The Hopes and Hurdles of Automatic Verification

Marta Kwiatkowska
School of Computer Science
University of Birmingham

Inaugural Lecture, 17th October 2002

Lecture plan

- Setting the context
- Automatic verification
  - what is it?
  - why verify?
  - what are the challenges?
  - what are the future prospects?
- A glimpse of what we do in Birmingham
- What we hope to work on in future
- Personal corner

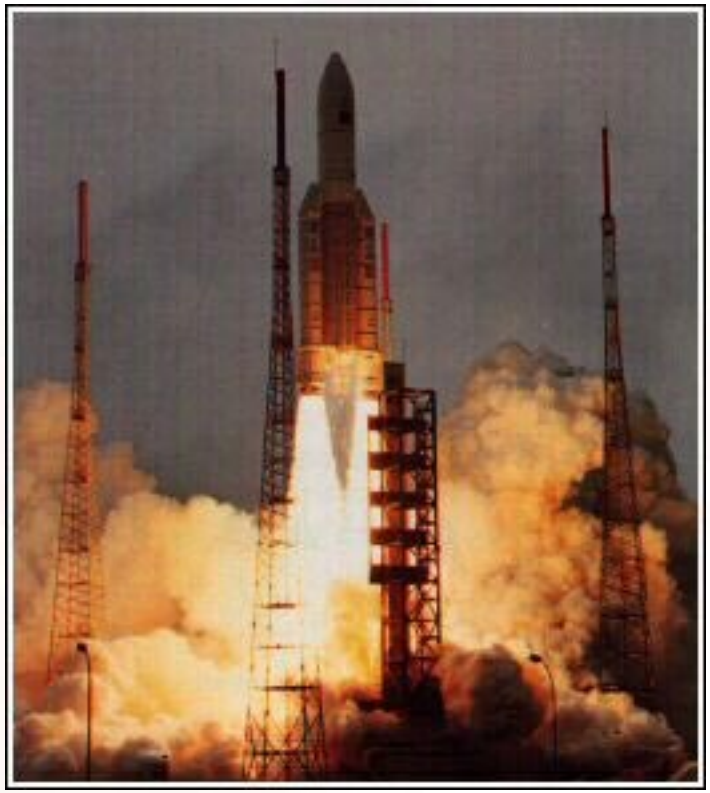
Setting the context

From the news, some real-life stories you may have heard about…

The Ariane 5 story

Ariane 5 launcher, ESA
(European Space Agency)

Shown here in maiden flight
on 4th June 1996



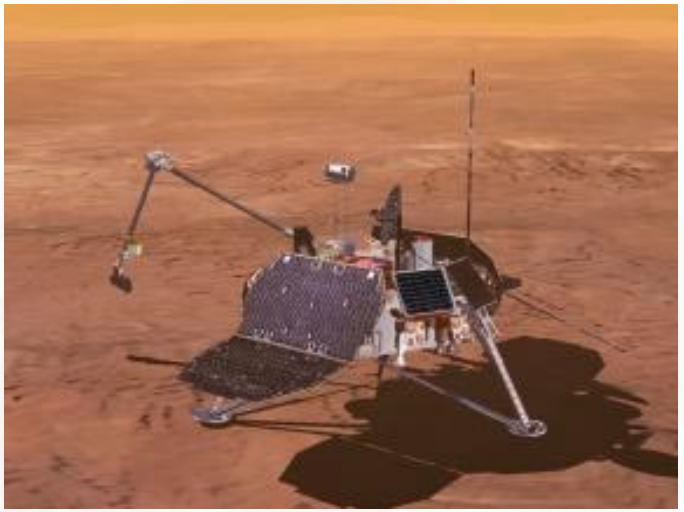
37secs later self-destructs:
Uncaught exception: numerical overflow in a conversion routine results in incorrect altitude sent by the on-board computer

Source: http://sunnyday.mit.edu/accidents/Ariane5accidentreport.html

The NASA Mars space mission story

Mars Climate Orbiter
Launched 11th December 1998

LOST 23rd September 1999
Conversion error from English units to metric in navigation software



Mars Polar Lander
Launched 3rd January 1999

LOST 3rd December 1999
Engine shutdown due to spurious signals that gave false indication that spacecraft had landed

Source: http://mars.jpl.nasa.gov/msp98/

The London Ambulance Service story

Computer Aided Despatch system, London Ambulance Service

Area 600sq miles
Population 6.8million
5000 patients per day
2000-2500 calls per day
1000-1200 999 calls per day

Introduced
26th October 1992

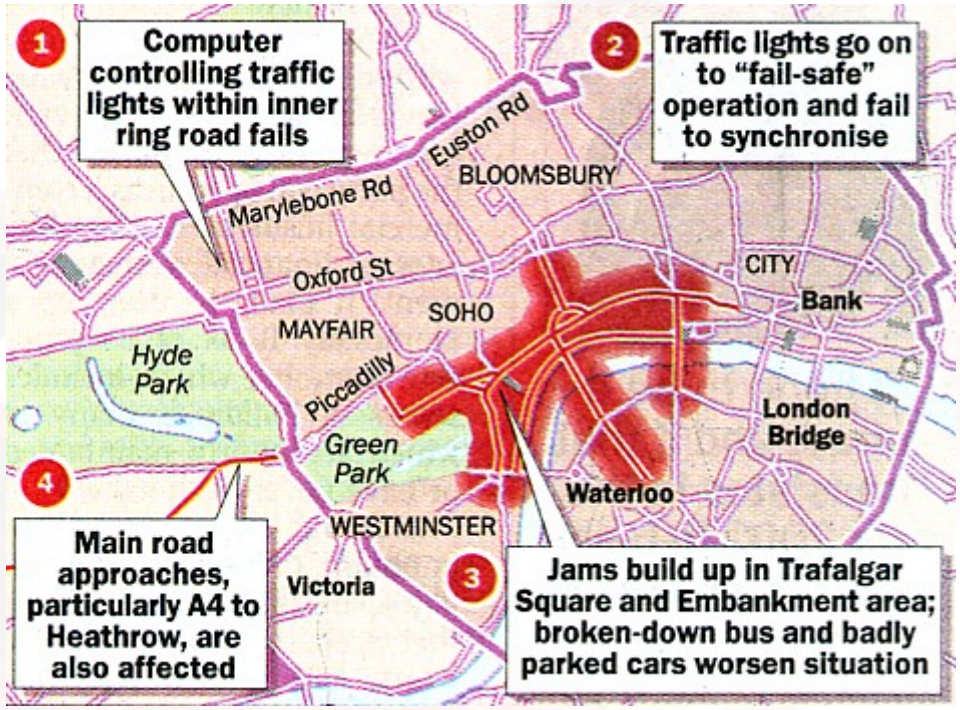
Severe system failure:
Position of vehicles incorrectly recorded, multiple vehicles sent to the same location. 20-30 people claimed to have died as a result

Source: http://www.lond.ambulance.freeuk.com/cad.html

The London traffic lights story

New software to govern traffic in Piccadilly area

4,500 traffic lights
3 computers
180,000 vehicles



25th July 2002, 6-7.30am

Total jam in Trafalgar area: one of the computers fails and forces the other two into default mode.

Source: The Times 25th July 2002

The BMW story

BMW 3 Series,
biggest selling BMW

100s of embedded components used in modern cars

Extensively tested

BMW known for quality, but ....

