theorem 6.1. *If every RR of length n with the pointer at the left in its LHS is reducible, then every RR of length greater than n with the pointer at the left*

Table 1: For each irreducible regular rule (IRR) of TM1 and each new symbol read, the next IRR to be applied is given and its effect

IRR	RHS of rule + symbol	Next IRR	Result of rule
$a1$	$B1\underline{0}$	$a3$	$C11\_$
$a1$	$B1\underline{1}$	$a4$	$H11\_$
$a2$	$C\underline{0}1$	$a5$	A_01
$a2$	$C\underline{1}1$	$b1$	A_01
$a3$	$C1\underline{0}$	$b2$	C_10
$a3$	$C1\underline{1}$	$a6$	$A10\_$
$a5$	$A\underline{0}0$	$b3$	$C11\_$
$a5$	$A\underline{1}0$	$a2$	C_10
$a6$	$A0\underline{0}$	$a1$	$B01\_$
$a6$	$A0\underline{1}$	$b4$	A_01
$b1$ and $b4$	$A\underline{0}01$	$b6$	$A110\_$
$b1$ and $b4$	$A\underline{1}01$	$a2$	C_101
$b2$	$C\underline{0}10$	$a5$	A_010
$b2$	$C\underline{1}10$	$b1$	A_010
$b3$	$C11\underline{0}$	$b5$	A_010
$b3$	$C11\underline{1}$	$a6$	$A110\_$
$b5$	$A\underline{0}010$	$b8$	$B1101\_$
$b5$	$A\underline{1}010$	$a2$	C_1010
$b6$	$A110\underline{0}$	$a1$	$B1101\_$
$b6$	$A110\underline{1}$	$b9$	A_0101
$b7$	$C\underline{0}101$	$a5$	A_0101
$b7$	$C\underline{1}101$	$b1$	A_0101
$b8$ and $b10$	$B1101\underline{0}$	$a3$	$C11011\_$
$b8$ and $b10$	$B1101\underline{1}$	$a4$	$H11011\_$
$b9$	$A\underline{0}0101$	$b12$	$H11011\_$
$b9$	$A\underline{1}0101$	$a2$	C_10101

in its LHS is also reducible. Let n_L, if it exists, be one less than the smallest such value of n, i.e. the largest value of n such that some RR exists of length n with the pointer at the left in its LHS that is irreducible. Then for every RR r with the pointer at the left in its LHS, $I(r)$ is another RR of the same type, with length $\leq n_L$.

Proof. Let r'' be a RR of length $n+1$ with the pointer on the left in its LHS which is say $q\underline{s}s_1s_2\ldots s_n$. Then by Algorithm 5.3 there exists a RR r' of length n such that its RHS. is $q\_s_1s_2\ldots s_n$. By the induction hypothesis r' is reducible. Then r' must have type LL, type RL being excluded because the pointer cannot go so far in the single substitution for its derivation from a RR r of length $n-1$. So r' is derivable from a RR r of length $n-1$ with RHS. $q\_s_1s_2\ldots s_{n-1}$ i.e. the same as the RHS of r' with 1 symbol removed. Therefore from Algorithm 5.3 there is a RR t of length n with LHS matching the RHS of r and with symbol s i.e. $q\underline{s}s_1s_2\ldots s_{n-1}$. The RR t is derivable in 1 step by the induction hypothesis, therefore t is of type LL. Therefore t can be used to derive the RHS of r'' in 1 step. Induction on the length of the rules completes the proof.

The following diagram relates the 4 different RRs used in this proof, with the subscripts denoting the length of the rules. The vertical arrows represent the relation “the RHS of the first RR determines the LHS of the second RR by the addition of symbol s”.

$$\begin{tikzcd}
r_{n-1} \arrow[r, "\text{used in the derivation of}"] \arrow[d] & r'_n \arrow[d] \\
t_n \arrow[r, "\text{used in the derivation of}"] & r''_{n+1}
\end{tikzcd}$$

□

Similar arguments of course hold in the mirror image case where the pointer starts at the right and n_R is defined to be the corresponding parameter. Either or both of these values may not exist therefore in the set of all TM's, there are 4 cases to consider, neither n_L nor n_R exist,n_L only exists, n_R only exists, and both n_L and n_R exist.

Suppose n_L exists. Consider a computation on a portion of the tape of length n, where $n > n_L$. Using the representation of it as a sequence of substitutions given by Theorem 5.5, after m substitutions, where $m \geq n_L$, m of the original symbols on the tape have been read, and if the next symbol to be read is on the left, the next IRR r to be applied must be such that length$(r) \leq n_L$. This must leave the pointer at a location where a symbol has not been read before i.e. on the left, so all remaining original symbols to be scanned will be on the left. This establishes

Theorem 6.2. *If n_L exists, then in any computation of the TM if at some stage in the computation at least n_L of the original symbols on the tape have been read, and the next original symbol on the tape to be read is on the left of these, then all the subsequent substitution cycles in Theorem 5.5 use regular rules of type LC, LH or LL with length $\leq n_L$, therefore the computation from that point onward is carried out within a moving window of length n_L that moves 1 step to the left each time the pointer reaches a location where it has never been before, unless terminated by a regular rule of type LC or LH.*

Of course, there is a corresponding left-right reversal of this theorem involving n_R and right-moving substitution steps. Now it is possible to describe (exercise) the types of behaviour of TM's when either n_L or n_R or both exist. In TM1, from (6), both n_L and n_R exist, and are both 5.

