Theory of Computation

MIEIC, 2nd Year

João M. P. Cardoso

Dep. de Engenharia Informática
Faculdade de Engenharia (FEUP)
Universidade do Porto
Porto
Portugal

Email: jmpc@acm.org

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Outline

- Motivation
- Context
- The Church-Turing Thesis
- Turing Machine (TM)
- Languages of a TM
- Techniques to program a TM
- Extensions to TMs
- Summary

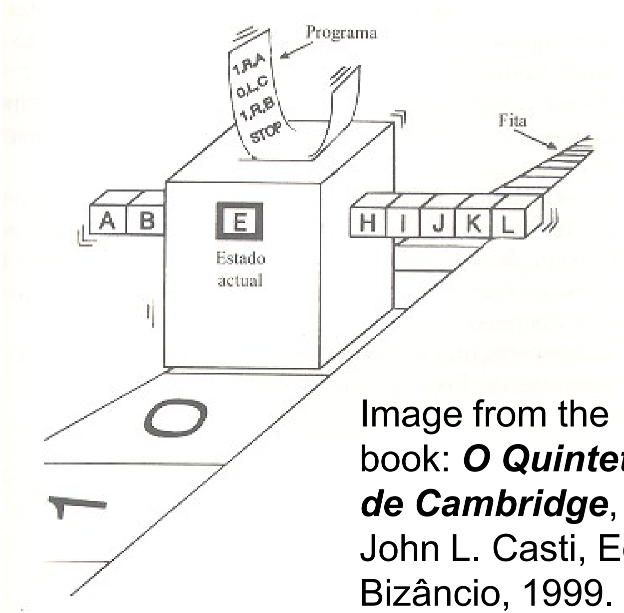


Image from the book: ***O Quinteto de Cambridge***, John L. Casti, Ed. Bizâncio, 1999.

Motivation

- Undecidable Problems
  - There is no algorithm
- Intractable Problems
  - The known algorithms are very costly
  - Simplification and the use of heuristics
- Simple model to study the computability
  - Turing Machines
  - A model of a computer

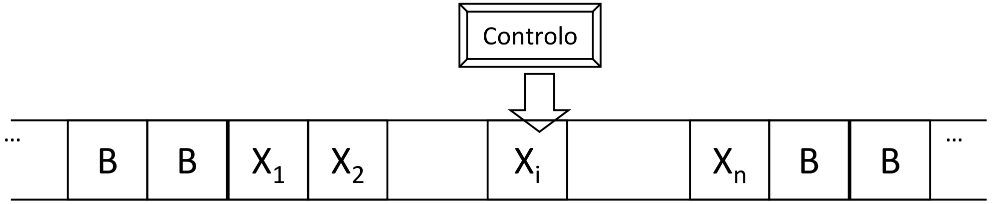
Context

- David Hilbert (beginning of 20th century)
  - Is there a way to determine whether any formula in the first-order predicate calculus, applied to integers, is true?
- Kurt Gödel (1931)
  - Incompleteness theorem. He constructed a formula in the predicate calculus applied to integers, which asserted that the formula itself could be neither proved nor disproved within the predicate calculus.
- Alan Turing (1936)
  - proposed the Turing machine as a model of any possible computation
- A. Church
  - Church-Turing hypothesis (unprovable)
  - “any general way to compute will allow us to compute only the partial-recursive functions” (or equivalently to what Turing machines or modern-day computers can compute)

The Church-Turing Thesis

- *“All reasonable models of (general-purpose) computers are equivalent. In particular, they are equivalent to a Turing machine.”*
- A Turing Machine is, according to the Church-Turing thesis, a general model of computation, potentially able to execute any algorithm.
- First formulated by Alonzo Church in the 1930s and is usually referred to as Church’s thesis, or the Church-Turing thesis

Turing Machine



- Finite State Controller
  - Finite number of states
- Tape with infinite length and consisting of cells
  - Each cell can store a symbol
- Input
  - Finite string consisting of symbols of the input alphabet
  - Placed in the tape in the beginning (all other cells are marked with blank (B))
- Symbols in the tape
  - Input alphabet + blank + other symbols if needed

Turing Machine

- Head of the tape
  - Always positioned in a cell
  - In the beginning it is in the leftmost cell of the input string
- Movement or step of the machine
  - Function of the state of the control and of the symbol being read by the head
  - 1. State transition
    - It can be the same
  - 2. Write of a symbol in the cell where is the head
    - It can be the same
  - 3. Movement of the head by one cell left or right
    - A transition always implies a step of the head in the tape