1999 - safety recall of 16,000+ cars
Faults with airbag control unit - in certain conditions airbag can inflate for no reason

Source: http://www.theaa.co.uk/motoringandtravel/carrecall/

The Amulet story

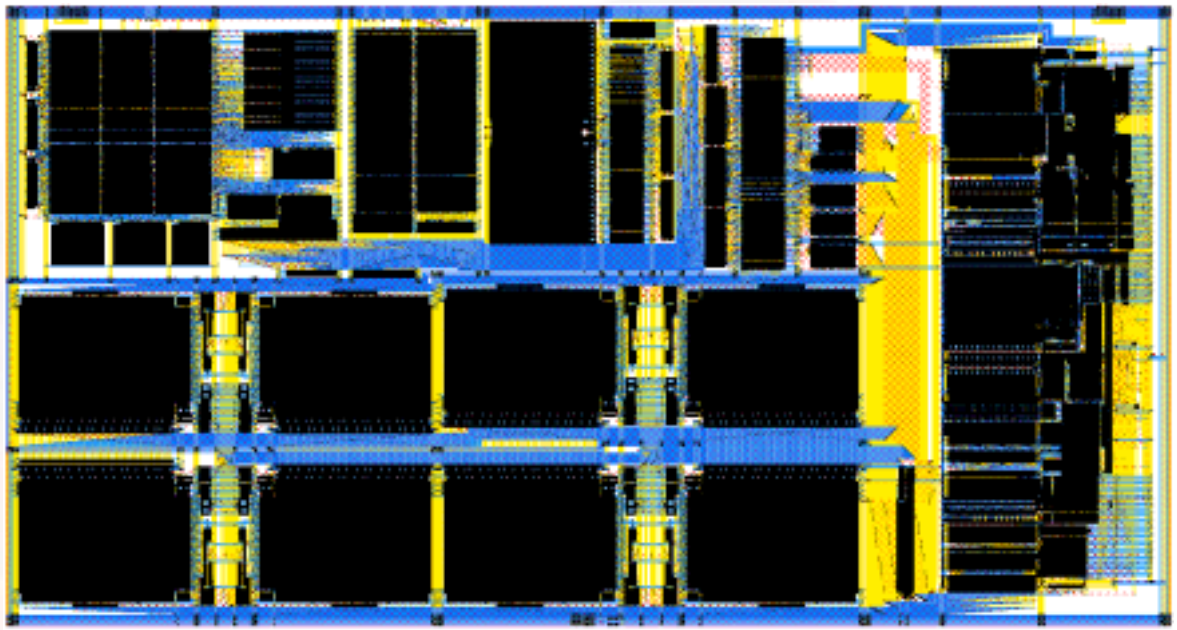
Amulet3i, includes 8KB RAM, 16KB ROM and Amulet3 processor

Designed at the University of Manchester

Low power, high performance, asynchronous, ARM equivalent.

Size **7.5x3.5mm**

Silicon in year 2000



825,000 transistors! (500,000 in RAM)
Flaws detected after chip manufacture (certain operations fail due to logic oversight)

Source: http://www.cs.man.ac.uk/amulet/projects/AMULET3i.html

The BlueTooth meets WiFi story

Ericsson T68, BlueTooth enabled, 3G

PDA, BlueTooth enabled

Laptop PC, BlueTooth & WiFi enabled

Printer, BlueTooth enabled

No wires!

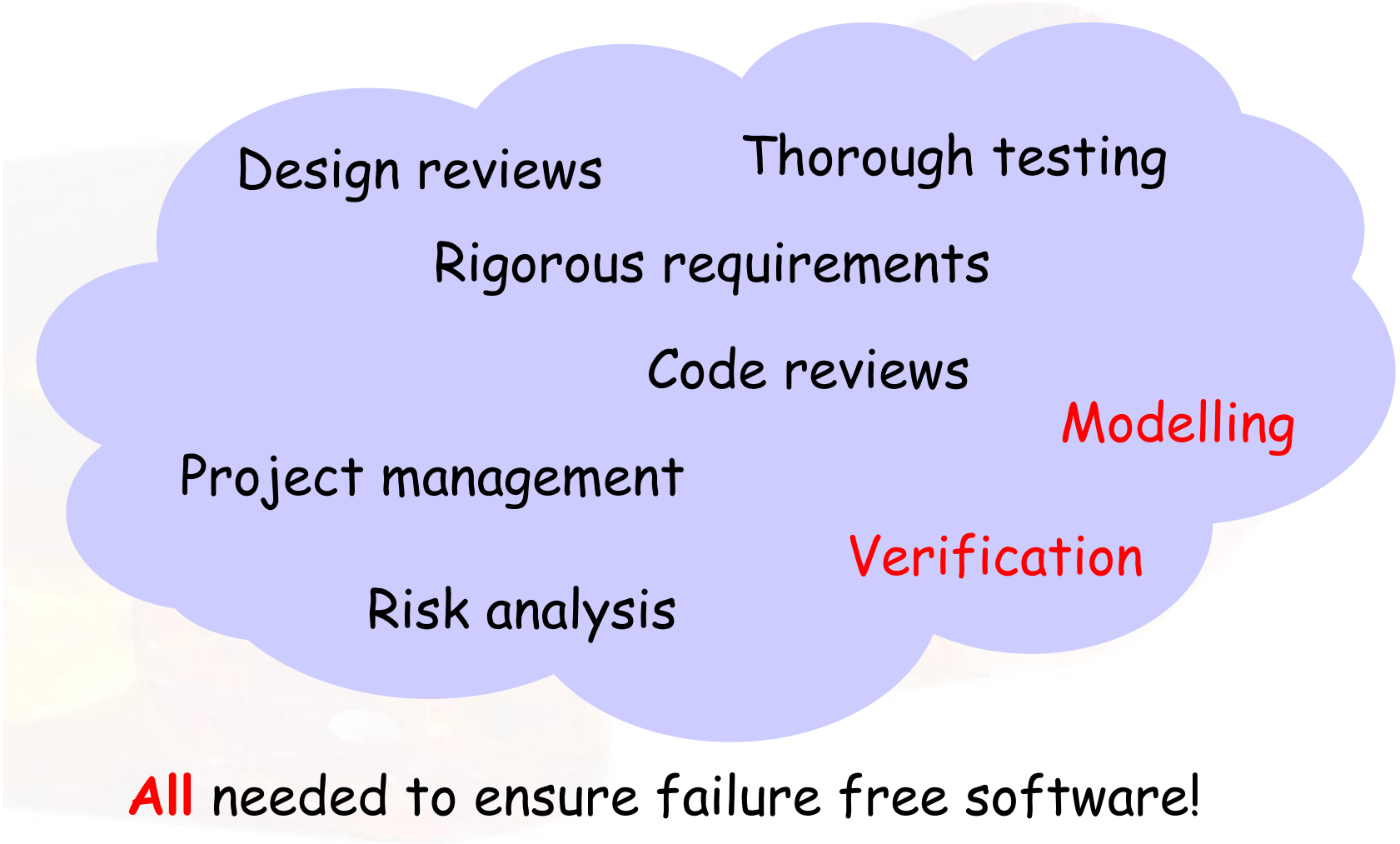
2002 – BlueTooth interferes with WiFi (802.11b)
If less than 6in apart, BlueTooth forces WiFi to retransmit, causing substantial delays

Source: http://www.extremetech.com/article2/0,3973,66509,00.asp

What do these stories have in common?