7 Analysis of TM's with an infinite number of irreducible regular rules

In the following example (TM2) a similar analysis shows that the number of IRR is infinite but it appears that the IRR can all be obtained by simple formulae. TM2 is as follows:

$$\begin{array}{l}
(1)\ A\underline{0} \to B1\_ \\
(2)\ A\underline{1} \to A\_1 \\
(3)\ B\underline{0} \to A\_0 \\
(4)\ B\underline{1} \to C0\_ \\
(5)\ C\underline{0} \to D1\_ \\
(6)\ C\underline{1} \to C1\_ \\
(7)\ D\underline{0} \to B0\_ \\
(8)\ D\underline{1} \to H1\_
\end{array} \tag{7}$$

It is easy to check that the IRR for TM2 with lengths up to 4, together with the number of TM steps needed for each, are as follows:

$$\begin{array}{rll}
(1) & B1\underline{0} \to A\_10 & 2 \\
(2) & A\underline{0}1 \to C10\_ & 2 \\
(3) & A\underline{0}0 \to A\_10 & 3 \\
(4) & B0\underline{0} \to A\_10 & 4 \\
(5) & A\underline{0}10 \to D101\_ & 3 \\
(6) & A\underline{0}11 \to C101\_ & 3 \\
(7) & B10\underline{0} \to A\_110 & 5 \\
(8) & A\underline{0}110 \to D1011\_ & 4 \\
(9) & A\underline{0}111 \to C1011\_ & 4 \\
(10) & B010\underline{0} \to D1011\_ & 9 \\
(11) & B110\underline{0} \to A\_1110 & 6
\end{array} \tag{8}$$

After continuing with these computations a little further it is easy to conjecture general results for the IRR as follows:

	Rule	# TM steps	condition	
(a)	$B1\underline{0} \to A\_10$	2		
(b)	$A\underline{0}0 \to A\_10$	3		
(c)	$A\underline{0}1^{n+1}0 \to D101^{n+1}\_$	$n+3$	$n \geq 0$	(9)
(d)	$A\underline{0}1^{n+2} \to C101^{n+1}\_$	$n+3$	$n \geq -1$	
(e)	$B01^n0\underline{0} \to D101^{n+1}\_$	$2n+7$	$n \geq 1$	
(f)	$B1^{n+1}0\underline{0} \to A\_1^{n+2}0$	$n+5$	$n \geq -1$	

Note that apart from the first two rules, the other IRR with the parameter n are expressed so that they all have the same length $(n+3)$. This is necessary to avoid the possibility of circularity in the induction proof of the following theorem that simultaneously uses induction on n to prove that all these sets of rules are IRR.

Theorem 7.1. *For TM2, the set of rules given by* (9) *are all IRR.*

What needs to be proved here is that these rules are (i) valid (i.e. the computations are correct) (ii) regular (iii) irreducible.

Proof. To establish validity by induction on the length of the rules it is straightforward to first check that the rules in (9) hold for length ≤ 4. For the induction step, take the LHS's of each of the rules in (9) and replace n by $n+1$. Apply the rules in (7) and (9) as necessary to continue the computation until the procedure terminates in each case. The symbols above the arrows indicate either the number of machine steps or the regular rule being used. The results are as follows, with the total number of TM steps indicated at the right.

These results are all the same as the corresponding conjectured results with n replaced by $n+1$, so that validity can be extended to the next value of n. This proof also shows that the derived rules are irreducible because more than one derivation step is involved in each case.

Rule	# TM steps
(c') $A\underline{0}1^{n+2}0 \xrightarrow{d} C101^{n+1}\underline{0} \xrightarrow{1} D101^{n+2}\_$	$n+4$
(d') $A\underline{0}1^{n+3} \xrightarrow{d} C101^{n+1}\underline{1} \xrightarrow{1} C101^{n+2}\_$	$n+4$
(e') $B01^{n+1}0\underline{0} \xrightarrow{f} A\underline{0}1^{n+2}0 \xrightarrow{d} C101^{n+1}\underline{0} \xrightarrow{1} D101^{n+2}\_$	$2n+9$
(f') $B1^{n+2}0\underline{0} \xrightarrow{f} A\underline{1}1^{n+2}0 \xrightarrow{1} A\_1^{n+3}0$	$n+6$

To establish that these rules are regular, the same type of reasoning works in all four cases. This reasoning is based on Algorithm 5.1 and works backwards from the rule being assessed and ends in a rule that is so short that it is obviously regular. The regularity of the rule being assessed follows from the regularity of the next rule in the sequence, and so on, hence the result is proved. I will only quote the LHS of the rules being referred to. In this manner the proof of the regularity of $A\underline{0}1^{n+3}$ is indicated by following:

$$A\underline{0}1^{n+3} \leftarrow A\underline{1}1^{n+2} \leftarrow A\underline{1}1^{n+1} \cdots \leftarrow A\underline{1} \qquad (10)$$

The first step is argued as follows: $A\underline{0}1^{n+3}$ is the LHS of a regular rule if and only if A_1^{n+3} is the RHS of a regular rule by algorithm 5.1. But because $A\underline{1}1^{n+2} \rightarrow A\_1^{n+3}$, the regularity of $A\underline{0}1^{n+3}$ follows from the regularity of $A\underline{1}1^{n+2}$. The fact that only 1 TM step is needed makes the argument as simple as possible. Note that the argument cannot work in the other direction and the left pointing arrows indicate this. The $\cdots$ represents an obvious induction argument. The proofs of the other regularity statements are indicated thus:

$$B01^{n+1}0\underline{0} \leftarrow D01^{n+1}\underline{0} \leftarrow C01^{n}\underline{1} \leftarrow C01^{n-1}\underline{1} \ \cdots \leftarrow C0\underline{1} \leftarrow B\underline{1} \qquad (11)$$

$$B1^{n+2}0\underline{0} \leftarrow D1^{n+2}\underline{0} \leftarrow C1^{n+1}\underline{0} \leftarrow C1^{n}\underline{1} \leftarrow C1^{n-1}\underline{1} \ \cdots \leftarrow C\underline{1} \qquad (12)$$

$$A\underline{0}1^{n+2}0 \leftarrow A\underline{1}1^{n+1}0 \leftarrow A\underline{1}1^{n}0 \ \cdots \leftarrow A\underline{1}0 \leftarrow B\underline{0} \qquad (13)$$

□

To establish completeness of the set of IRR using Theorem 5.6, Table 2 similar to Table 1 is easily constructed showing which IRR is needed to continue the computation following each IRR and symbol pair such that after this rule application a new symbol needs to be read unless a halting or cycling condition occurs.

The body of Table 2 gives the next rule to be applied, given the first rule and the next symbol read. Note that in each case after the new rule is applied,

	next symbol	
rule	0	1
1	a	4
2	$d, n = 0$	2
3	b	2
4	5	6
5	7	8
6	5	6
7	$f, n = 0$	4
8	halt	halt
a or b	$c, n = 0$	2
c or e	7	8
d	5	6
f	c	2

Table 2: Table establishing the completeness of the IRR for TM2.

the computation needs a new symbol to continue, and all the rules are IRR from (7) or (9).

