TATA CONSULTANCY SERVICES

TATA

Computing: 50 Years On

Mathai Joseph

TIFR Alumni Association Lecture, 27 July 2007.

1957

Many reasons to remember 1957: e.g.

- *Sputnik* satellite launched from Baikonur USSR
- Nobel Physics prize to Yann Chen Ning (China, IAS Princteon) & Lee Tsung-Dao (China, Columbia) for work on parity laws

In India:

- Nuclear reactor *Apsara* in operation (went critical in August 1956)

1957

Many reasons to remember 1957: e.g.

- *Sputnik* satellite launched from Baikonur USSR
- Nobel Physics prize to Yann Chen Ning (China, IAS Princteon) & Lee Tsung-Dao (China, Columbia) for work on parity laws

In India:

- Nuclear reactor *Apsara* in operation (went critical in August 1956)

And

- Bobby Fischer became US chess champion at age 14!

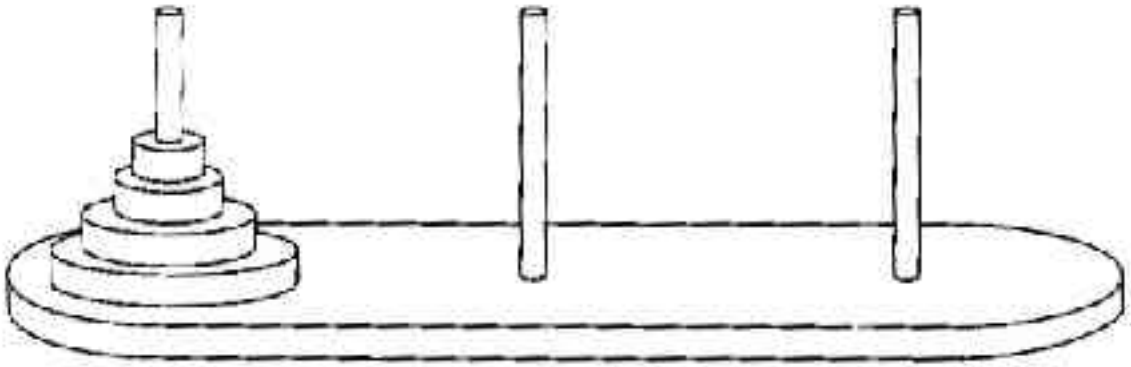
Computing in 1957

What was happening in computing?

- 1000 computers sold worldwide *(none in India)*
- New computer series announced (mostly solid-state)
  - IBM, Univac, NCR, Siemens. …
- Integrated circuit (IC) development near completion
  - Jack Kilby at Texas Instruments, Robert Noyce at Fairchild
- Fortran 1 programming language formally published
  - Team led by John Backus
- First attempt at automated proof by computer
  - Simon, Newell, Shaw devise *General Purpose Solver* (GPS)

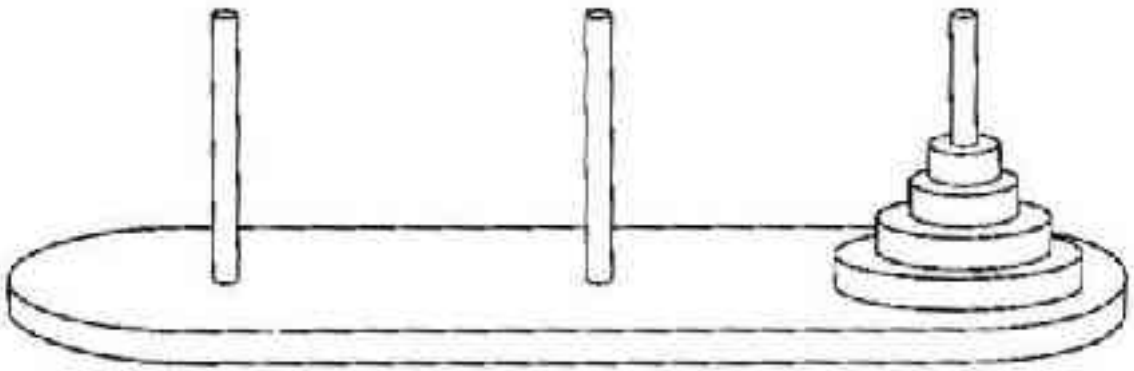
General Problem Solver

Used to give an automated solution to the *Towers of Hanoi* problem:



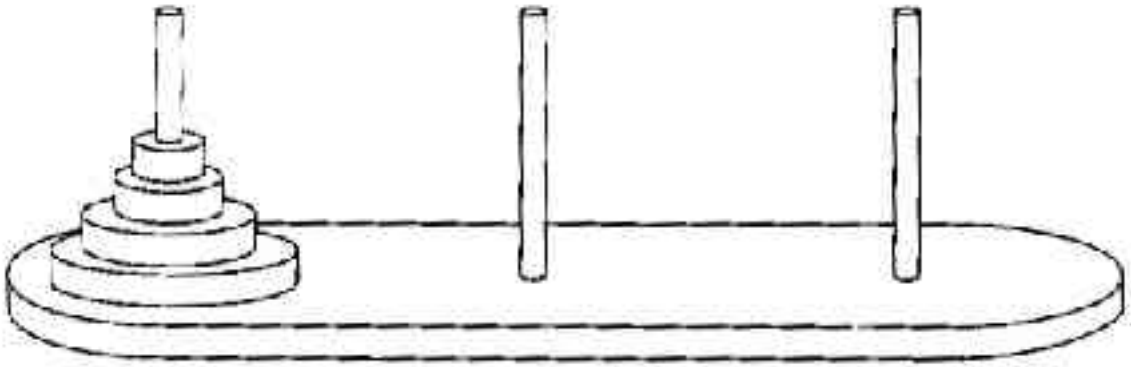
General Problem Solver

Used to give an automated solution to the *Towers of Hanoi* problem:



General Problem Solver

Used to give an automated solution to the *Towers of Hanoi* problem:



GPS worked by Means-Ends analysis:

- *Where are you now? What is the goal?*
- *Which operations can take you to the goal?*