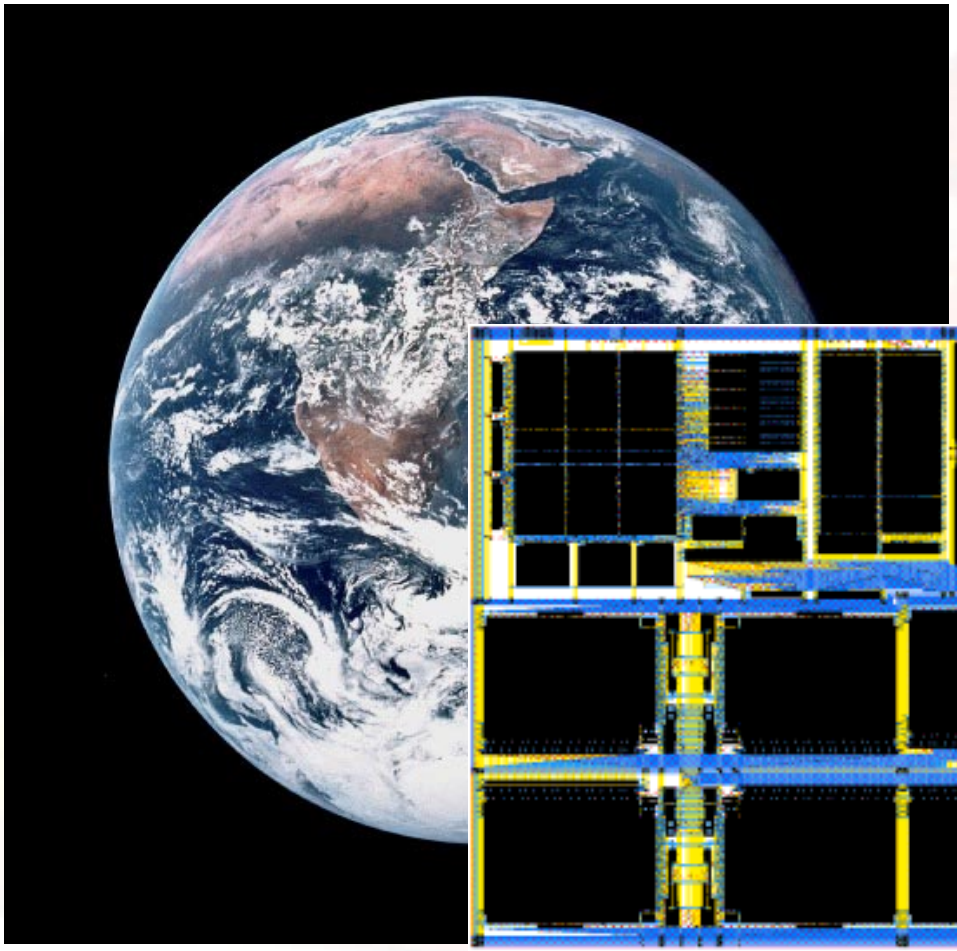
- Programmable computing devices
  - conventional computers and networks
  - software embedded in devices
    - airbag controllers, mobile phones, etc
- Programming error direct cause of failure
- Software critical
  - for safety
  - for business
  - for performance
- High financial cost incurred
- Failure avoidable

Can we engineer better software?



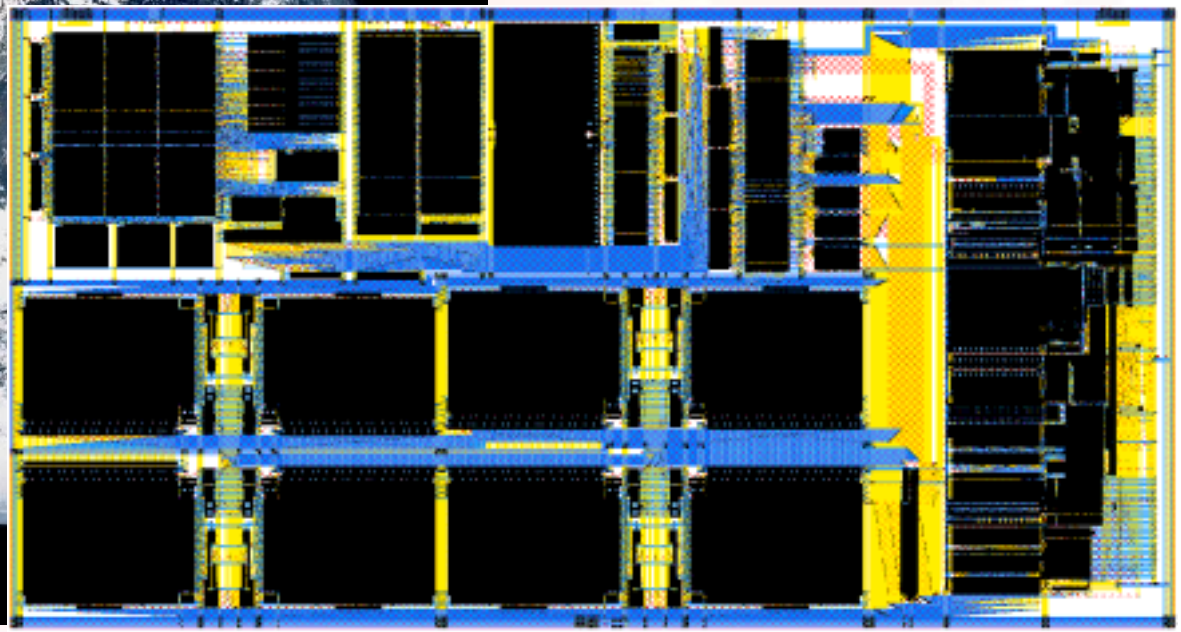
All needed to ensure failure free software!

The problem of size and complexity

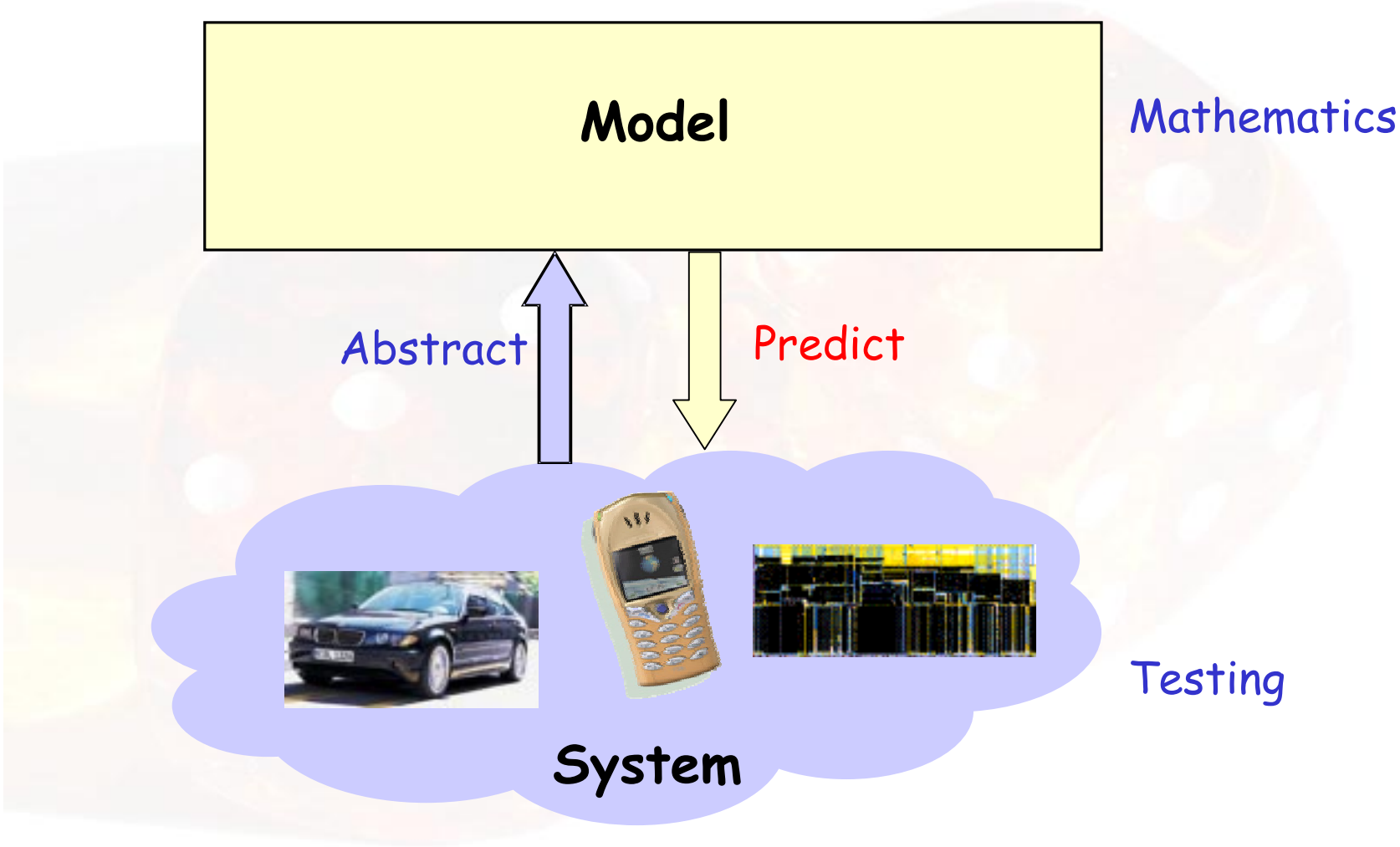


10^{70} atoms

$10^{500,000}$ states



Managing design complexity



Why model and analyse?