Putting all these results together for TM2 we have

Theorem 7.2. *Given TM2 defined by* (7)*, the complete infinite set of IRR for TM2 consists of the rules in* (7) *and* (9)*.*

8 A small TM with a complex behaviour

Consider TM3 which has the machine table given by:

$$\begin{array}{l} (1)\ A\underline{0} \to B1\_ \\ (2)\ A\underline{1} \to H1\_ \\ (3)\ B\underline{0} \to C\_1 \\ (4)\ B\underline{1} \to A1\_ \\ (5)\ C\underline{0} \to A0\_ \\ (6)\ C\underline{1} \to C\_0 \end{array} \tag{14}$$

The IRR derived from TM3 of length up to 5 together with the number of TM steps needed for the derivation of each are as follows:

Rule	# TM steps
(1) $B1\underline{0} \to C\_01$	2
(2) $C\underline{0}0 \to B01\_$	2
(3) $C\underline{0}1 \to H01\_$	2
(4) $B01\underline{0} \to A011\_$	5
(5) $C\underline{0}01 \to A011\_$	3
(6) $B11\underline{0} \to C\_001$	3
(7) $C\underline{0}00 \to A011\_$	7
(8) $B111\underline{0} \to C\_0001$	4
(9) $C\underline{0}000 \to B0111\_$	8
(10) $C\underline{0}001 \to H0111\_$	8
(11) $B0111\underline{0} \to A01111\_$	13
(12) $C\underline{0}0001 \to A01111\_$	9
(13) $B1111\underline{0} \to C\_00001$	5
(14) $C\underline{0}0000 \to A01111\_$	21

(15)

Note that in these results the general formulae appear to be mostly alternating between the even and odd numbers of symbols. This is confirmed by continuing the calculations to larger values of n. The general formulae for the set of IRR can be conjectured as follows where the numbers of machine steps are defined by the 7 functions yet to be determined, and they are all valid when $n \geq 0$:

	Length	Rule	# TM steps
(a)	$2n+2$	$B1^{2n+1}\underline{0} \to C\_0^{2n+1}1$	$a(n)$
(b)	$2n+2$	$C\underline{0}0^{2n+1} \to B01^{2n+1}\_$	$b(n)$
(c)	$2n+2$	$C\underline{0}0^{2n}1 \to H01^{2n+1}\_$	$c(n)$
(d)	$2n+3$	$B01^{2n+1}\underline{0} \to A01^{2n+2}\_$	$d(n)$
(e)	$2n+3$	$C\underline{0}0^{2n+1}1 \to A01^{2n+2}\_$	$e(n)$
(f)	$2n+3$	$B1^{2n+2}\underline{0} \to C\_0^{2n+2}1$	$f(n)$
(g)	$2n+3$	$C\underline{0}0^{2n+2} \to A01^{2n+2}\_$	$g(n)$

(16)

The proof of validity of these rules is again by induction on n. Applying the same method as before gives the following results with the total number of TM steps indicated at the right. These results (17) are all the same as the corresponding conjectured results with n replaced by $n + 1$, provided that Equations 18 relating the numbers of TM steps are satisfied. In these results, only the IRR are used as Theorem (5.5) indicates and the even (a-c) and odd (d-g) length rules have to be established simultaneously. The length of the rules increases by 2 for each increase of n by 1. The argument is as follows:

$$\begin{array}{ll}
\text{Rule} & \#\ \text{TM steps} \\
(a')\ B1^{2n+3}\underline{0} \xrightarrow{f} C\underline{1}0^{2n+2}1 \xrightarrow{(1)} C\_0^{2n+3}1 & f(n)+1 \\
(b')\ C\underline{0}0^{2n+3} \xrightarrow{g} A01^{2n+2}\underline{0} \xrightarrow{(1)} B01^{2n+3}\_ & g(n)+1 \\
(c')\ C\underline{0}0^{2n+2}1 \xrightarrow{g} A01^{2n+2}\underline{1} \xrightarrow{(1)} H01^{2n+3}\_ & g(n)+1 \\
(d')\ B01^{2n+3}\underline{0} \xrightarrow{a'} C\underline{0}0^{2n+3}1 \xrightarrow{b'} B01^{2n+3}\underline{1} \xrightarrow{(1)} A01^{2n+4}\_ & f(n)+g(n)+3 \\
(e')\ C\underline{0}0^{2n+3}1 \xrightarrow{b'} B01^{2n+3}\underline{1} \xrightarrow{(1)} A01^{2n+4}\_ & g(n)+2 \\
(f')\ B1^{2n+4}\underline{0} \xrightarrow{a'} C\underline{1}0^{2n+3}1 \xrightarrow{(1)} C\_0^{2n+4}1 & f(n)+2 \\
(g')\ C\underline{0}0^{2n+4} \xrightarrow{b'} B01^{2n+3}\underline{0} \xrightarrow{d'} A01^{2n+4}\_ & f(n)+2g(n)+4
\end{array} \tag{17}$$

where the equations relating the numbers of TM steps are

$$\begin{array}{l}
a(n+1)=f(n)+1 \\
b(n+1)=g(n)+1 \\
c(n+1)=g(n)+1 \\
d(n+1)=f(n)+g(n)+3 \\
e(n+1)=g(n)+2 \\
f(n+1)=f(n)+2 \\
g(n+1)=f(n)+2g(n)+4 \\
a(0)=2,\ b(0)=2,\ c(0)=2,\ d(0)=5,\ e(0)=3,\ f(0)=3,\ g(0)=7
\end{array} \tag{18}$$

from which it follows that $f(n)=2n+3$ and

$$g(n+1)-2g(n)=2n+7. \tag{19}$$

A solution of (19) for $g(.)$ of the form $g(n)=An+B$ can be easily found which is given by $g(n)=-2n-9$. The general solution of the reduced equation $g(n+1)-2g(n)=0$ is easily shown to be $C2^n$, where C is an arbitrary constant, therefore the general solution of (19) is $g(n)=C2^n-2n-9$. The initial condition determines $C=16$ therefore $g(n)=16.2^n-2n-9$. Now all these functions can be determined as follows in this order:

$$\begin{aligned}
f(n)&=2n+3 \\
g(n)&=2^{n+4}-2n-9 \\
a(n)&=2n+2 \\
c(n)&=2^{n+3}-2n-6 \\
d(n)&=2^{n+3}-3 \\
e(n)&=2^{n+3}-2n-5 \\
b(n)&=2^{n+3}-2n-6.
\end{aligned} \tag{20}$$

Note that the initial conditions for a to e are not involved, so these could be deduced from the other equations.