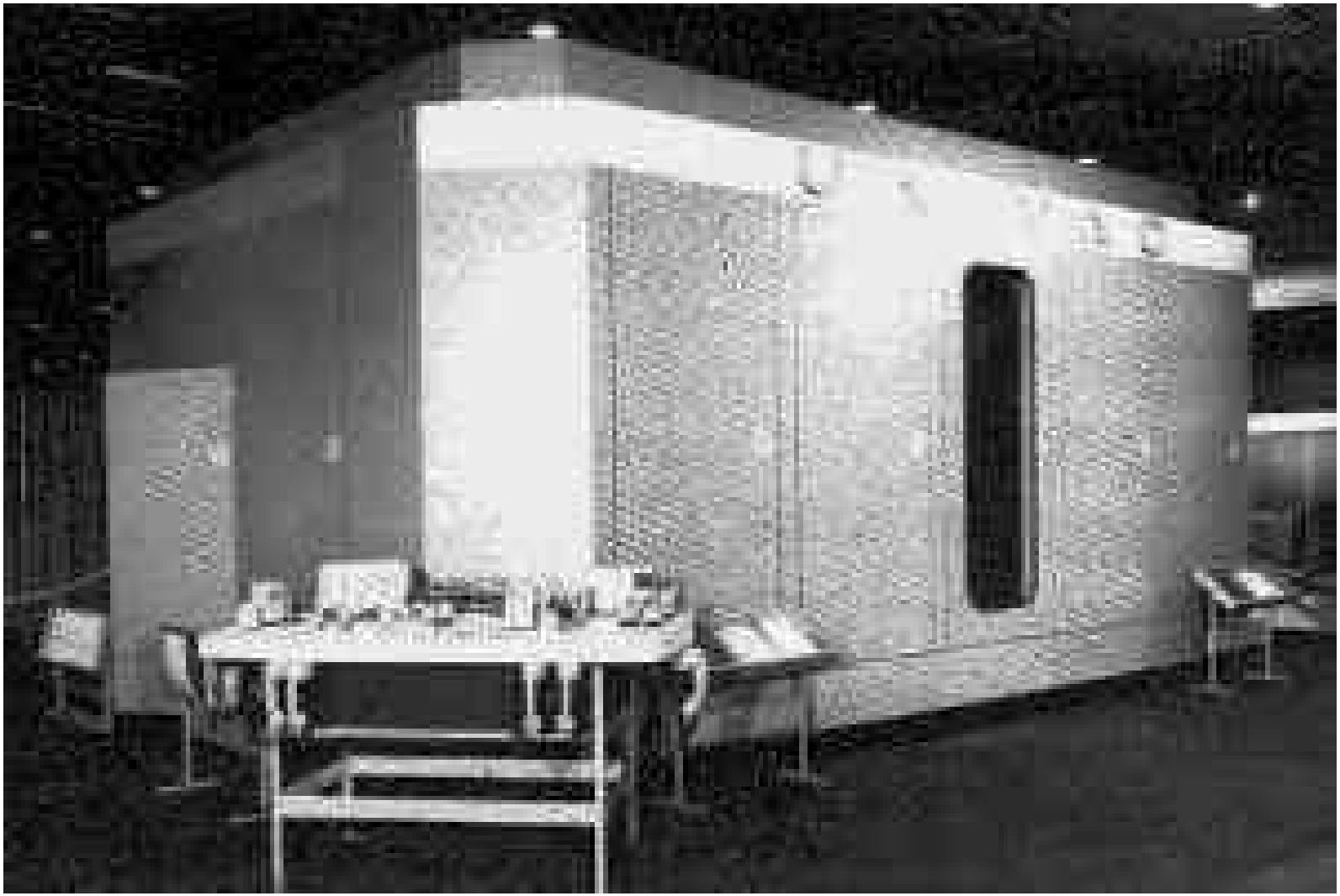
Top US university computing

1957: Univac 1 gifted to University of Pennsylvania

- 1000 vacuum tubes
- acoustic delay line memory (1000 words of 12 characters)
- decimal computation

University of Pennsylvania famous for construction of:

- ENIAC 1943-46: first large-scale computer
- EDVAC 1949: first US stored program computer



UNIVAC 1 processor and memory

Computing in India 1957

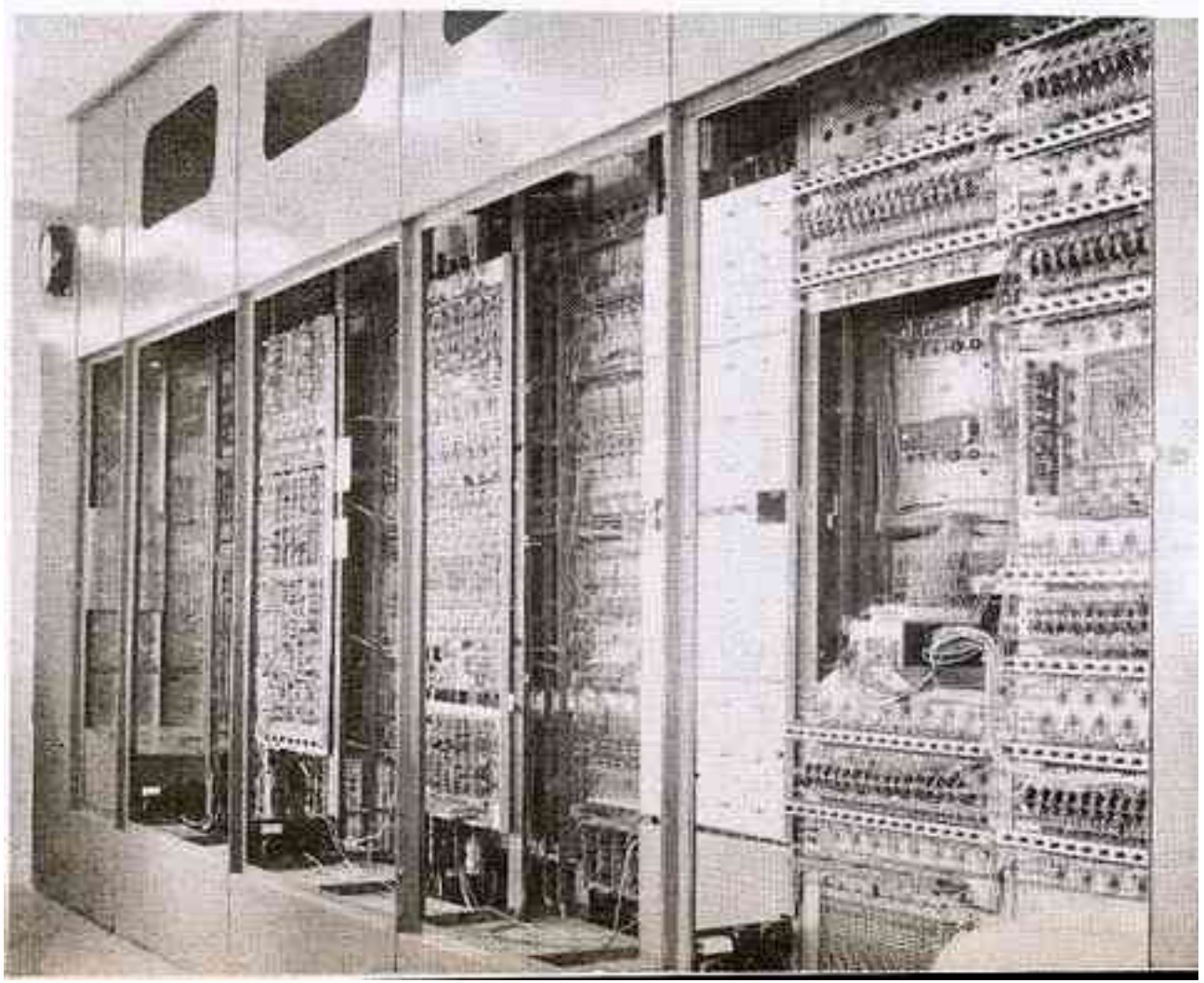
India takes first steps towards digital computing:

- Follows pilot study from 1954-1956

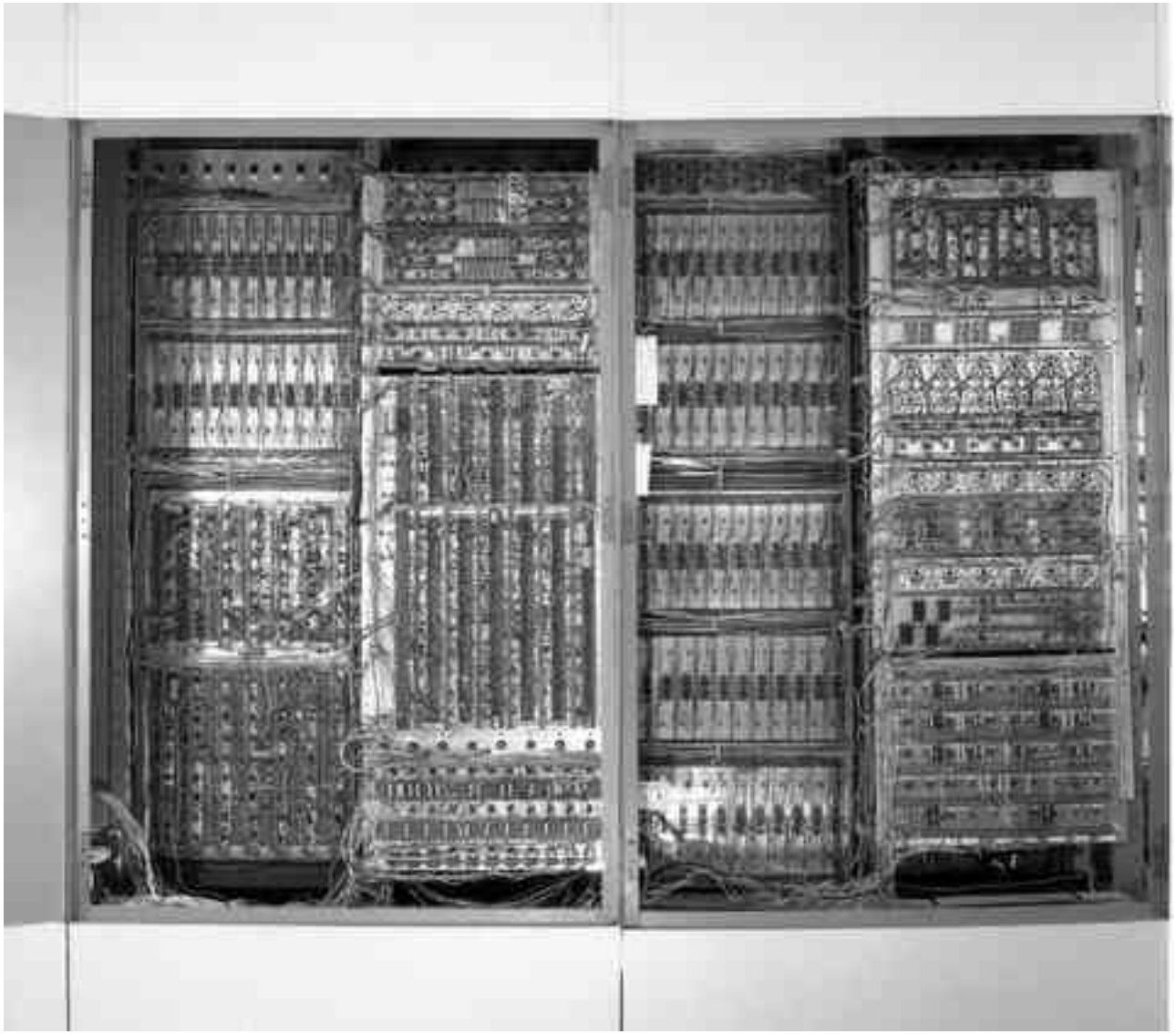
TIFR Automatic Computer (TIFRAC) construction starts:

- Enormous challenge for Indian electronics engineering
  - 2700 vacuum tubes, 1700 germanium diodes
- Many innovations
  - 2048 words of ferrite core memory
- First attempt at system programming in India

TIFRAC was completed in 1960.



TIFRAC Photograph courtesy TIFR Archives



TIFRAC Photograph courtesy TIFR Archives

Early 1960's in India

TIFRAC introduced computing to many Indian scientists:

"The first assignment given to me was to work with TIFRAC computer team. TIFRAC was at that time under development ..."

(Former President APJ Kalam, Address to CHEP06)

TIFRAC was used by scientists from TIFR, e.g.

*"thanks are due to R. Subramanian for helping us in programming the calculations for the TIFRAC" (*SK Bhattacharya, SK Mitra*, Phys. Rev.*1962*)*

… and other institutions, e.g. BHU

"... a similar work has been reported by carrying out the calculations with the TIFRAC …" (ML Rustgi, SN Mukherjee, *Phys. Rev*. 1963)

Elsewhere, by the 1960s

Major impact of IBM 360 series of computers:

- Wide range of commercial & scientific computers
- First models used hybrid solid-state + IC technology
- By mid-1960's, all modules used only IC
- Wide commercial and educational use in US

Similar systems manufactured in US, Europe.

Other systems bringing new forms of use:

- mini-computers
- time-sharing computers
- high-performance computers

Spread of computing across developed world.

1960's in India

Commercial computing started:

- Esso installed IBM1401 at Backbay office
- First of similar computers installed in other companies

1962: first Indian transistorized computer built by *Indian Statistical Institute* and *Jadavpur University* – ISIJU.

Large scientific computers installed:

- CDC3600 at TIFR, IBM7044 at IIT Kanpur …

First Indian software consultancy created:

- Tata Consultancy Services formed in 1968

Computer Science in the 1960's

Enormously formative time:

- Theory of algorithms, complexity theory
- Start of program verification, formal techniques for programming
  - Attempts at automatic program verification (following GPS)
- Major developments in artificial intelligence

  "Russian computer chess program will beat world champion"
  (1963 World chess champion Botvinnik).
  - Concerted focus on automatic language translation
  - First steps towards automatic speech recognition
- Experimentation with:
  - Parallel computing
  - Multi-access & time-sharing

…

And in India …

Many years before the new face of computer science was reflected in India:

- acute shortage of computers in India
- 'traditional' focus on numerical methods etc.
- but areas like formal language theory grew

***Cannot underestimate the effect of lack of computers**:*

- *like doing natural science without experimental facilities!*

And in India …

Many years before the new face of computer science was reflected in India

- acute shortage of computers in India
- ‘traditional’ focus on numerical methods etc.
- but areas like formal language theory grew

Cannot underestimate the effect of lack of computers

- *like doing natural science without experimental facilities!*

1960 – 75 was a quiet period for computer science in India

Computer science without computers?