- Ensure safety
  - must never administer wrong dosage
- Ensure reliability
  - will never crash
- Reduce production costs
  - analyse and debug first, then manufacture chip
- Ensure timely response
  - will produce the next batch within 0.5sec
- Predict performance
  - will transmit 1 MB per second on average

What is automatic verification?

Verification =

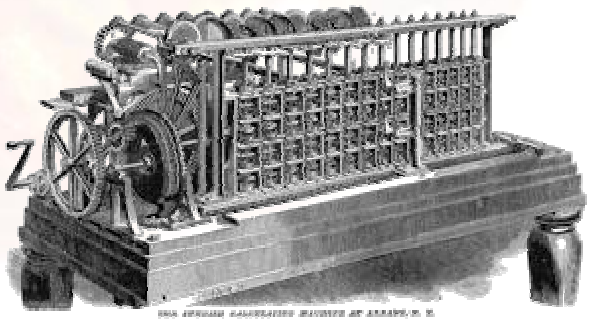
method for ensuring that the program is correct before it is executed.

Amounts to proving that the program satisfies the specification.

Automatic verification =

mechanical, performed

without human intervention.



Why must we verify?

"Testing can only show the presence of errors, not their absence."

"In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, computers are without precedent in the cultural history of mankind."

Edsger Wybe Dijkstra

1930-2002

To rule out errors must consider all possible executions – often not feasible mechanically!

Source: http://www.cs.utexas.edu/users/UTCS/notices/dijkstra/ewdobit.html

Verification: it began in the beginning

Friday, 24th June 1949
EDSAC inaugural conference, Cambridge

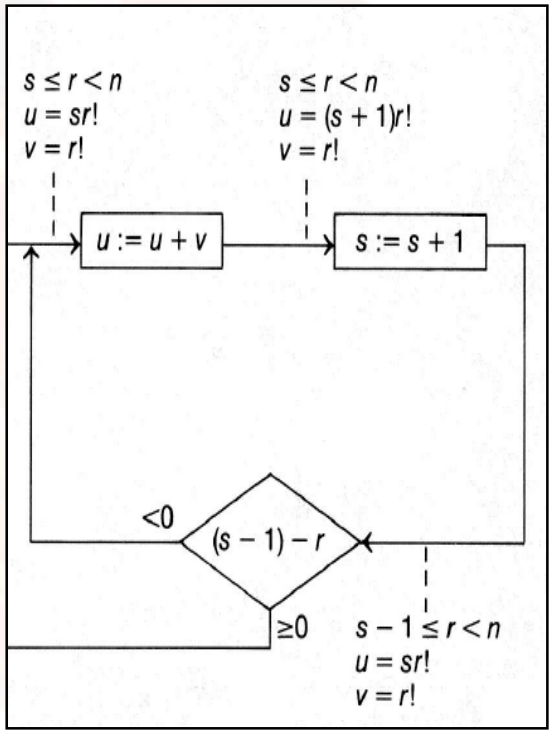
Checking a large routine by Dr A. Turing.

"How can one check a routine in the sense of making sure that it is right? In order that the man who checks may not have too difficult a task the programmer should make a number of definite assertions which can be checked individually, and from which the correctness of the whole programme easily follows."

Source: Annals of the History of Computing, Vo 6, No 2, April

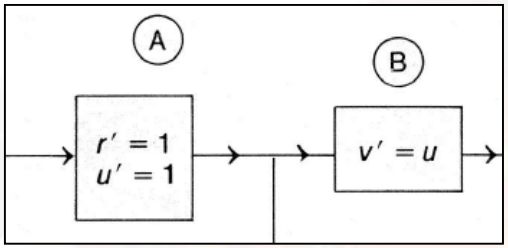
Assertion-based verification

- Introduced by Floyd (1967)
  - annotation with boolean assertions
    {P} command {Q}
  - prove that all assertions true on all possible executions
  - use automatic theorem prover
- Axiomatic (Hoare 1969)
  - deductive inference rules
  - formal proof of correctness
- Program calculus (Dijkstra 1976)
  - programs correct by construction



Turing's view of verification

Program



Assertions

From A to B
with r'=1, u'=1

Programmer

Checker

Source: Annals of the History of Computing, Vo 6, No 2, April 1984

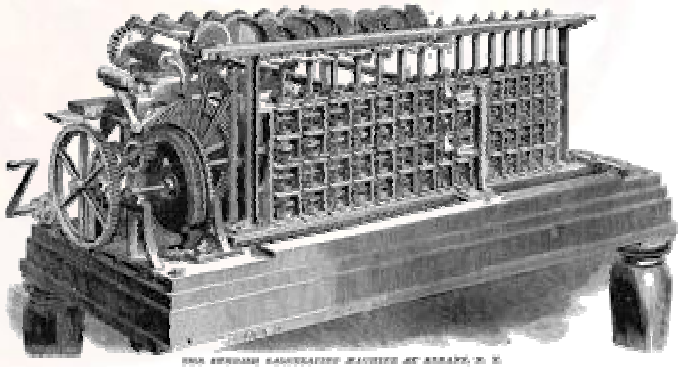
Floyd's view of verification

Annotated program

```
{a<N & b=0}
begin
a:=a+1; b:=b+a
end
{a<N+1 & b=a}
```

Programmer

∀p∃q ∀p ∀p∃q



The verifying compiler

Source: R.W. Floyd, Assigning Meanings to Programs, 1967

Dijkstra's view of verification

Program correct by construction

wp(a<N+1 & b=a)
do a<N → a:=a+1; b:=b+a
od {a<N+1 & b=a}

guard (a<N)

command

wp(a<N+1…)

∀p

Programmer & Prover

Source: E.W.Dijkstra, A Discipline of Programming, 1976

Assertion reasoning: the hurdles

- How to deal with concurrency?
  - interference for shared variables
  - proof rules difficult
- How to automate?
  - need a theorem prover
- How to ensure acceptance by programmers?
  - impractical for realistic programs
  - methodological difficulty?
  - market demand?

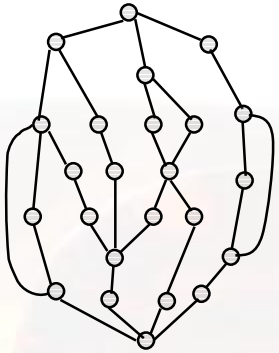
Progress slow since 1967...
The verifying compiler still an ideal.

Model-based verification

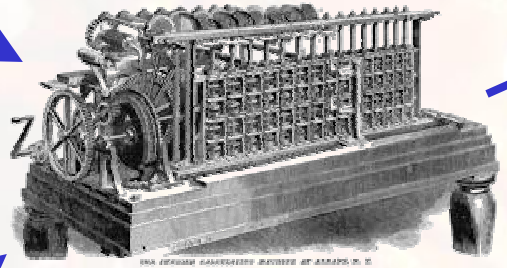
- Implicit in Pnueli (1977)
  - pertains to a single program
  - specifications given in temporal logic, e.g.
    for all executions, eventually response received
  - decidable for finite-state models
  - uniform for sequential & concurrent systems
- Implemented already in 1981 (Clarke & Emerson; Quielle & Sifakis) as "model checking"
  - model derivation automatic
  - fully automatic property checking

Not as powerful – but push-button technology!