What is interesting about this argument is that if some partial information about the IRR in general can be conjectured correctly, (in this case the form of the initial and final CS's), it might still be possible to complete the induction proof after solving some other recurrence relations for the remaining information (here the number of machine steps).

This establishes validity of (16) and (20), and that these rules are irreducible follows from (17) which shows that the number of steps of derivation is greater than 1 in each case. The proof that these rules are regular follows a similar form to the previous example. The only extra symbol used here is $\equiv$ which is used to replace the symbol at the pointer because this does not affect the regularity of the CS. The proofs are indicated as follows:

$$\begin{gathered}
B1^{2n+1}\underline{0} \leftarrow A1^{2n}\underline{0} \leftarrow B1^{2n-1}\underline{1} \equiv B1^{2n-1}\underline{0} \ \cdots \ B1\underline{0} \leftarrow A\underline{0} \\
C\underline{0}0^{n} \leftarrow C\underline{1}0^{n-1} \equiv C\underline{0}0^{n-1} \ \cdots \ C\underline{0}0 \leftarrow C\underline{1} \equiv C\underline{0} \\
B01^{2n+1}\underline{0} \leftarrow A01^{2n}\underline{0} \leftarrow B01^{2n-1}\underline{1} \equiv B01^{2n-1}\underline{0} \ \cdots \ B01\underline{1} \leftarrow A0\underline{0} \leftarrow C\underline{0} \\
C\underline{0}0^{n}1 \equiv C\underline{1}0^{n}1 \leftarrow C\underline{1}0^{n-1}1 \ \cdots \ C\underline{1}1 \leftarrow B\underline{0} \\
B1^{2n+2}\underline{0} \equiv B1^{2n+2}\underline{1} \leftarrow A1^{2n+1}\underline{0} \leftarrow B1^{2n}\underline{1} \ \cdots \ B\underline{1}
\end{gathered} \tag{21}$$

The completeness of the set of IRR can be established as before using Theorem (5.6). The table is as follows:

	next symbol	
rule	0	1
1	$a, n = 0$	4
2	halt	halt
3	$c, n = 0$	6
4 or 5	1	2
6	$b, n = 0$	6
a	e	6
b	d	4
c	halt	halt
d or e	1	2
f	$c, n \leftarrow n+1$	6
g	1	2

Table 3: Table establishing the completeness of the IRR for TM3.

The body of the Table 3 gives the next rule to be applied, given the first rule and the next symbol read. Note that in each case after the new rule is applied, the computation needs a new symbol to continue, and all the rules are IRR.

9 More complex examples and the branching tree structure of the irreducible regular rules

After analysing these examples I started looking at a far more complex example (TM4) with 24 instructions.

Although this example was too complex for me to completely analyse like the above examples, the partial analysis I did revealed an important property of the IRR and lead to an efficient algorithm [2] for computing the IRR for any TM. By trying to use the above technique it is obvious that it is not going to be easy to conjecture the general formula for the IRR. The reason is that their number appears to grow exponentially with the length of the regular rule, although computing each case to get the RHS seems to be easy. The example was as follows:

$$\begin{array}{llll}
A\underline{0} \to F\_2 & A\underline{1} \to D\_3 & A\underline{2} \to A1\_ & A\underline{3} \to B2\_ \\
B\underline{0} \to E\_2 & B\underline{1} \to D3\_ & B\underline{2} \to C1\_ & B\underline{3} \to A0\_ \\
C\underline{0} \to F\_3 & C\underline{1} \to A1\_ & C\underline{2} \to C\_2 & C\underline{3} \to D\_3 \\
D\underline{0} \to B0\_ & D\underline{1} \to A0\_ & D\underline{2} \to B\_2 & D\underline{3} \to D\_1 \\
E\underline{0} \to E3\_ & E\underline{1} \to E\_3 & E\underline{2} \to A1\_ & E\underline{3} \to B1\_ \\
F\underline{0} \to B2\_ & F\underline{1} \to C\_2 & F\underline{2} \to A\_0 & F\underline{3} \to F2\_
\end{array} \tag{22}$$

The first thing needed is formulae for all the LHS's of the IRR, then the formulae for their RHS's. The computer program [2] can generate the lists of such regular rules up to any given length in principle and sort them sensibly. After looking at these lists for this TM it was noticed that one could draw out their LHS's as branches of a tree rooted at the top with one main branch for each machine state. It is also obvious that there is some regularity because some sub-branches are identical to others at the same or different levels, which suggests a recursive description but it just evades an obvious formula. In fact it seems that a finite state machine is likely to be the description of these LHS's of regular rules, which would allow the choice of the next symbol in a LHS to depend not only on the current symbol but on which node that symbol is in a diagram.

To make the logical arguments easier to follow to demonstrate the 'tree' property of the IRR, the following abbreviations are introduced:
$D(r)$ is the number of derivation steps for regular rule r as defined in Algorithm 5.3.
$l(r)$ means the length of rule r.
IRR(l) is the set of irreducible regular rules of length l i.e. regular rules r such that $l(r) = l$ and $D(r) > 1$.
RRR(l) is the set of reducible regular rules of length l i.e. regular rules r such that $l(r) = l$ and $D(r) = 1$.
RR(l) = RRR$(l) \cup$ IRR(l) is the set of regular rules of length l.

