

Survey Tutorial for *Deus ex machina* (Designed by Nicolae O. Savoiu)
Simulation Software Accompanying
R. Gregory Taylor, *Models of Computation and Formal Languages*
(Oxford University Press, 1998)

We consider a number of *models of computation*, where by a “model” one means an abstract machine perhaps that captures the essence of computation (cf. model airplane). Three computational paradigms can be identified: machines can *compute functions*, they can *accept (recognize)* formal languages, and, most generally, they can transform input symbol strings (transduction). In our brief survey of machine models, we shall focus on language recognition exclusively.

In particular, we shall concern ourselves with language

$$L = \{w \in \Sigma^* \mid n_a(w) = n_b(w)\}$$

where alphabet $\Sigma = \{a, b\}$. (This is just the language of all and only those words w containing just as and bs such that the number of as in w is equal to the number of bs in w .) So we have $abbbbabaaa \in L$ but $aba \notin L$. (Also, empty word $\epsilon \in L$.)

The first model of computation that we consider is the Turing machine model. We forego presentation of the notion of language acceptance on the part of Turing machine and move immediately to

Example 1.3.1 (“Same Number of as and bs ”). We are about to describe a Turing machine M that behaves as follows. Suppose that M 's tape initially contains a finite, unbroken string of as and bs and is otherwise blank. We designate this string of as and bs as w . We do not assume that the as and bs that make up w appear in any particular order within w . As usual, M will start scanning the leftmost symbol of w . M will perform in accordance with the following specification:

When starting scanning the leftmost symbol of word w consisting of an unbroken string of as and bs in any order, Turing machine M halts scanning a single l on an otherwise blank tape if and only if the number of as in w is equal to the number of bs in w , i.e., if and only if $n_a(w) = n_b(w)$.

Thus for initial tape configuration

$BabbbaaB$
 \uparrow

we require that M halt in configuration

$B1B$
 \uparrow

(We use \uparrow to indicate the position of the tape's read/write head.) Our task is now to design such an M . One rather straightforward solution is the following. We eliminate as and bs in pairs until either

- (1) every symbol has been eliminated, in which case the number of as was equal to the number of bs or
- (2) some symbol is eliminated for which no companion can be found, in which case the number of as was initially *not* equal to the number of bs .

To run this example, open icon **Same Number of a's and b's - single-tape machine** within the Turing group. From the File menu, choose Tape(s) and then Load, selecting either of the tapes 4a4b.tt or 5a4b.tt. With the read-head positioned over the leftmost nonblank, choose Run from within the Simulation menu.

Time analysis of Example 1.3.1: $O(n^2)$. Interpretation: if $length(w) = n$, then the number of steps, worst case, required for M to complete its computation is on the order of n^2 . Very roughly, processing a word of length 1024 will require about 1024^2 steps.

Turing machines may have several tapes. A multitape Turing machine that accepts L is given as

Example 2.3.1. Double-clicking on icon **Same Number of a's and b's - multitape machine** brings to the screen the state diagram of a four-tape Turing machine M that accepts the language $\{w \in \Sigma^* | n_a(w) = n_b(w)\}$ with $\Sigma = \{a, b\}$. Either of the tape sets 6a4bmult.tt or 9a9bmult.tt may be used with this machine. (Again, select File/Tape(s)/Load.) Since we have four read/write heads and four tapes, the arcs of M 's state diagram are labeled so as to give a current symbol/action pair for *each* of the four tapes. Since frequently only one of the four read/write heads will be moving or writing during a given time unit, it is convenient to introduce instructions such as $B:B$ or $a:a$ for the inactive head. M reads its input word w from left to right and, as it does so, copies any occurrence of a onto worktape₁ and any occurrence of b onto worktape₂. Afterward, M ascertains whether the number of as on worktape₁ is equal to the number of bs on worktape₂. A single I is written to the output tape just in case equality holds.

Our multitape machine does better, accepting L in $O(n)$ steps worst case.

An alternative model of computation is that of a so-called **Markov algorithm**. The Markov algorithm schema S associated with the icon labeled **Example 4.2.3**, within the Markov group, accepts the language L . Select any word within the Input Words clickbox. Then select Run/Apply Algorithm. The computation of S is operationally quite like that of the Turing machine M of Example 1.3.1 ("Same Number of as and bs "). Briefly, as and bs are erased in pairs from input word w until either nothing remains, in which case an accepting I is written, or just as or just bs remain, in which case no accepting I is written. (See the documentation accompanying S on disk by selecting Algorithm Description from within the File menu.)

As for a time analysis of S , considerations come into play that are perfectly analogous to those that arose with regard to M .

- In general, longer input words correspond to longer computations. Also, all else being equal, accepted words involve a greater number of computation steps than do nonaccepted words. In particular, input words of odd length may be disregarded altogether in giving a time analysis of S .
- Among accepted words of length n , the most costly will be $a^{n/2}b^{n/2}$ and $b^{n/2}a^{n/2}$. Moreover, the reader can use the software to verify that these words are accepted after $(n/2)^2 + n + 2$ computation steps. In other words, we have that $time_S(n) = (n/2)^2 + n + 2$ for even n so that $time_S(n)$ is $O(n^2)$.

The space analysis of S is easy enough. Examination of the productions of S —see the software—reveals that, with one exception, none are length-increasing: applied to a given computation word w , they each yield a new word w' whose length is less than or equal to that of w . The single exception is the physically last production $\varepsilon \rightarrow @$, which is applied precisely once to any input word. Moreover, since it is invariably applied first, it follows that $space_S(n) = n + 1$ so that $space_S(n)$ is $O(n)$.