Formal Definition

- Turing Machine (TM) M= (Q, Σ, Γ, δ, q_0, B, F)
  - Q: Finite set with the control states
  - Σ: Finite set of the input symbols
  - Γ: Finite set of symbols in the tape
  - δ: Transition function δ(q, X) = (p,Y,D)
    - q is a state, X is a symbol in the tape
    - p is the next state (in Q);
    - Y is a symbol in Γ which substitutes X;
    - D is L or R, left or right ($\leftarrow$ or $\rightarrow$), direction in the movement of the head after the substitution of the symbol in the tape
  - q_0: initial state
  - B: blank, a symbol to represent the blank symbol and the tape is fully filled (excepting the cells with the input string) with this symbol
  - F: Set of final or accept states, F $\subseteq$ Q

Computations

- Instantaneous descriptions
  - $X_1X_2 \ldots X_{i-1}qX_iX_{i+1} \ldots X_n$
  - Cells from the first non-blank to the last non-blank (finite number)
    - With blank suffixes and prefixes depending where the head is
  - The state (q) and the cell (i) where the head is
- Step of the Turing Machine M ($\vdash_M$; $\vdash^*_M$ – 0 or more steps)
  - Supposing $\delta(q,X_i) = (p,Y,L)$
  - $X_1X_2 \ldots X_{i-1}qX_iX_{i+1} \ldots X_n \vdash_M X_1X_2 \ldots X_{i-2}pX_{i-1}YX_{i+1} \ldots X_n$
    - Transition from q to p; X_i cell is changed to Y; head moves left
    - If i=1: $qX_1X_2 \ldots X_n \vdash pBYX_2 \ldots X_n$
    - If i=n and Y=B: $X_1X_2 \ldots X_{n-1}qX_n \vdash X_1X_2 \ldots X_{n-2}pX_{n-1}$
    - Symmetric for $\delta(q,X_i) = (p,Y,R)$

Example 0^n1^n

- TM to accept the language $\{0^n1^n \mid n \geq 1\}$
- Idea
  - In the beginning the tape contains the input string (0s and 1s)
  - Change the first 0 to X; move to right until the first 1 and change it to Y; move to left until the first X; move o right; repeat
  - If in a state there is a symbol in the tape not expected the TM dies
    - If the input is not 0^n1^n
  - If in the iteration that marks the last 0 it also marks the last 1 then it accepts

Example 0^n1^n

State	0	1	X	Y	B
q_0	(q_1,X,R)			(q_3,Y,R)	
q_1	$(q_1,0,R)$	(q_2,Y,L)		(q_1,Y,R)	
q_2	$(q_2,0,L)$		(q_0,X,R)	(q_2,Y,L)	
q_3				(q_3,Y,R)	(q_4,B,R)
q_4					

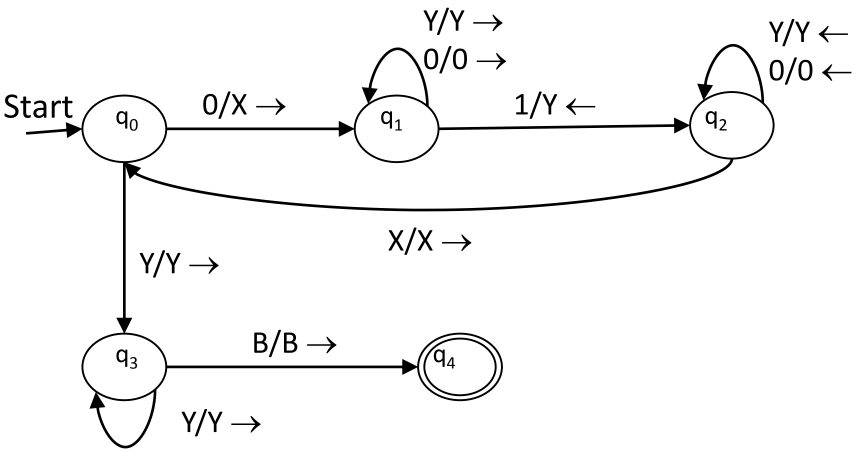
- $M = (\{q_0,q_1,q_2,q_3,q_4), \{0,1\}, \{0,1,X,Y,B\}, \delta, q_0, B, \{q_4\})$
  - q_0: changes 0 to X
  - q_1: moves to right until the first 1 and it is changed to Y
  - q_2: moves to the left until it finds an X and goes to q_0
  - If it has a 0 reinitiates the loop; if it has a Y goes to right; if it finds a blank goes to q_4 and accepts; otherwise dies without accepting