Verification via model checking



Finite-state model



Model Checker

$init \rightarrow \Diamond response$

Temporal logic specification

or

Error trace

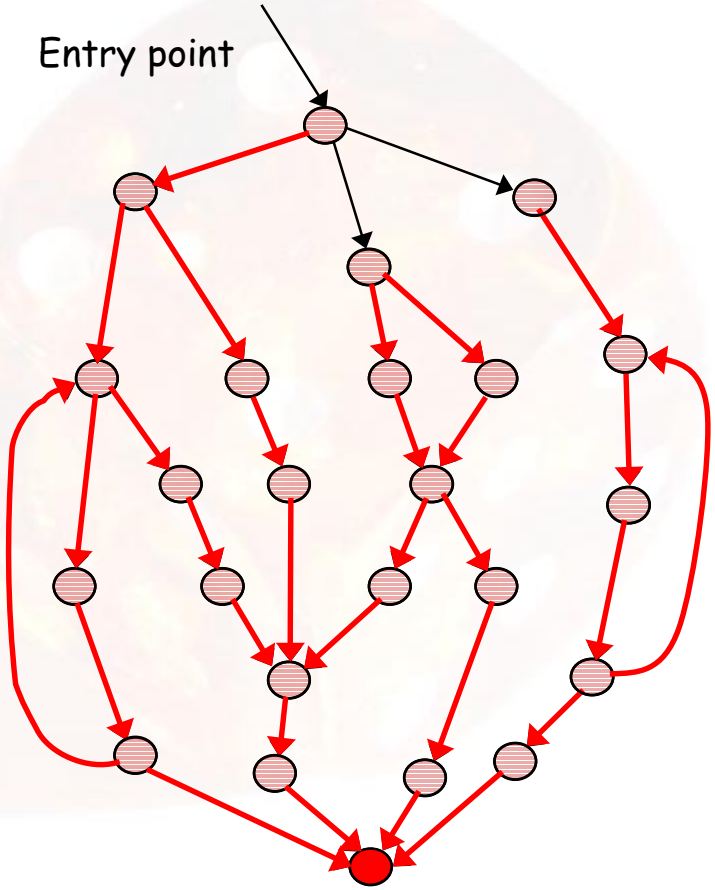
```
Line 5: ...
Line 21: ...
Line 15: ...
  ...
Line 27: ...
Line 45: ...
```

How does it work?

Is it possible to reach a danger state from the initial state?

Does every execution from the initial state lead to danger state?

Model finite-state, hence termination assured.



Verification… or falsification?

- More value in showing property violation?
  - model checkers used as debugging tool!
  - at IBM bugs detected in arbiter that could not be found with simulations
- Widely accepted in industrial practice
  - Intel, Cadence, Bell Labs, IBM, Microsoft, …
- Many software tools, including commercial
  - smv, SPIN, FDR2, FormalCheck, RuleBase, …
  - hardware design, protocols, software, …

Much progress since 1981!
Hopes are being raised. But…

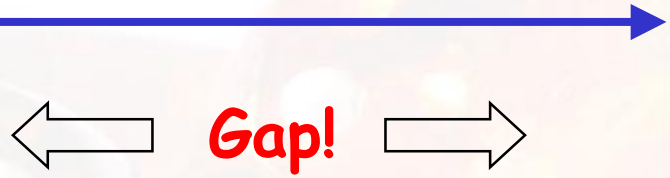
Source:http://www.haifa.il.ibm.com/projects/verification/RB_Homepage/acm.html

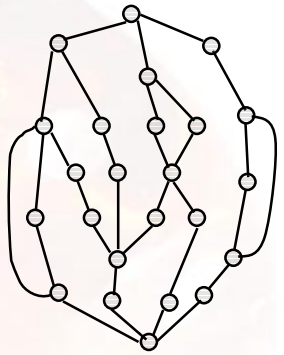
Real software!

```
void add(Object o) {
 buffer[head] = o;
 head = (head+1)%size;
}

Object take() {
 ...
 tail=(tail+1)%size;
 return buffer[tail];
}
```

Program





Model

- Semantic gap between programming languages
  - methods, inheritance, dynamic creation

  and model description formalisms
  - automata
- Infinite state models, must abstract

Concurrency!

- Many systems composed of concurrent components
  - microprocessors, Java programs with threads
  - complex interaction patterns
- State space explosion problem
  - 2 components, 10 states each – potentially 10^2 states
  - 20 components, 10 states each – potentially 10^{20} states
- Symbolic model checking helps
  - **"10^{20} states and beyond"**, Clarke et al
  - **10^{1300}** achieved with decompositions

But 850,000 transistors in Amulet3i - potentially $2^{850,000}$ states

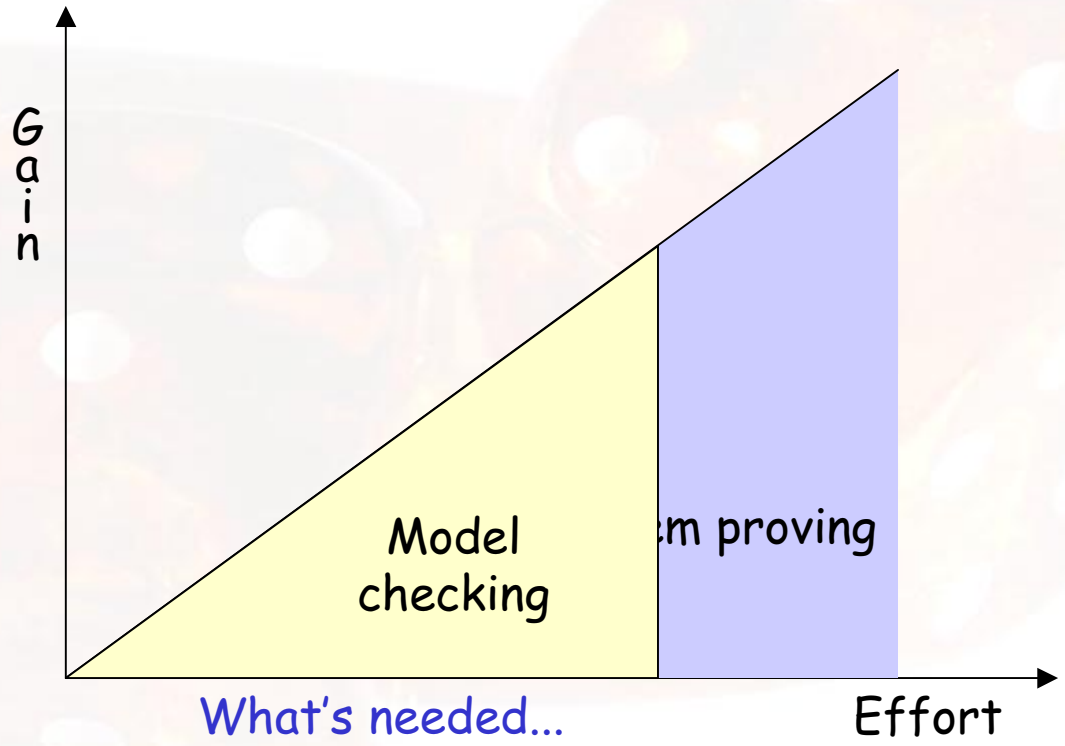
Clocks and other sensor input

- Many systems are hybrid
  - discrete transitions
  - continuous flows
    - real-time, distance, temperature
- Models are highly infinite state
  - undecidability results abound
  - automation problematic
- Need methods for
  - derivation of finite state models
  - proof methodologies
  - verification algorithms & tools

What about probability?

- Coin tossing used in distributed protocols
  - as a symmetry breaker
  - for faster protocols
- Probability used to model
  - failures
  - packet loss rates
  - packet arrival rates
- Need
  - probabilistic specifications
  - modelling formalisms
  - verification algorithms & tools

Model checking versus theorem proving



Source: John Rushby, FORTE'99 invited lecture

The software industry response

"… automatic software verification had advanced. **Let's just spend $50 million**, buy PREfix, and use it on our own stuff to improve it, and then put it out as a product. And this is where you "prove" programs -- can they have a buffer overrun? can they have an invalid condition?--which has been a Holy Grail.