Theorem 9.1. *For every rule* $r' \in \mathrm{IRR}(n+1)$ *there is a unique corresponding rule* $r \in \mathrm{IRR}(n)$ *such that* $\mathrm{LHS}(r') \subset \mathrm{LHS}(r)$*.*

If a rule $r \in \mathrm{IRR}(n)$ is such that $\mathrm{LHS}(r') \subset \mathrm{LHS}(r)$ then r will be said to match r', because the LHS of r is a substring of the LHS of r' with 1 symbol deleted at the end opposite end of the string of symbols to the pointer.

This theorem therefore guarantees that the LHSs of rules in $\mathrm{IRR} = \bigcup_{n=1}^{\infty} \mathrm{IRR}(n)$ have a tree structure that can 'grow' as the length of the rules increases.

Proof. From Theorem 5.4, every rule $r' \in \mathrm{RR}(n+1)$ has a matching (on the LHS) rule $r \in \mathrm{RR}(n)$. This is in particular true for all rules $r' \in \mathrm{IRR}(n+1)$. Suppose $r \in \mathrm{RRR}(n)$ so cannot have type LR or RL, therefore r' must be derived from r in one step (see Theorem 5.4). i.e. $r' \in \mathrm{RRR}(n+1)$. This contradicts the assumption, so $r \notin \mathrm{RRR}(n)$ therefore $r \in \mathrm{IRR}(n)$. $\square$

After computing these 'trees' for TM4 a few steps it becomes obvious that some die out quickly, while others are maintained indefinitely as single branches, while others grow exponentially with repeating patterns. Also doing some analysis by hand for this example revealed that the reasoning used to construct the $\mathrm{IRR}(n)$ was quite complex but could be simplified by use of the following two theorems.

Theorem 9.2. *If* $q_1 s_2 \ldots s_{n-1}\underline{s_n} \rightarrow q_2\_s'_2 \ldots s'_n$ *is a RR then the following statements are equivalent: (1)* $q_1 s_1 s_2 \ldots s_{n-1}\underline{s_n}$ *is the LHS of an IRR and (2)* $q_1 s_1 s_2 \ldots s_{n-1}\_$ *is the RHS of a RR.*

This is the right-left reversed form.

If $q_1\underline{s_n}s_{n-1} \ldots s_2 \rightarrow q_2 s'_n \ldots s'_2\_$ *is a RR then the following statements are equivalent: (1)* $q_1\underline{s_n}s_{n-1} \ldots s_1$ *is the LHS of an IRR and (2)* $q_1\_s_{n-1} \ldots s_1$ *is the RHS of a RR.*

Proof. If $q_1 s_1 s_2 \ldots s_{n-1}\underline{s_n}$ is the LHS of an IRR, then simply because it is the LHS of a RR, by the definition of the set of RRs, $q_1 s_1 \ldots s_{n-1}\_$ is the RHS of a RR. Conversely, suppose $q_1 s_1 \ldots s_{n-1}\_$ is the RHS of a RR, and r is the RR $q_1 s_2 \ldots s_{n-1}\underline{s_n} \rightarrow q_2\_s'_2 \ldots s'_n$. Applying the RR r to $q_1 s_1 \ldots s_{n-1}\underline{s_n}$, which is the LHS of a RR, gives $q_2\underline{s_1}s'_2 \ldots s'_n$. Because here there is a symbol at the pointer, this derivation can be continued and therefore it requires more than one step, so the rule derived from this LHS is an IRR. $\square$

Theorem 9.3. $q_1 s_1 \ldots s_{n-1} s_n\_$ *is the RHS of a RRR if and only if there are states* q' *and* q'' *and strings of symbols* $s'_t \ldots s'_n$ *and* $s''_1 \ldots s''_{n-1}$ *and an integer* t *such that* $2 \leq t \leq n$ *and* $q' s'_t \ldots \underline{s'_n} \rightarrow q s_t \ldots s_n\_$ *is an IRR and* $q'' s''_1 \ldots s''_{n-1} \rightarrow q' s_1 \ldots s_{t-1} s'_t \ldots s'_{n-1}\_$ *(type LR or RR) is a RR.*

Proof. If $qs_1\dots s_{n-1}s_{n\_}$ is the RHS of a RRR r' then its irreducible form $I(r') = r$ has RHS equal to $qs_t\dots s_{n\_}$ for some t such that $2 \le t \le n$ and there is a sequence of rules $r^{(1)}, r^{(2)}\dots r^{(t)}$ such that $r^{(1)} = r'$ and $r^{(t)} = r$ and $l(r^{(i)}) = n - i + 1$ and each rule in the sequence is derived from the next in one step. By Theorem (5.4) $r^{(1)}\dots r^{(t)}$ have type RR or LL and must be clearly RR in this case with the pointer ending on the right. Therefore this IRR r can be written as $q's'_t\dots \underline{s'_n} \to qs_t\dots s_{n\_}$. Then r' can be written as $q's_1\dots s_{t-1}s'_t\dots s'_{n-1}\underline{s'_n} \to qs_1s_2\dots s_{n\_}$ because the symbols on the left are not involved and can be simply added to both sides of the rule, one at a time, for each step in the recursive definition of the RR in Algorithm (5.1). Therefore from the definition of RR in Algorithm 5.1, $q's_1\dots s_{t-1}s'_t\dots s'_{n-1\_}$ is the RHS of a RR.

Conversely, adding the symbol s'_n at the pointer position to the RHS of the second RR in the theorem gives $q's_1\dots s_{t-1}s'_t\dots s'_{n-1}\underline{s'_n}$ is the LHS of a RR. Applying the first RR to this gives $qs_1\dots s_{t-1}s_t\dots s_{n-1}s_{n\_}$ in one step. ☐

Theorem 9.3 can be represented more succinctly as follows

$$
\begin{gathered}
qs_1\dots s_{n\_} \text{ is the RHS of a RRR} \Leftrightarrow \\
\exists q', q'', s'_t\dots s'_n, s''_1, \dots s''_{n-1}, t : 2 \le t \le n \text{ such that} \\
\left[\begin{array}{lr}
q's'_t\dots \underline{s'_n} \to qs_t\dots s_{n\_} & \text{is an IRR and} \\
q''s''_1\dots s''_{n-1} \to q's_1\dots s_{t-1}s'_t\dots s'_{n-1\_} & \text{is a RR of type LR or RR}
\end{array}\right]
\end{gathered}
$$

This is the right-left reversed form.

$$
\begin{gathered}
q\_s_n\dots s_1 \text{ is the RHS of a RRR} \Leftrightarrow \\
\exists q', q'', s'_t\dots s'_n, s''_1, \dots s''_{n-1}, t : 2 \le t \le n \text{ such that} \\
\left[\begin{array}{lr}
q'\underline{s'_n}\dots s'_t \to q\_s_n\dots s_t & \text{is an IRR and} \\
q''s''_{n-1}\dots s''_1 \to q'\_s'_{n-1}\dots s'_t s_{t-1}\dots s_1 & \text{is a RR of type RL or LL}
\end{array}\right]
\end{gathered}
$$

I wrote both mirror image forms of theorems 9.2 and 9.3 because it makes it easier to apply them by hand for finding extensions to the LHS's of rules in IRR(n) to obtain the LHS's of rules in IRR($n + 1$).