Computations for the Previous Example

- Input 0011
  - Initial instantaneous description: $q_0 0011$
  - $q_0 0011 \vdash Xq_1 011 \vdash X0q_1 11 \vdash Xq_2 0Y1 \vdash q_2 X0Y1 \vdash$
  - $Xq_0 0Y1 \vdash XXq_1 Y1 \vdash XXYq_1 1 \vdash XXq_2 YY \vdash Xq_2 XYY \vdash$
  - $XXq_0 YY \vdash XXYq_3 Y \vdash XXYYq_3 B \vdash XXYYBq_4 B$
    - accepts
- Input 0010
  - $q_0 0010 \vdash Xq_1 010 \vdash X0q_1 10 \vdash Xq_2 0Y0 \vdash q_2 X0Y0 \vdash$
  - $Xq_0 0Y0 \vdash XXq_1 Y0 \vdash XXYq_1 0 \vdash XXY0q_1 B$
    - dies

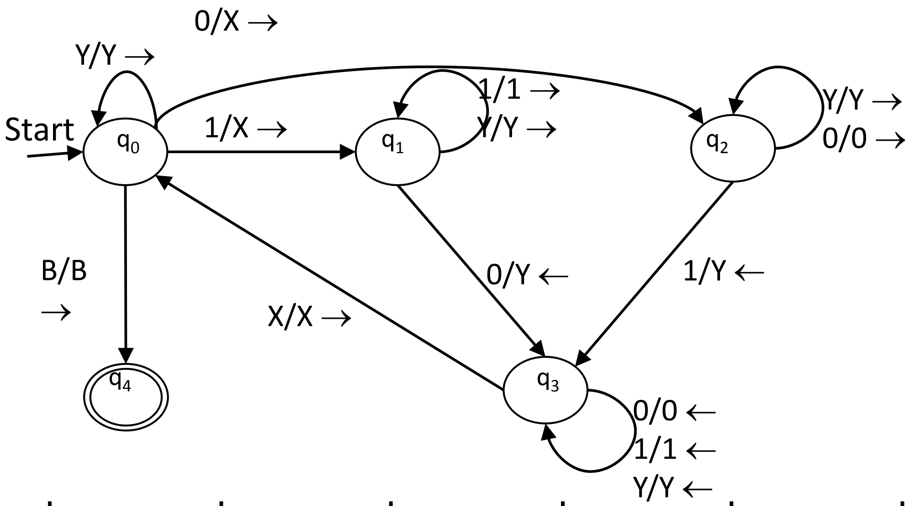
Transition Diagram

- Similar to PDA
  - Nodes are TM states
  - Edge from q state to p state with label X/YD
    - $\delta(q,X) = (p,Y,D)$, X and Y are symbols in the tape and D is L or R ($\leftarrow$ or $\rightarrow$)
  - "Start" arrow indicates the initial state; double circle represents final states; B, blank



Example

- Transition diagram for a TM which accepts the language of the strings with equal number of 0s and 1s



- $q_0 0110 \vdash Xq_2 110 \vdash q_3 XY10 \vdash Xq_0 Y10 \vdash XYq_0 10 \vdash XYXq_1 0 \vdash XYq_3 XY \vdash XYXq_0 Y \vdash XYXYq_0 B \vdash XYXYBq_4 B$
- $q_0 110 \vdash Xq_1 10 \vdash X1q_1 0 \vdash Xq_3 1Y \vdash q_3 X1Y \vdash Xq_0 1Y \vdash XXq_1 Y \vdash XXYq_1 B$

Example (cont.)

- Basic idea of the behavior of the MT
  - Identify in the tape 0-1 or 1-0 pairs, marking with X the first element and with Y the second element, until cannot find another pair (the case where the TM dies) or find blank (the case where the TM accepts)
- Meaning of the states
  - q_0: goes to right until it finds the first element of the next pair; if it finds a blank goes to accept state
  - q_1: found 1; goes to right until it finds a 0 or dies if it finds a blank;
  - q_2: found 0; goes to right until it finds a 1 or dies if it finds a blank;
  - q_3: found the second element of a pair; goes back to the left until it finds the rightmost X, case in which it transits to q_0;
  - q_4: accept.