When I was a computer science student, which is now 25 years ago, we thought we were on the verge of proving programs at Harvard."

Bill Gates,
Microsoft

Q&A: Bill Gates on Trustworthy Computing,
May 20, 2002

Towards the Verifying Compiler? Strachey lecture by Tony Hoare

Source: http://www.informationweek.com/story/IWK20020517S0011

What we actually do in Birmingham

- Focus on probability and real-time
  - modelling formalisms
  - model checking algorithms
  - implementation techniques
  - simulation

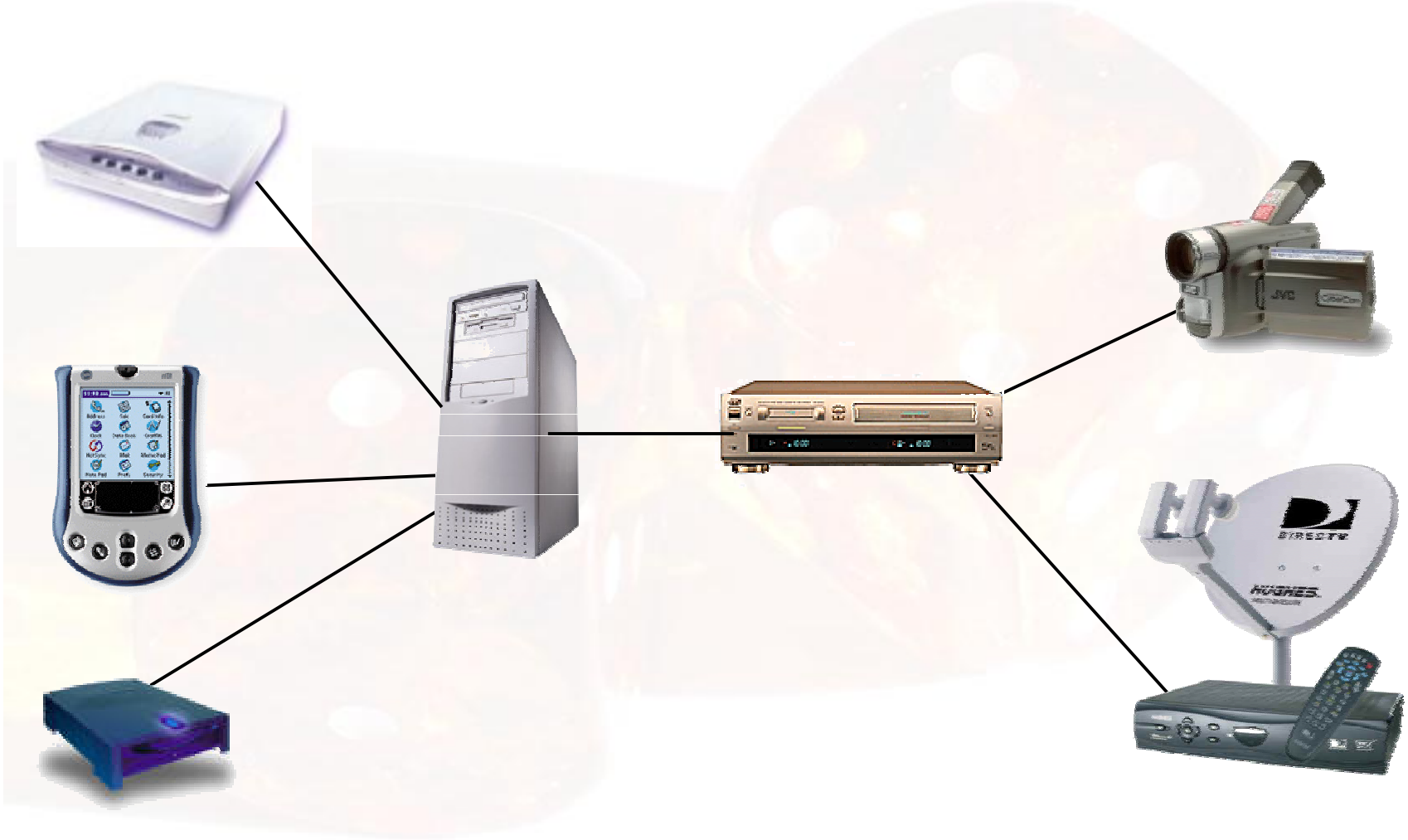
- The software tool PriSM (Probabilistic Symbolic Model Checker)
  - www.cs.bham.ac.uk/~dxp/prism/
  - open source, 300+ downloads, already in used to analyse
- Applications of model checking in telephony
- Case studies

Case study: FireWire protocol

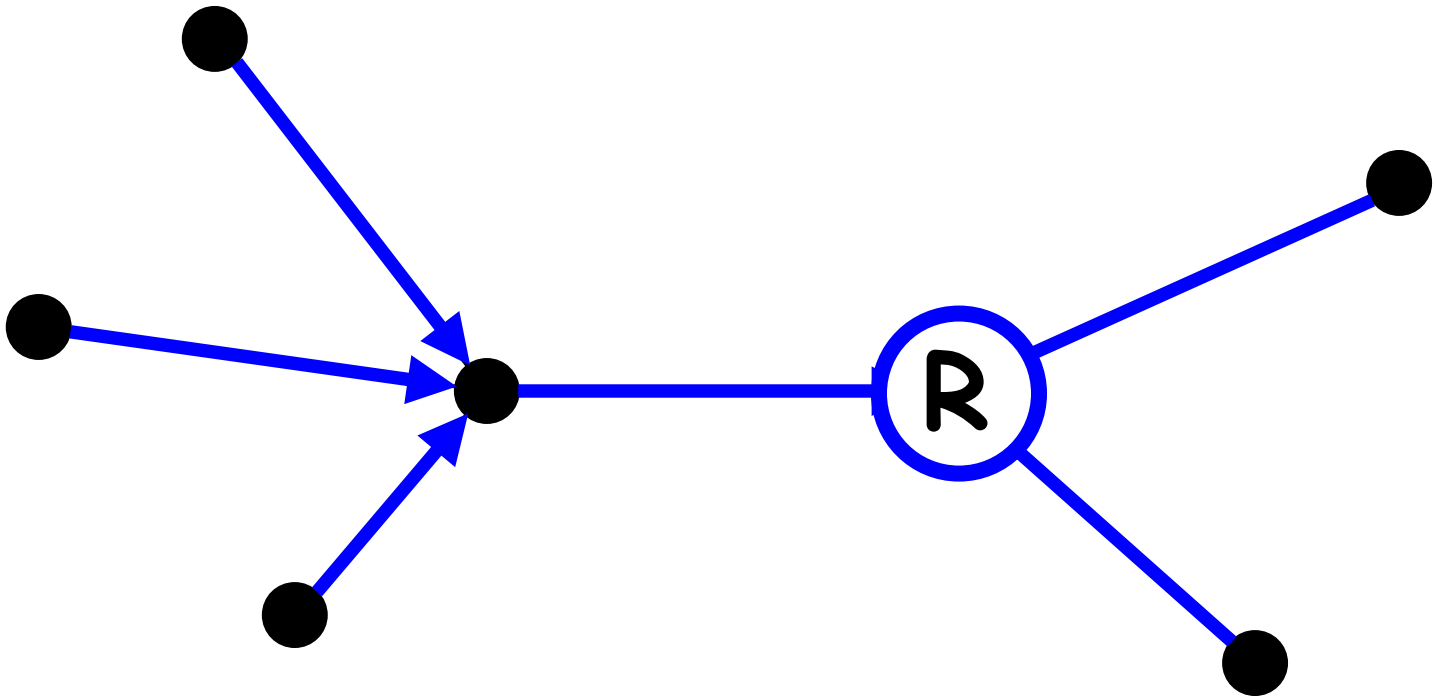
- FireWire (IEEE 1394)
  - one of fastest standards, high data rate
  - multimedia data
  - originally by Apple, mid-90s
  - winner of 2001 PrimeTime Emmy Engineering Award
  - no requirement for a single PC (acyclic topology, not tree)
  - "plug and play"
- Initial configuration
  - involves leader election
  - symmetric, distributed protocol
  - uses electronic coin tossing and timing delays