I give in outline one example next based on TM4 (22) to illustrate the procedure. The rule $B\underline{1}2322 \to A01211\_ \in$ IRR(5). The problem is to find the LHS's of all members of IRR(6) of the form $B\underline{1}2322x \to \dots$. By theorem 9.2, $B\underline{1}2322x$ is the LHS of an IRR if and only if B_2322x is the RHS of a RR. B_2322x is the RHS of an IRR if and only if $x = 3$, found from a previously computed list of IRR(5). Is B_2322x the RHS of a RRR? To answer this, Theorem 9.3 can be applied in its reversed form with $n = 5$ reducing this question to

$$\exists q', q'', s'_t \ldots s'_5, s''_1, \ldots s''_4, t : 2 \le t \le 5 \text{ such that}$$
$$\left[\begin{array}{l} q'\underline{s'_5} \ldots s'_t \to B\_ \text{ [initial substring of length } 6-t \text{ of } 2322x\text{] is an IRR and} \\ q''s''_4 \ldots s''_1 \to q'\_s'_4 \ldots s'_t\text{[final substring of length } t-1 \text{ of } 2322x\text{] is an RR} \\ \text{ of type RL or LL} \end{array}\right]$$

To determine whether this is true or not, start searching with $t = 2$ and increasing to 5. The first question should again be answered by reference to the previously computed set of IRR of length ≤ 4. If the answer is yes, the string(s) $s'_5 s'_4 \ldots s'_t$ must be identified. The second question is of the same type as before referring to a RR of length 4 i.e. 1 shorter than before. Thus whole procedure is recursive.

I have implemented a fast algorithm for generating these trees, based on automating the use of these theorems. The algorithm is fast because it does not waste a lot of time generating the RRR.

The following example (TM5) proved to be more of a challenge than TM1, TM2 or TM3 but not as tough as TM4.

$$\begin{array}{lll} A\underline{0} \to B2\_ & A\underline{1} \to D\_1 & A\underline{2} \to A0\_ \\ B\underline{0} \to C2\_ & B\underline{1} \to A\_1 & B\underline{2} \to D0\_ \\ C\underline{0} \to C1\_ & C\underline{1} \to D\_1 & C\underline{2} \to C2\_ \\ D\underline{0} \to A0\_ & D\underline{1} \to B2\_ & D\underline{2} \to C\_1 \end{array} \tag{23}$$

For this TM, the computations for the IRR of length 2 and 3 mostly follow the same sequences of CS's, so this situation is best represented by (24). Here the single-headed arrows each represent a single TM step, and the double headed arrow represents the computation that can go in either direction. Thus the endpoint (RHS) of the IRR here is a stationary cycle of two CS's, and this is the first example of a stationary cycle in this paper. Such an endpoint is recognised when a CS reached by the computation is repeated (here $A0\underline{1}$), and this is regarded as the endpoint of the computation except if the computation starts at $D\underline{0}1$ when this CS is also the first to be reentered. Thus the LHS of the IRR of length 2 are all the CS's leading to these endpoints in more than one step i.e. $C\underline{2}1 \to C\_11$ and $C2\underline{1} \to C\_11$ from the first part of (24), and all 9 such rules obtained from the second and third parts of (24).

$$\begin{array}{l} C\underline{2}1 \to C2\underline{1} \to D\underline{2}1 \to C\_11 \\ D0\underline{2} \to C\underline{0}1 \to C1\underline{1} \to D\underline{1}1 \to B2\underline{1} \to A\underline{2}1 \to A0\underline{1} \leftrightarrow D\underline{0}1 \\ A\underline{0}1 \to B2\underline{1} \to \ldots D\underline{0}1 \end{array} \tag{24}$$

The computer generated IRR of length n such that $3 \leq n \leq 6$ are as follows:

$$\begin{array}{lll}
C12\underline{1} \to D\_111(3) & C22\underline{1} \to C\_111(5) & \\
C112\underline{1} \to A0\underline{1}11(8) & C212\underline{1} \to C\_1111(4) & C022\underline{1} \to A0\underline{1}11(12) \\
C122\underline{1} \to D\_1111(6) & C222\underline{1} \to C\_1111(8) & C1212\underline{1} \to D\_11111(5) \\
C2212\underline{1} \to C\_11111(7) & C1122\underline{1} \to A0\underline{1}111(11) & C2122\underline{1} \to C\_11111(7) \\
C0222\underline{1} \to A0\underline{1}111(15) & C1222\underline{1} \to D\_11111(9) & C2222\underline{1} \to C\_11111(11) \\
C11212\underline{1} \to A0\underline{1}1^4(10) & C21212\underline{1} \to C\_1^6(6) & C02212\underline{1} \to A0\underline{1}1^4(14) \\
C12212\underline{1} \to D\_1^6(8) & C22212\underline{1} \to C\_1^6(10) & C12122\underline{1} \to D\_1^6(8) \\
C22122\underline{1} \to C\_1^6(10) & C11222\underline{1} \to A0\underline{1}1^4(14) & C21222\underline{1} \to C\_1^6(10) \\
C02222\underline{1} \to A0\underline{1}1^4(18) & C12222\underline{1} \to D\_1^6(12) & C22222\underline{1} \to C\_1^6(14)
\end{array} \tag{25}$$

of which I only verified those for $n = 3$ by hand.