Exercise 1

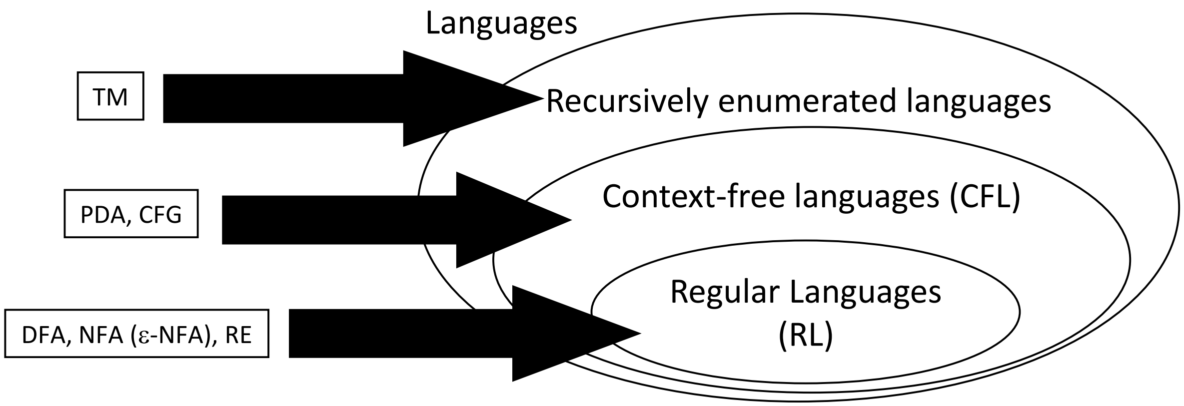
- Design a Turing Machine able to convert an input binary number to its two’s complement representation. In case of overflow, i.e., if the operation produces a number with more bits than the ones in the input number, that extra bit is ignored, maintaining the same number of bits.
  - Describe the meaning of each state.
  - Show the Computing trace when the input is 100.

Exercise 2

- Design a Turing Machine able to decrement by 1 an input binary number.
  - What are the strings accepted by the machine?
  - Where is the head of the machine in the end of the processing?
  - Show the sequence of instantaneous descriptions of the machine when it processes the input 100.

Language of a TM

- Input string placed in the tape
  - Head of the machine in the leftmost symbol of the input
- If the machine stops in an accept states, the input string is accepted
- Language of the TM M= $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$
  - Set of strings w in Σ^* such that $q_0 w \vdash^* \alpha p \beta$ and $p \in F$
  - <u>Recursively Enumerated Languages (Turing-recognizable)</u>



Stop

- A TM stops if it enters in a state q, reads a symbol X and $\delta(q,X)$ is not defined
  - It allows to approach a TM as executing a computation with start and end
    - example: calculate the sum of two integers
  - TMs that always stop, accepting or not the input, constitute models of algorithms (recursive languages)
- We can assume that a TM always stops when accepts
- Unfortunately it is not always possible to enforce that a TM stops when it does not accept (the halting problem)
  - Undecidability (recursively enumerated languages)
  - Possibility of a TM to refer to itself (it can be undecidable)

Techniques to program a TM

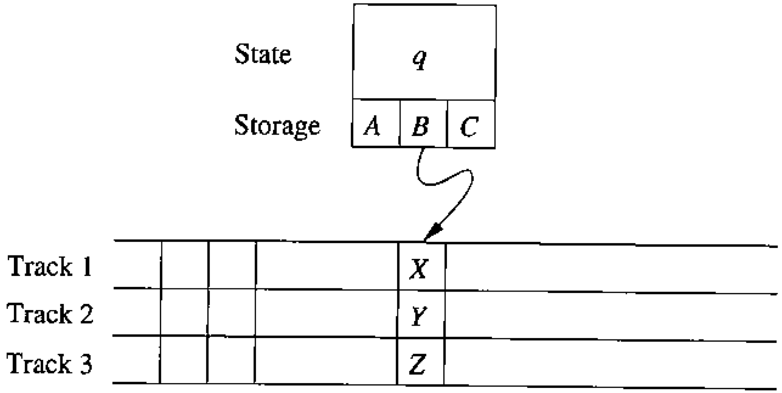
- A TM is as powerful as any actual computer (but too slow)
- Memory in the state
  - State = control + data memory
  - See the state as a tuple
- Additional symbols used in the tape can make easier the programming of the TM