So-called **register machines** were introduced by Shepherdson and Sturgis in the 1960s (see [2]).

Example 5.3.1. Our plan to represent an input word w over Σ as a sequence of 1s and 2s, one per tape square, terminated by end-of-input sentinel 0. If started scanning the leftmost square of its input tape, on which w is represented, M reads until sentinel 0 is encountered. Whenever a 1 is read, M increments register R_2 . Whenever 2 is read, M increments register R_3 . Afterward, M writes a 1 to the output tape just in case the contents of R_2 equals the contents of R_3 , i.e., just in case $w \in L$. By virtue of this, we shall say that M *accepts language* L .

Generally speaking, we shall not much care what a register machine does in response to an input word that is not in the accepted language—so long as it does not write an accepting 1. In fact, however, M does something very nice in that case as well; namely, M writes 0 to the output tape just in case the contents of R_2 do *not* equal the contents of R_3 , i.e., just in case input word $w \notin L$. And by virtue of this, we shall say that, in addition, M *recognizes language* L . As usual, our notion of language recognition amounts to computation of characteristic function χ_L defined as

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{if } w \notin L \end{cases}$$

Click twice on the icon labeled **Same Number of a's as b's** within the Register Machine group and load tapes 4a4b.rt and 5a3b.rt in succession in order to observe the operation of this machine. (Use File/Tape/Load and then Simulation/Run.)

Let us agree to count each primitive operation, *regardless of the size of its natural-number operand(s)*, as one step in M 's computation. In this context, this is called the *uniform cost assumption*. It is not hard to see that, for each symbol in input word w , M carries out exactly four steps as it traverses the loop at the top of the flowchart. Before entering that loop for the first time, there are two steps. Finally, after end-of-input sentinel 0 is encountered, exactly five steps are executed. In summary, where n is the length of input word w , register machine M executes $4n+7$ computation steps. Consequently, we shall say that M computes in linear time, i.e., in $O(n)$ steps worst case.

Finally, we consider one model of parallel computation, namely, the so-called vector machine model of Pratt and Stockmeyer from the 1970s (see [1]). Naturally, we expect there to be a vector machine M that accepts L faster even than the multitape Turing machine or the register machine.

Example 7.5.3 (Same Number of a s and b s). Suppose that input word w over $\Sigma = \{a, b\}$ is of even length. Then w consists of $|w|/2$ symbol pairs. Each of these pairs is either aa , ab , ba , or bb . Now suppose that all of the aa -pairs are eliminated obtaining w_1 . Similarly, suppose that the result of eliminating all bb -pairs from w is a new word w_2 . Then it is evident that $\text{length}(w_1) = \text{length}(w_2)$ just in case $n_a(w) = n_b(w)$. (Try it.) Pseudocode describing a vector machine M that recognizes L appears below. (See also icon **Same Number of a's and b's** within the Vector Machine group. Load vector sets `samenum*.vs` and `diffnum*.vs` in succession, using `File/Vector Sets/Load`.)

Input

$$V_1 = +\text{Trans}(w) \quad \{\text{For example, if } w \text{ is } baabaabb, \text{ then } V_1 \text{ will be } +10010011.\}$$

$$V_2 = U(|w|) = +I^{2^k} \quad \{\text{Input } V_2 \text{ is needed since, in general, } |\text{Trans}(w)| \neq |\text{Trans}(w^{-1})|.\}$$

Output

$$V_3 = \begin{cases} +1 & \text{if } n_a(w) = n_b(w) \\ +0 & \text{otherwise} \end{cases}$$

We are writing $(baabaabb)^{-1}$ for $abbabbaa$ —the result of simultaneously replacing all a s with b s and all b s with a s.

Algorithm

- (1) begin
- (2) $V_4 := V_1 = +\text{Trans}(w)$
- (3) $V_5 := \neg V_1 \& V_2 = +\text{Trans}(w^{-1})$;
- (4) delete from V_4 all 00 -pairs from within $\text{Trans}(w)$; {in $O(\log_2^3 n)$ steps}
- (5) delete from V_5 all 00 -pairs from within $\text{Trans}(w^{-1})$; {in $O(\log_2^3 n)$ steps}
- (6) if pair-length of $V_4 =$ pair-length of V_5 {There are several cases to consider.}
- (7) then $V_3 := +1$
- (8) else $V_3 := +0$
- (9) end.

Time Analysis: $O(\log_2^3 n)$ steps worst case. Interpretation: M 's computation for input word w of length 1024 will require something on the order of 10^3 steps worst case.

Finally, the four computational models considered can be related by virtue of the following

Theorem. The following are all equivalent for any given language L over alphabet Σ :

- L is accepted by some single-tape Turing machine.
- L is accepted by some multitape Turing machine.
- L is accepted by some Markov algorithm.
- L is accepted by some register machine.
- L is accepted by some vector machine.

Software. The demonstrated software was designed by Nicolae O. Savoiu and is available on the Internet. To download this software, point your WWW browser to <http://www.ics.uci.edu/~savoiu/dem> and follow the on-line instructions. Help files are available at the same site.

On-Line Resources. The website for the textbook is found at <http://starbase.cs.trincoll.edu/~rtaylor/thcomp/>, wherein instructor's guide, solutions manual, errata list, and so forth may be found.

- [1] Pratt, Vaughn R., and Stockmeyer, Larry J., "A Characterization of the Power of Vector Machines," *Journal of Computer and System Sciences* 12 (1976) 198–221.
- [2] Shepherdson, J. C., and Sturgis, H. E., "Computability of Recursive Functions," *Journal of the Association of Computing Machinery* 10 (1963) 217–255.