The problem now as before is to conjecture and prove the general formulae for the IRR of length n. The first step is to conjecture the formula for the LHS's of these rules. Let X be a CS in the LHS of an IRR of length n and let X' be a CS in the LHS of a corresponding (in the sense of Theorem 9.1) IRR length $n + 1$, and let Y be any other string. Then it is easy to check, by three applications of it, that the following recursive definition defines the LHS's of the IRR up to length 6 (with the obvious definition of concatenated strings representing CS's) starting from those of length 3. This is therefore conjectured to hold up to any length $n > 3$.

$$\begin{array}{l}
\text{If the length of X is 3 then X} = 12\underline{1} \text{ or } 22\underline{1} \\
\text{For any string Ywith the pointer at its rightmost symbol :} \\
\text{If X} = 22\text{Y then X}' = 0\text{X or 1X or 2X} \\
\text{If X} = 12\text{Y then X}' = 1\text{X or 2X} \\
\text{If X} = 21\text{Y then X}' = 1\text{X or 2X}
\end{array} \tag{26}$$

Suppose X is a string satisfying 26 of length $n + 3$ then the conjectured IRR using these LHS's are as follows:

$$\begin{array}{ll}
C\left\{\begin{matrix}0\\1\\2\end{matrix}\right\} X \to \left\{\begin{matrix}A0\underline{1}\\D\_11\\C\_11\end{matrix}\right\} 1^{n+2} \text{ where X} = 22\ldots\underline{1} \text{ and } n \geq 0 & (1-3) \\
C\left\{\begin{matrix}1\\2\end{matrix}\right\} X \to \left\{\begin{matrix}A0\underline{1}\\C\_11\end{matrix}\right\} 1^{n+2} \text{ where X} = 12\ldots\underline{1} \text{ and } n \geq 0 & (4-5) \\
C\left\{\begin{matrix}1\\2\end{matrix}\right\} X \to \left\{\begin{matrix}D\_\\C\_\end{matrix}\right\} 1^{n+4} \text{ where X} = 21\ldots\underline{1} \text{ and } n \geq 1 & (6-7)
\end{array} \tag{27}$$

Suppose X' is a string one symbol longer than X i.e. with length $n + 4$ and satisfying 26 then X' can start with 02, 12, 22, 11, or 21. In the result whose proof is required, namely 27 with n increased by 1, i.e. with X replaced by X', the cases when X' starts with 02 and 11 are not included (because in 27 itself, X starts with only 12, 21, or 22). This leaves the cases $X' = 22Y$, $X' = 12Y$, and $X' = 21Y$. Then referring again to 26, X' must be one of the forms $122Y$ or $222Y$ with $22Y$ satisfying 26, or $212Y$ with $12Y$ satisfying 26, or finally $121Y$ or $221Y$ with $21Y$ satisfying 26. Now the RHS's corresponding to $C0X'$, $C1X'$ and $C2X'$ must be found. This leads to 15 computations of which those starting from $C0122Y$, $C0212Y$, and $C0121Y$ are not required because they cannot contribute to 27. The remaining 12 computations are as follows (repeated computations are indicated by ...):

$$\begin{array}{ll}
(1) & C0222Y \overset{27.3}{\to} C\underline{0}1^{n+4} \overset{24}{\to} D\underline{0}1^{n+4} \leftrightarrow A0\underline{1}1^{n+3} \\
(2) & C0221Y \overset{27.7}{\to} C\underline{0}1^{n+4} \ldots \to A0\underline{1}1^{n+3} \\
(3) & C1122Y \overset{27.2}{\to} D\underline{1}1^{n+4} \overset{24}{\to} A0\underline{1}1^{n+3} \leftrightarrow D\underline{0}1^{n+4} \\
(4) & C1222Y \overset{27.3}{\to} C\underline{1}1^{n+4} \overset{23.8}{\to} D\underline{\;}1^{n+5} \\
(5) & C1212Y \overset{27.5}{\to} C\underline{1}1^{n+4} \ldots \to D\underline{\;}1^{n+5} \\
(6) & C1121Y \overset{27.6}{\to} D\underline{1}1^{n+4} \ldots \to A0\underline{1}1^{n+3} \leftrightarrow D\underline{0}1^{n+4} \\
(7) & C1221Y \overset{27.7}{\to} C\underline{1}1^{n+4} \ldots \to D\underline{\;}1^{n+5} \\
(8) & C2122Y \overset{27.2}{\to} D\underline{2}1^{n+4} \overset{23.12}{\to} C\underline{\;}1^{n+5} \\
(9) & C2222Y \overset{27.3}{\to} C\underline{2}1^{n+4} \overset{24.1}{\to} C\underline{\;}1^{n+5} \\
(10) & C2212Y \overset{27.5}{\to} C\underline{2}1^{n+4} \ldots \to C\underline{\;}1^{n+5} \\
(11) & C2121Y \overset{27.6}{\to} D\underline{2}1^{n+4} \ldots \to C\underline{\;}1^{n+5} \\
(12) & C2221Y \overset{27.7}{\to} C\underline{2}1^{n+4} \ldots \to C\underline{\;}1^{n+5}
\end{array} \qquad (28)$$

This demonstrates (27) by induction and shows that all the rules in (27) are non-trivial in the sense that more than one derivation step is needed. Note that in this set of results, many of the calculations are duplicated in pairs with different symbols in the fourth place, and the result is the same. Another possible source of confusion is that because of the cycle of 2 at the end of (24).2, either of the results in the cycle can be taken as the endpoint of the calculation. Specifically, (27).1 corresponds to (28).1 and (28).2, (27).2 corresponds to (28).4 and (28).7 etc..