Techniques to program a TM: subroutines

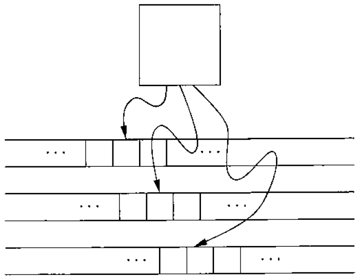
- A TM is a set of states which executes a process
  - It as an input and final states
- Seen as a subroutine of a major TM
  - Call goes to initial state
  - There is not notion of a return address
    - Return is done by using a state
  - If a subroutine is called from various distinct states we copy it (like a macro) to return to the states where was called

Extensions to TMs

- Multiple tracks in a tape
  - A tape consists now by a finite number of tracks: a symbol in each track
  - Alphabet in tape consists of a tuple
- TM power is not changed



Extensions



- TM with various tapes
  - Each one has a head
  - Input string only in the first tape
  - Movement: left, right, stationary
  - Equivalent to one tape
    - Quadratic temporal complexity
- Non-Deterministic TMs
  - Transition function give set of tuples (q,Y,D)
  - Equivalent to deterministic
    - Place in the tape a queue with the instantaneous descriptions to be processed
    - Simulate the non-determinism traversing them by breath first order
- They do not empower in terms of language recognition

Restrictions

- Machines with several stacks
  - A PDA with two stacks is equivalent to a Turing Machine
    - In the first stack we store what is in the left of the head of the TM
    - In the second stack we store what is on the right of the TM (the top contains the current symbol read by the head)
    - In the controller we simulate the control of the TM
    - Movement of the head of the TM is to do a pop in a stack and a push in the other

Restrictions (cont.)

- Machines with counters
  - The same structure of a machine with multiples stacks
  - Each stack is a counter
    - Contains a non-negative integer (number of X in stack)
    - We only distinguish if the counter is 0 or different of 0
  - Movement depends
    - State
    - Input symbol
    - Each one of the counters is 0
  - In the movement
    - Change state (or stay in the same)
    - Adds or subtracts 1 to each counter independently

Power of the machines with counters

- Theorem: the machines with 3 counters are equivalent to the TM, i.e., they accept the recursively enumerated languages
- A counter simulates a stack
  - A stack $X_1X_2\ldots X_n$ in an alphabet with r-1 symbols can be seen as a number in base r $X_nr^{n-1}+X_{n-1}r^{n-2}+\ldots+X_2r+X_1$
  - Pop is to divide by r and remove the remainder (X_1)
- Two counters for 2 stacks ($\equiv$ TM) and another one for the multiplication and division operations
- Theorem: the machines with 2 counters are equivalent to the TM
  - Code the 3 counters i, j, k in a counter $2^i3^j5^k$ (2,3,5 are primes)
  - Second counter for operations

TMs and computers

- A Computer is able to simulate a TM
  - The number of states and transitions is finite: states represented by strings and table of transitions
  - Number of symbols in the tape is finite: strings of fixed length
  - Tape infinite!
    - If the memory of the Computer is finite, this is a finite automaton
      - Only regular languages!
    - Assuming the capability to change disks: it is reasonable to consider infinite memory
      - Stack of disks in the left and in the right of the tape: disk in use according to the position of the head
      - Recursively enumerable language

TMs and Computers (cont.)

- Simulate a Computer with a TM
  - memory: long sequence with words with an address
  - Program stored in memory
    - Simple instructions, assembly
    - Consider indirect addressing
  - Each instruction uses a limited number of words and changes one word at the maximum
  - Registers of the Computer considered as more memory
- TM with multiple tapes simulates a computer ($\equiv$ TM with 1 tape)
  - Memory (address-value pairs) + Instruction counter + memory address + Input file + area of temporaries
  - Simulate the instruction cycle; copy, sum, jump, …

The multi-tape TM simulates n steps of a Computer in $O(n^3)$ of its own steps; and a TM simulates n steps of a Computer in $O(n^6)$ TM steps [See Hopcroft book]

Summary

- The TM is a valid representation of what a Computer can compute
- It is realistic to consider the TM as equivalent to a Computer
  - Relation between execution times is polynomial
  - The division between tractable and intractable problems is between the polynomial and greater than polynomial complexity
  - It allows to study the efficiency of the algorithms in the TM and not only the decidability