Alan Turing defined fundamentals in computer science before any computers existed

Turing led the way in many areas:

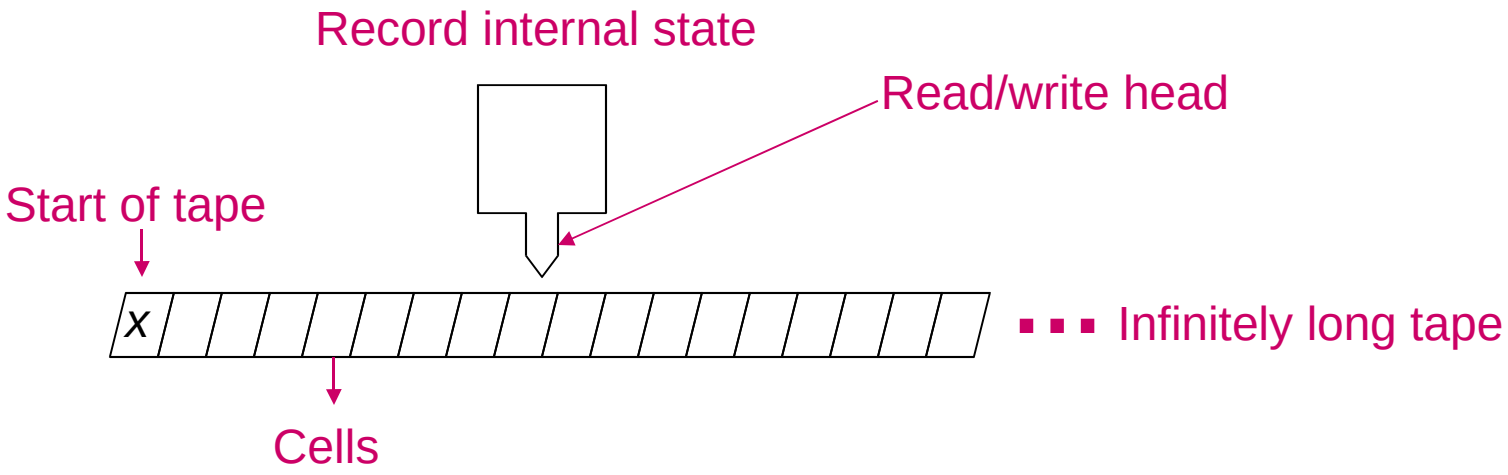
- invented concept of the universal computing machine
- defined computability
- proved basic decidability results
- worked on formal program proof, artificial intelligence, chess programs, neural nets, statistical methods for code breaking, …

Turing also ran a full marathon in 2hr 46min 3sec (just 11 sec slower than the Olympic winner of the time).

Computability

Turing machines (1936):

- Abstract computation device
  - Machine has an internal state
  - Head can read or write one cell
  - Tape can move one cell left or right
  - Next state & action depend on current state & tape symbol



Turing used this idea to prove basic properties:

- computability,

  Any problem that is computable, can be computed by a Turing Machine

- decidability, etc.

  Example: ***Halting Problem***

  Given any program and an input to the program,
  will the program will eventually stop when it is given that input?

Example of an *undecidable* problem.

Nondeterministic Turing machines:

  – For a given symbol and state, more than one action possible.

Algorithms, complexity

Known from the time of the Greeks:

- e.g. Euclid's algorithm for GCD (350B.C.)

Lame's Theorem (1845)

- Euclid's algorithm takes at most *4.2 log(n) / log (10) -0.32* steps

Long history of mathematical interest in complexity:

- Cantor, Hilbert, Pocklington, Post, Church, Gödel, Turing, …

Given a new basis in the 1960's:

- Yamaha, Myhill, Smullyan, Cobham, Edmonds …

Algorithms, complexity

Known from the time of the Greeks:

- e.g. Euclid's algorithm for GCD (350B.C.)

Lame's Theorem (1845)

- Euclid's algorithm takes at most *4.2 log(n) / log (10) -0.32* steps

Long history of mathematical interest in complexity:

- Cantor, Hilbert, Pocklington, Post, Church, Gödel, …

Given a new basis in the 1960's:

- Yamaha, Myhill, Smullyan, Cobham, Edmonds

Importance of non-determinism:

- unique to computer science

1960's: Complexity

Can all computable problems be solved equally easily?

Hartmanis & Stearns(1965): quantified time & space of a computation.

- Time: number of steps that the tape moves
- Space: number of cells of the tape that are used

1960's: Complexity

Can all computable problems be solved equally easily?

Hartmanis* & Stearns(1965): quantified time & space of a computation.

- Time: number of steps that the tape moves
- Space: number of cells of the tape that are used

*Juris Hartmanis lectured on switching theory at TIFR in 1966.

Complexity

Can all computable problems be solved equally easily?

Hartmanis* & Stearns(1965): quantified time & space of a computation.

- Time: number of steps that the tape moves
- Space: number of cells of the tape that are used

Later developed into decision problems about classes of algorithms, e.g.

- P
  - Class of polynomial time algorithms
- NP
  - Class of nondeterministic polynomial time algorithms

Example of P

Sorting N items can be done in time that is a polynomial function of N.

- Given 17, 101, 2, 31, 7, 19, 52, 1, 91
- Sort to 1, 2, 7, 17, 19, 31, 52, 91, 101

Example of P

Sorting N items can be done in time that is a polynomial function of N.

- Given 17, 101, 2, 31, 7, 19, 52, 1, 91
- Sort to 1, 2, 7, 17, 19, 31, 52, 91, 101

Or

- Given efg, bc, de, aa, bcc, ee, ab, ef, b
- Sort to ?

Example of P

Sorting N items can be done in time that is a polynomial function of N.

- Given 17, 101, 2, 31, 7, 19, 52, 1, 91
- Sort to 1, 2, 7, 17, 19, 31, 52, 91, 101

Or

- Given efg, bc, de, aa, bcc, ee, ab, ef, b
- Sort to aa, ab, b, bc, bcc, de, ee, ef, efg

Sorting

"*Sorting is in P*"

- Different sorting algorithms may take different times
- Comparing each number with the remaining numbers during sorting will on average take of the order of N^2 time

  e.g. find the largest number (*N-1* comparisons)

  then the next largest number (*N-2* comparisons)

  and so on …

Sorting

"*Sorting is in P*"

- Different sorting algorithms may take different times
- Comparing each number with the remaining numbers during sorting will on average take of the order of N^2 time

  e.g. find the largest number (*N-1* comparisons)

  then the next largest number (*N-2* comparisons)

  and so on …

  $$(N-1) + (N-2) \ldots = (N \times (N-1)) \div 2$$

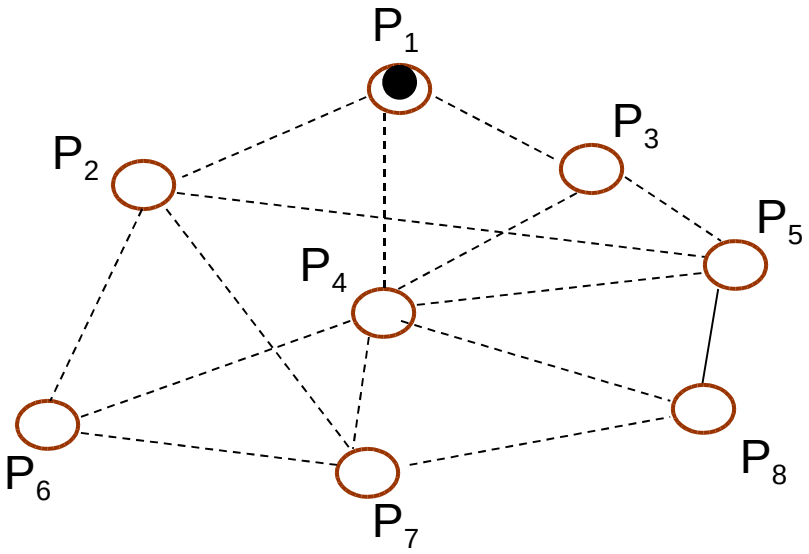
  … $O(N^2)$

- A better algorithm (e.g. Quicksort) will on average take ***N log N*** time

Travelling Salesman Problem

"*The travelling salesman problem is in NP*"

Given N places with the distances between them, there is no polynomial time solution that will find a route touching each place exactly once in a total distance less than K.