To demonstrate that (27) are all regular rules, note that except in the leftmost position in (27.1), the LHS's in (27) have strings that are only 1's and 2's. This is obvious by referring to the definition of X in (26). In this case the regularity of the rule follows from adding symbol 0 when 1 is to appear and 2 when 2 is to appear starting from either $C1\_$ or $C2\_$, which are both regular rules being RHS's of single TM steps in Equation 23. For the other case, the LHS of the rule starts $C022\ldots$. In this case the computation starting from $D0\_$ with 1 added, then 0 leads to $C022\_$. Then the above formula applies extending the CS showing that all CS's of the form $C022\{1,2\}^*\_$ are RHS's of RR. These arguments are trivial because in these cases the TM only moves to the right. Here $\{1,2\}^*$ is any string containing only 1 and 2.

		next symbol		
rule	CS	0	1	2
23.1, 23.11	$B2\_$	23.4	24.2	23.6
23.3, 23.10	$A0\_$	23.1	24.2	23.3
23.4, 23.9	$C2\_$	23.7	24.1	23.9
23.5	A_1	24.2	23.2	24.2
23.6	$D0\_$	23.10	23.11	24.2
23.7	$C1\_$	23.7	24.2	23.9
23.12, 24.1, 27.3, 27.5, 27.7	C_1	24.2	23.8	24.1
23.2, 23.8, 27.2, 27.6	D_1	24.2	24.2	23.12

Table 4: Table establishing the completeness of the IRR for TM5. The body of the table has the next rule to be applied.

The above results show that the rules in (27) are IRR. Table 4 outlines the demonstration of the completeness of the set of IRR, consisting of (23), (24) and (27) according to Theorem (5.6). The numbered IRR refer to equations 23 and (24) numbered consecutively, (in both cases numbers are not shown) and (27) numbered as shown. In this example there were a lot of cases that could be combined as indicated by the lists of rules in the first column and the CS reached in each case is a subset of a CS of length 1 that together with the new symbol determines the next IRR to be applied to continue the computation.

What is noticeable in Table 4 is that in no circumstances is any rule followed by a rule in (27). Therefore as this TM runs, there is never more than one execution of any of the rules in (27). If such a rule is executed, it must happen immediately. Also regarding the right-moving steps in 23, after more than 2 such steps they cannot be followed by a left-moving IRR. The exception is if 24 .1 is reached after 2 right-moving steps then the TM can cycle indefinitely though left-moving IRR including 24 .1. This analysis ignores the cycling condition 24.2 that can happen at many points and effectively stop the TM. Although the behaviours of this TM are fairly complex, there is nothing here similar to an interpretive cycle, so this TM is not expected to be universal in any sense.

10 Conclusions

Regular computation rules for any Turing Machine are defined so that, every computation can be expressed as a sequence of application of these short-cut computation rules such that a new rule is only applied when the pointer reaches a point on the TM tape that has never been reached before. Generally, the number of such rules very rapidly increases with the length of the rule (i.e. the length of the tape segment to which it applies). Many such rules are redundant in the sense that they are derived in one step from a shorter such rule. The non-redundant set are the irreducible regular rules (IRR) that are derived in more than one step. It was shown that the LHS's of the IRR have a tree structure because a truncation of such an LHS is also the LHS of an IRR of shorter length. By analysing a series of examples of TM's, it was found possible except in one complex case to establish recursive formulae that define all the IRR for the TM and draw conclusions about the behaviour of the TM. This was aided by Theorems 9.2 and 9.3, that allow an efficient automation of a procedure for obtaining all the IRR of a given length for any TM. It is not expected to be possible to obtain such a formula by a mechanical procedure in general from a TM, but once guessed correctly in a specific case such a formula can be verified by induction. These results give information describing the long-term behaviour of the TM for any input tape as the examples show. For this purpose a table can be constructed giving, for each IRR applied and each next symbol read, the corresponding next IRR to be applied.

Thus I conjecture that once the general formula for the IRR has been correctly guessed for a TM, it is possible to verify it with an induction argument. Once a TM has been analysed in this way it leads to a satisfactory higher level description of the computation that allows many questions to be answered for it i.e. the TM is presented such that as much general analysis as possible has been done for it. Nothing further can be said about it without the input tape being specified.

Possible applications are many because of the universal nature of Turing Machines. Theoretically any computational problem can be expressed as a (probably inefficient) Turing Machine, so analysis of a TM to solve the problem by this method could lead to information about the problem in general and perhaps to more efficient computational procedures for it.

Functions mapping the set of real numbers to itself could be set up as Turing Machines with the property that the IRR are all right-moving, or are so after some point. The examples in this paper show that this can happen. This property would ensure that a given degree of precision of the input (initial tape) would determine a corresponding degree of precision in the answer, provided the moving-window in which the computation occurs has a left end that advances to the right as the computation proceeds.

There seems to be a connection of this method with 'term-rewriting' [4], Gröbner Bases and the Knuth-Bendix procedure [5], because the general principle in the latter is to bring out the consequences of sets of algebraic relations in a systematic way to get new results that were not obvious initially, the result being an effective way to determine in certain cases whether an algebraic identity is true or not as a consequence of certain axioms. The procedure described in this paper carries out a similar function for the Turing Machine table to get the set of irreducible regular rules, that themselves define an equivalent form of the Turing Machine such that one step of this is needed only when a new tape location is read for the first time by the TM. The examples show that many properties of a computation can be found from the results of this generally applicable algorithm that were not obvious initially.

I think it is likely that there is also a connection with the methods I developed for solving systems of linear PDEs [6][7] with many independent and dependent variables involving the use of commutator brackets, where the same general principle applies.

References

[1] Marvin L. Minsky "Computation: Finite and Infinite Machines" Prentice-Hall, Inc, Englewood Cliffs, N.J. U.S.A., 1967.

[2] Nixon, John TM analyser A computer program written in C++ for the analysis of Turing Machines.

[3] Machlin R. and Stout Q.F. (1990) "The complex behavior of simple machines" Physica D 42, 85-98.

[4] Yap, Chee Keng , Introduction to Chapter 12 in "Fundamental problems in Algorithmic Algebra" Oxford University Press, 2000.

[5] Knuth D. E. and Bendix P. B. "Simple Word Problems in Universal Algebras." In Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967).
Pergamon Press, pp. 263-297, 1970.

[6] Nixon J., "Application of Lie groups and differentiable manifolds to general methods for simplifying systems of partial differential equations"
J. Phys. A: Math. Gen. 24 (1991) 2913-2941.

[7] Nixon J., "Minimization of dimension for functional differential equations and the thermodynamics of the classical one-dimensional fluid"
J. Phys. A Math. Gen. 27 (1994) 1407-1426.

Received: October 2013