- NP-completeness
  - A problem is NP-complete if it is as hard to solve as any other NP problem.
  - If any problem in NP has a polynomial-time solution, they all do!
- Big Question: Is P = NP?
  - $1,000,000 Clay prize for proving or disproving this!

Practicality

Algorithms for many important problems are in NP.

- Therefore not of practical use
- But simplification or restriction may make the problem solvable

Not all problems in P will have practical solutions.

- Manindra Agrawal & students proved that primality testing is in P but their algorithm is not efficient

Many interesting problems have exponential time solutions:

- Algorithms for finding the prime factors of a number all take exponential time in the worst case.
- Possible future use of quantum computing?

1960's: Program Verification

Modern origins:

- Floyd (1967) showed how to prove properties of a program.
- Hoare (1971) gave an axiomatic basis for program properties.

Independently Dijkstra was leading efforts towards formal program derivation.

Start of *programming methodology*.

Much work in this area from 1970's onwards.

How do you prove that a program will compute a desired result?

1. ***Testing***: cannot test for all possible inputs
2. ***Proof by construction***: requires a programming methodology
3. ***Formal proof***: use automated theorem prover or model checker to check program properties.

Formal proof is now becoming more feasible:

- great improvement in machine speeds
- new techniques for proof

Can be used for critical parts of large programs.

1970's in India

By mid-1970's, availability of computers improved.

Work started in India in some new areas of computer science:

- More Indian computer scientists returned to India.
- Groups grew at TIFR, IISc, IITs, Jadavpur University.
- Major group in programming at National Centre for Software Development & Computing Techniques at TIFR.

Manufacture of computers started:

- Electronics Corporation of India produced mini-computers.
- Commercial computers assembled in Mumbai, Pune by IBM, ICL

… but limited by availability of components:

- ICs not manufactured in India

1980's

Two landmark events:

- Launch of the Space Shuttle
- Launch of the IBM Personal Computer (PC)

One can debate:

Which launch had greater scientific impact?

Which launch affected more people?

In India, the PC made computing more widely accessible.

And Indian software companies start to grow.

1980's onwards: Indian computer science

Significant achievements:

- Conferences series started by NCSDCT at TIFR in 1981: *Foundations of Software Technology & Theoretical Computer Science*
- FST&TCS has become a leading conference world-wide
- Tata Research Development & Design Centre created in 1981: first R&D centre in Indian software industry
- Important textbooks written
- Highly referenced publications start emerging
- Major result on primality testing by Manindra Agrawal and students
- …

Indian computer science is still small compared to the Indian IT industry!

New research directions …

General form of many problems in NP, EXP etc.

Can still be tackled in less general form:

- simplify problem
- restrict to particular subsets

Interesting questions:

- What is the largest subset that is practically solvable?

and

- How can a problem be posed so that it can be solved?

Checking the software

Ideally, software should be:

- proved correct for the requirements
- guarantee timing and fault-tolerance
- need no testing …

Checking the software

Ideally, software should be:

- proved correct for the requirements
- guarantee timing and fault-tolerance
- need no testing …

In fact, software must be tested at all stages.

Exhaustive testing is not feasible – takes exponential time.

In practice, testing is done by

- formal analysis
- exercising particular ‘representative’ cases

Code-based Testing

Formal program analysis should be the ideal method.

But it is

- slow (so cannot handle large programs)
- limited in capability

Dynamic adaptive random testing has been proved to be more effective:

- takes less time
- gives better test coverage
- can be made compositional

Patrice Godefroid, Compositional Dynamic Test Generation, POPL, 2007.

Termination

In general, showing that a program terminates is the same as the *Halting Problem* – undecidable.

… but proving the termination of some programs may be possible.

Termination can be proved as follows:

f. Construct a ranking function

g. Show that each action reduces the value of this function

Reachability Analysis

New results show how to automatically prove termination of parts of large programs.

Automated procedure:

a. constructs different ranking functions

b. checks that at least one function leads towards termination

This is the basis for *binary reachability analysis*.

Has been used to prove termination of programs of over 20,000 lines of code …

Byron Cook et al, Termination Proofs for Systems Code, PLDI'06, Ottawa, June 2006.

New research focus – embedded systems

Embedded systems perform critical functions but hide from view!

Few people notice their existence:

- How many embedded systems are there in each car made in India?

… but their failure can be catastrophic:

- SLV2, Ariane 5 launches failed due to embedded system errors

Problems ahead …

Hardware of an embedded system is increasingly reliable.

Making the software *near-error-free* is a major challenge.

Problems at all stages of software development:

- defining the requirements
  - may consist of 1000's of pages of text and diagrams
- designing the program
- keeping the design correct as changes are made
- changes take place at all times
  - new requirements & features
  - correcting old errors, etc.

Mixed Control, Real Platforms

Problems compounded for mixed continuous and discrete control:

- Lack of a common framework for reasoning
- Difficulties in handling concurrency

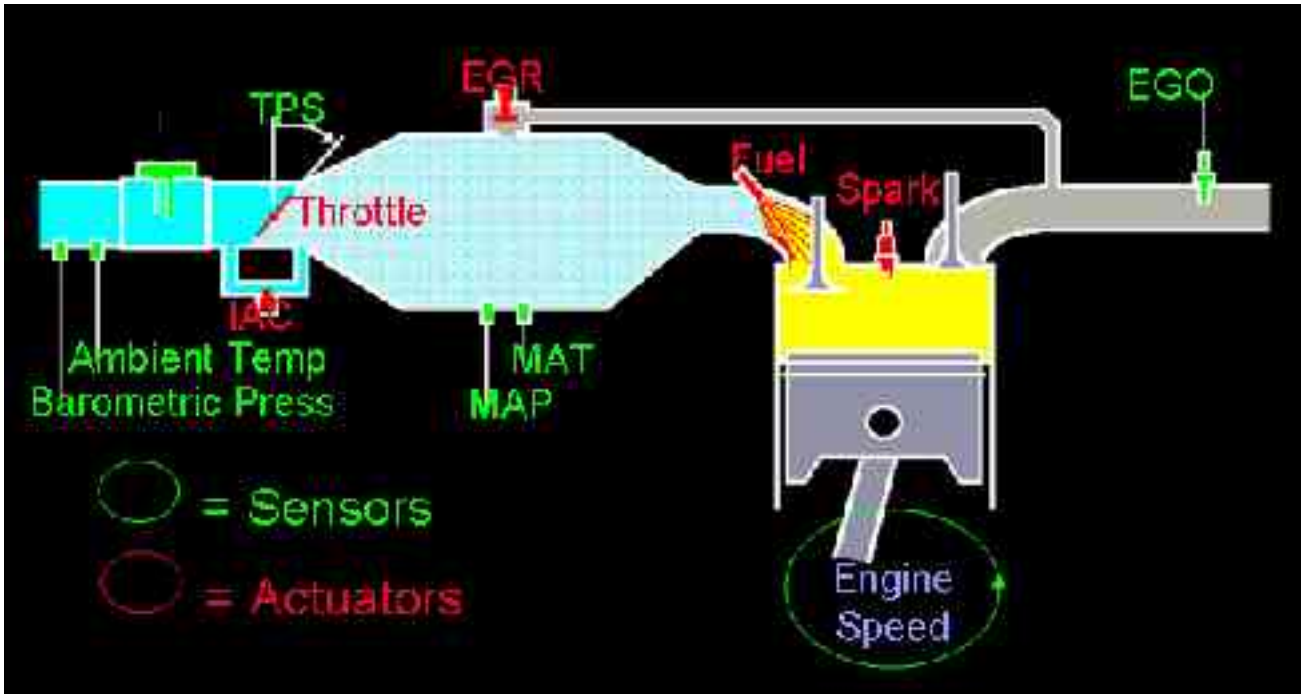
Despite advances in hardware performance:

- Effective use of physical resources is a driving force
- Smaller code footprint, faster execution time are critical

Example: Auto Engine Control

- Discrete events & continuous dynamics
- Limitations of single model
  - Non-linearities, significant parameter variations
- Need a combined approach to deal with different modes of engine operation (finite states) in an integrated way

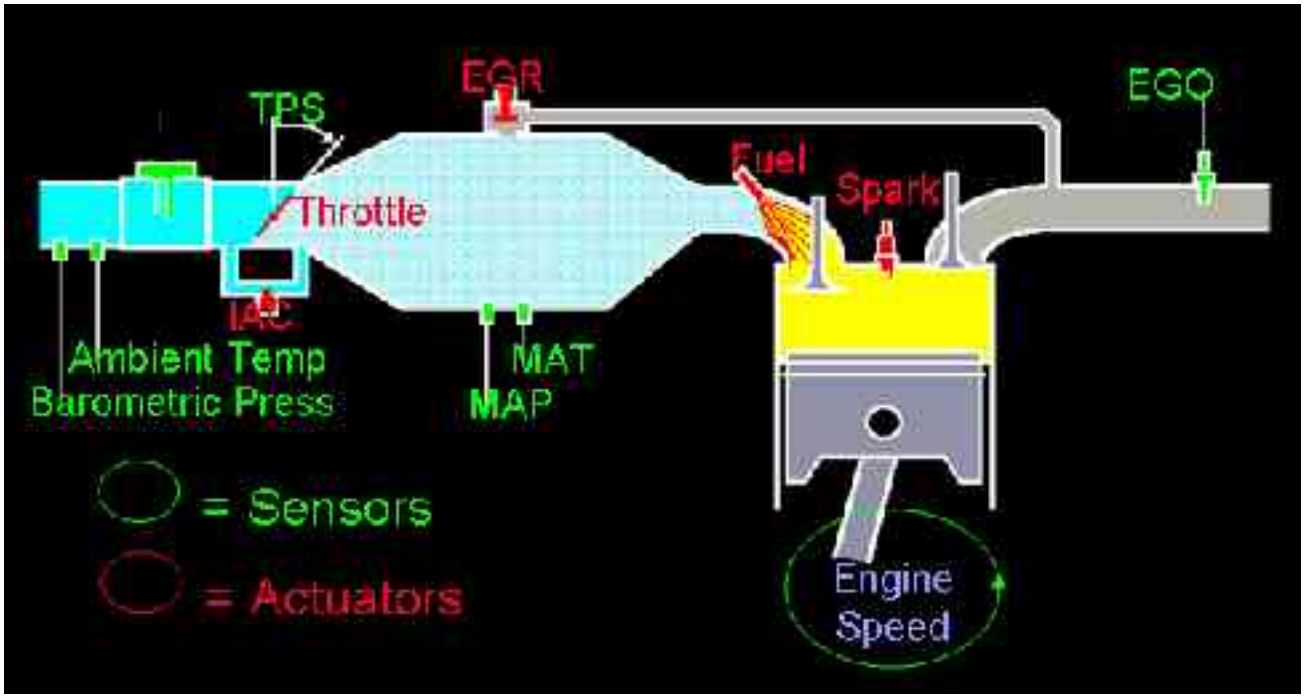
Example: Continuous & Discrete Engine Control



Continuous Control
TPS : Throttle position sensor
MAP: Manifold absolute pressure
MAT: Manifold absolute temperature
EGO: Exhaust gas oxygen
IAC: Idle air control
EGR: Exhaust gas recirculation

From: Vivek Diwanji

Example: Continuous & Discrete Engine Control



Discrete Control
Sequencing of cylinder firing

Continuous Control
TPS : Throttle position sensor
MAP: Manifold absolute pressure
MAT: Manifold absolute temperature
EGO: Exhaust gas oxygen
IAC: Idle air control
EGR: Exhaust gas recirculation

From: Vivek Diwanji

New Class of Embedded Systems emerging …

‘Traditional’ embedded systems have:

- Close hardware-software interaction
- Complex control operations

Sensor networks are different:

- Ultra-lightweight sensor nodes
- Limited functionality, low computing load
- Large number (1000+) of nodes
- Limited reliability of a single node but reliable aggregate behaviour

Smart Dust Mote

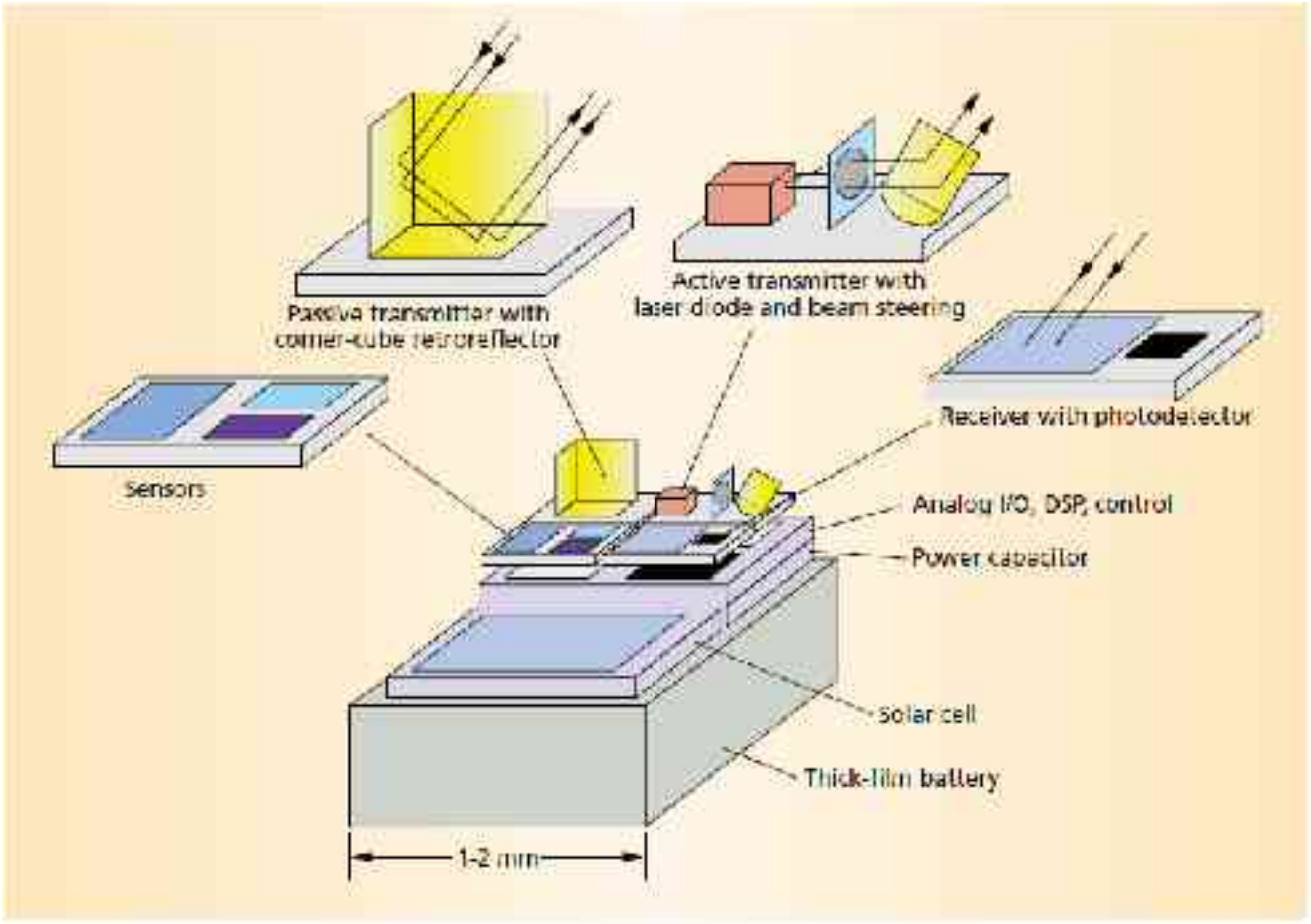


Figure 1. Conceptual diagram showing a Smart Dust mote's major components: a power system, sensors, an optical transceiver, and an integrated circuit.

From Warneke, Last, Liebovitz, Pister: Smart Dust: Communicating with a Cubic Millimetre Conputer, *IEEE Computer* January 2001.

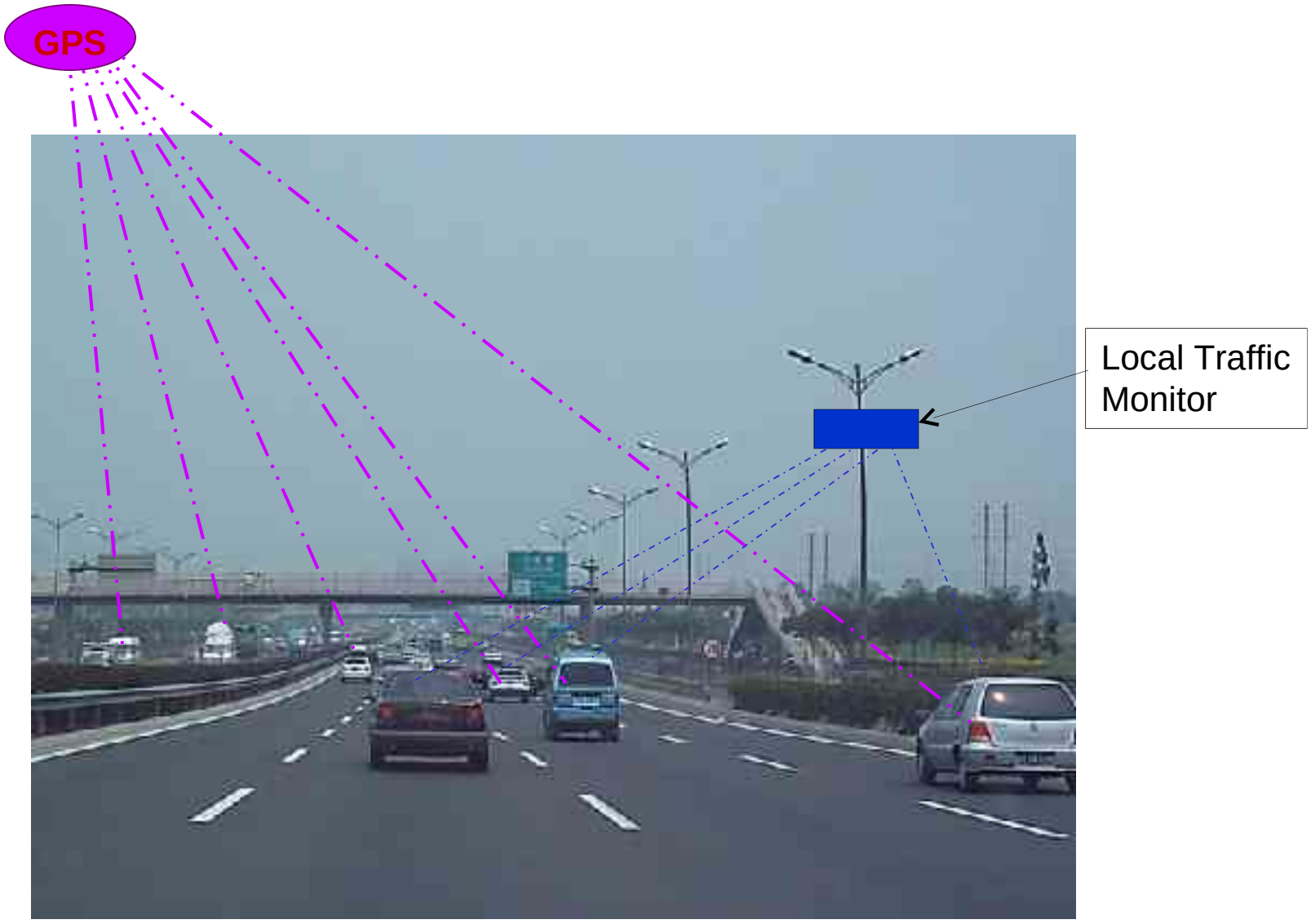
Motes and dust

Motes are small enough to be scattered:

- dropped in clusters
- used as medical sensors

Motes can also be integrated into other devices:

- e.g. added to car Global Positioning System (GPS) devices


GPS
Local Traffic Monitor

GPS

Area Traffic Monitor

Reasoning about motes

For traditional embedded systems:

- system reliability is defined
- prove software correctness for such a system

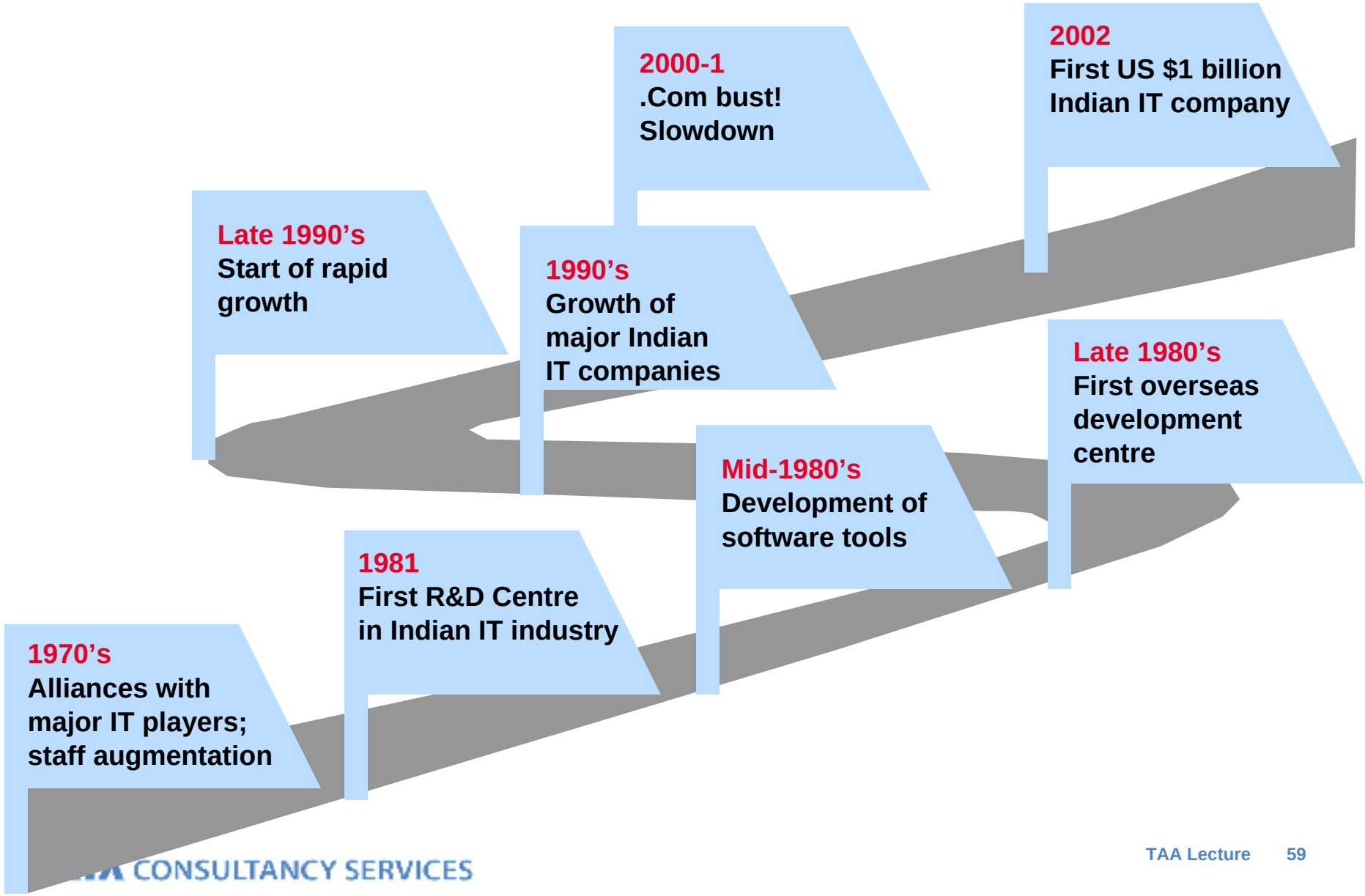
For motes:

- reliability and life of a single mote not certain
- prove properties for collections of motes

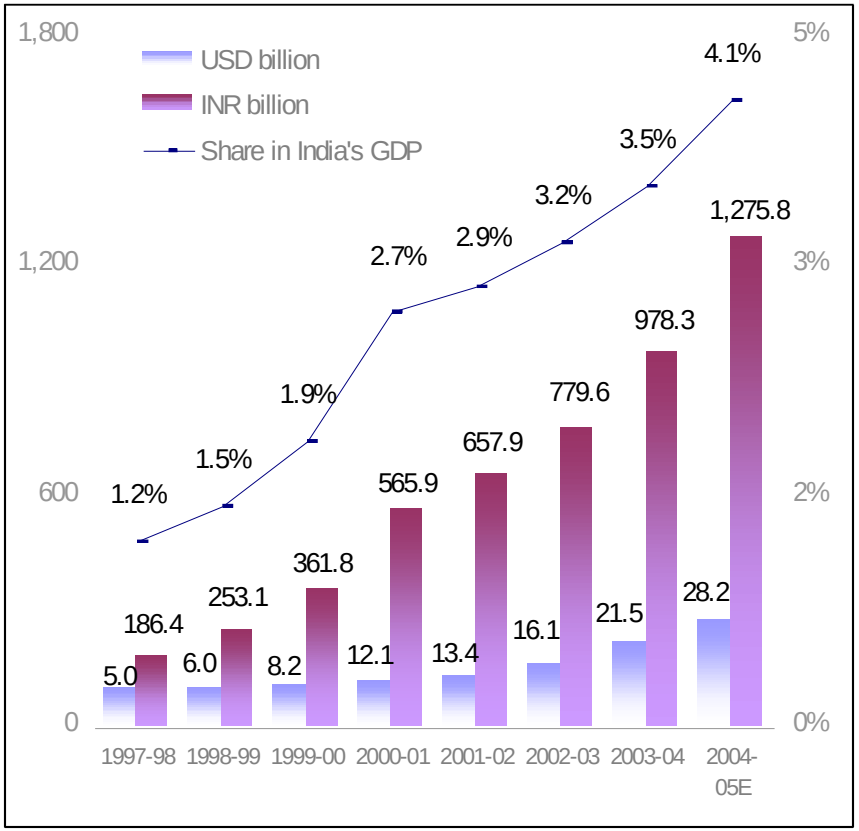
Show correctness under statistical assumptions.

New area for CS research.

Indian IT industry



Recent Growth of the Indian IT Industry



Source: NASSCOM

- Compounded Annual Growth Rate (CAGR) of 28% over 1998-2005.
- Share in India's GDP has grown from 1.9% to 4.1% - should reach 7% by 2008.
- Industry US$37.4B in 2006, US$47.8B in March 2007; could be US$80-100B by 2010.
- Employed 1.3 million professionals in 2006-07; 2.5 million expected by 2010.

Success

The Indian IT industry has changed the whole basis of software development:

- From single teams, to multiple teams using a *software process*
- Development across geographies

Most high-certified companies are in India.

Consequences

The Indian IT industry is a magnet for employment:

- Demand led to an educational shift towards IT
- Career options center on IT

Big growth in educational courses:

- Many new institutions but acute shortage of qualified staff

IT companies have a shortage of properly trained recruits.

Success at a price

Good news or bad news?

- New successful industry providing rewarding positions to young engineers
- This model could be replicated in other areas

Gaps: IT industry needs to promote education, research.

- Number of qualified teaching & research faculty very low
- Leading CS departments not much bigger than 10 years ago
- Few PhD students

Where will tomorrow's CS faculty, research leaders come from?

But is it Science?

Technology? Engineering? Science?

Computing *technology* usually ahead of computer *science*:

- generates the ‘phenomena’ that are observed/studied
- creates the need for new theory
- provides the environment for validating theory

Theory also proceeds independently.

Putting barriers between theory and practice is of no use.

Important to make theory and practice work together.

Science & … science

Traditional science relies on observations of natural phenomena at different levels.

Computer science:

- is not a natural science
- is unquestionably a mathematical science
- has theories, makes predictions
- observes phenomena from world of technology

Computer science is related to IT … as any science is related to technologies.

Finally …

The success of the Indian IT industry has set a challenge:

Can Indian computer science make as big an impact?

Like theory and practice, industry and research must work together.

There is a lot that both have to discover.

Finally …

The success of the Indian IT industry has set a challenge:

Can Indian computer science make as big an impact?

Like theory and practice, industry and research must work together.

There is a lot that both have to discover.

"Machines take me by surprise with great frequency!"

Alan Turing

Thanks

To former and present colleagues at TIFR, Warwick University, Tata Research Development & Design Centre.

To the TIFR Alumni Association for the annual events to commemorate Mr. JRD Tata's birthday on 29th July.