Source:http://www.apple.com/pr/library/2001/aug/22fwemmy.html

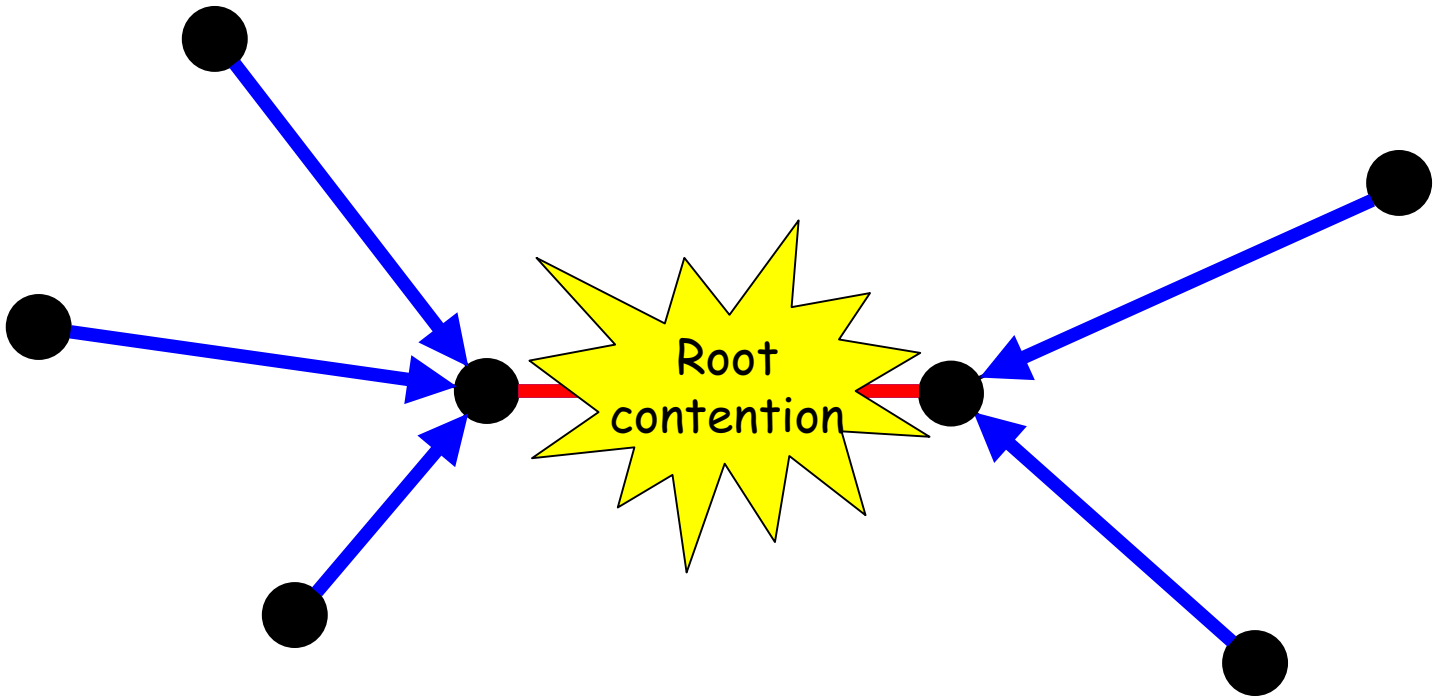
Typical FireWire configuration



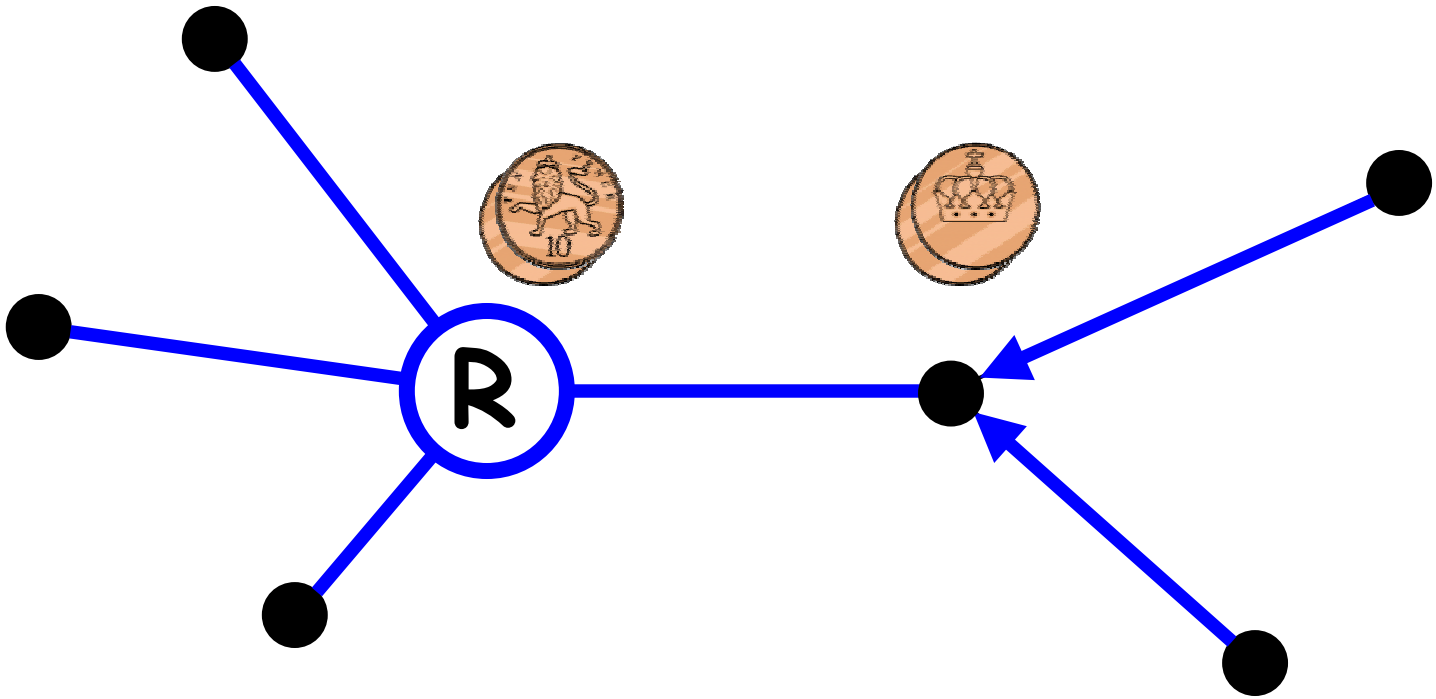
FireWire Initial Configuration



FireWire Root Contention



FireWire Root Contention

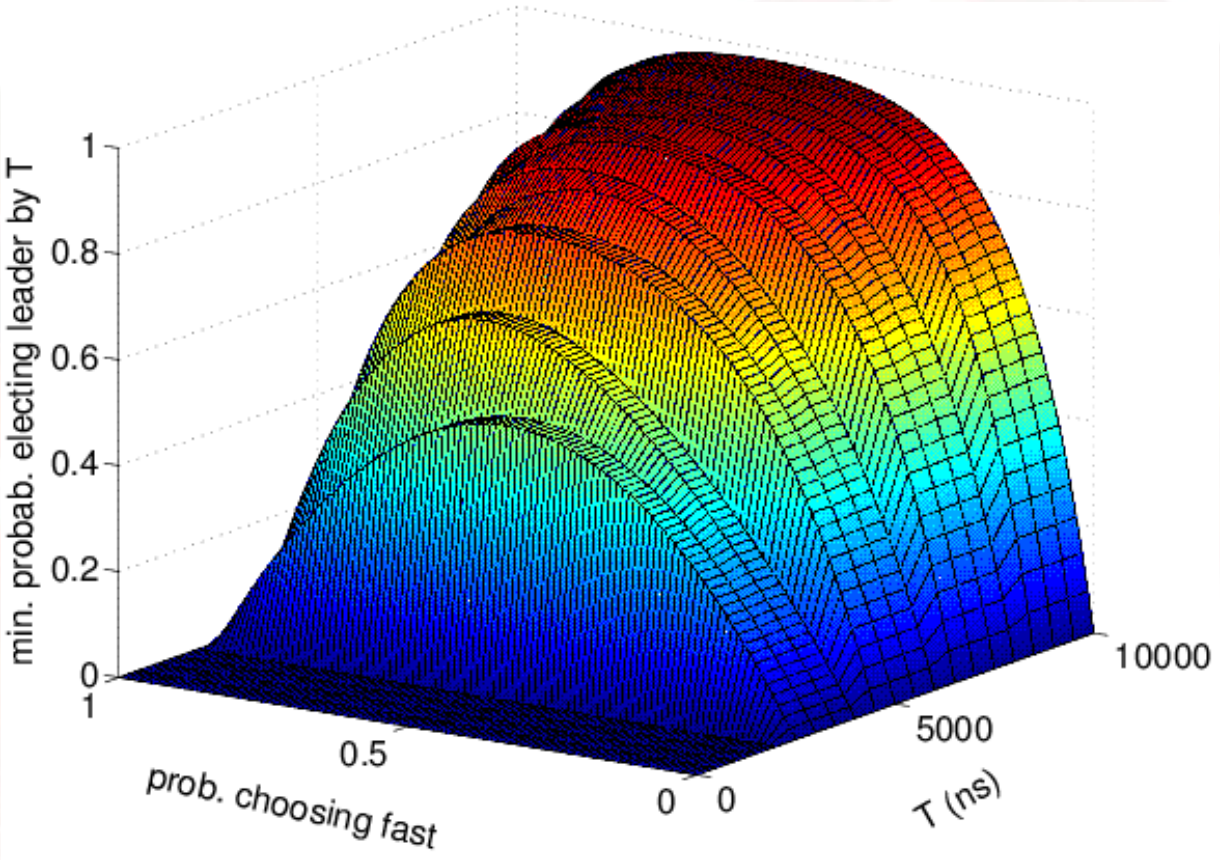


FireWire analysis

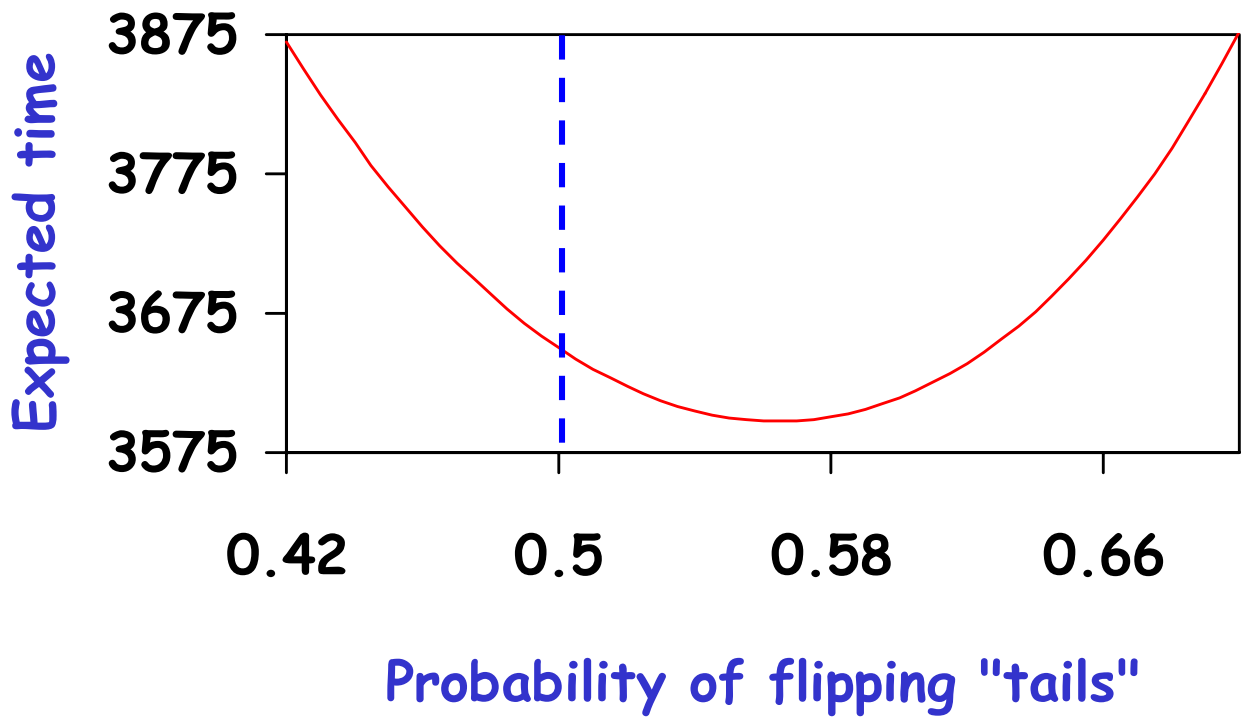
- Real-time properties
  - analysed by Vandraager and Stoelinga
  - used the UPPAAL model checker
  - shown correct wires longer than standard
- Probabilistic analysis
  - considered expected time to root contention
  - used the PriSM model checker
  - timing delays taken from standard
  - established that root contention resolved with probability 1
  - a peculiarity found… (conjectured by Stoelinga)

Source:http://www.cs.bham.ac.uk/~dxp/prism/

FireWire: Analysis Results



Unfair coin gives advantage!



More PriSM case studies

- IEEE 802.11 MAC protocol
  - random back-off scheme
  - probabilistic analysis

    "what is the probability that a station delivers a packet within 10,000microseconds?"

- Crowds anonymity protocol
  - for disguising the originator of messages
  - already implemented
  - unusual behaviour found: probability of being detected increases as the crowds size grows

Source: http://www.cs.bham.ac.uk/~dxp/prism/

Future verification projects

- Proof support for PriSM
  - compositionality and abstraction
  - analysis of probabilistic protocols
- Analysis of ad hoc networks
  - routing protocols
  - verification by model checking
  - Quality of Service and performance (BlueTooth, etc)
- Design support tools for asynchronous hardware
  - formal verification using FDR2
  - distributed simulation
  - asynchronous designs such as Amulet

Centre of Excellence

- Focus on Large Complex Systems
  - concurrency and interaction patterns
  - evolution over time and in space
- Modelling of
  - silicon chips for verification
  - highways for traffic prediction
  - particle physics experiments
  - molecular structures
  - alloys
  - etc
- Collaborate, share expertise

Personal corner

Pause to reflect and say thanks!

Personal journey

- Intellectually
  - from software practice, to theory, back to practice
- Geographically and culturally
  - from Kraków in Poland, via Leicester to Birmingham

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

University of Leicester

THE UNIVERSITY OF BIRMINGHAM

- Vertically
  - career-wise

Research Motto

" Imagination is more important than knowledge."

" Anyone who has never made a mistake has never tried anything new."

Albert Einstein

"Though a mathematician needs no laboratories or expensive equipment, he does need a proper mathematical atmosphere. For a research worker, collaborators are almost indispensable, for in isolation he will in most cases be lost. (..) An isolated researcher knows much less than those who work as a team. Only the results of research, the finished ripe ideas, can reach him and then only when they appear in print. The isolated researcher does not know when and how they have been obtained: he does not experience this process together with their authors."

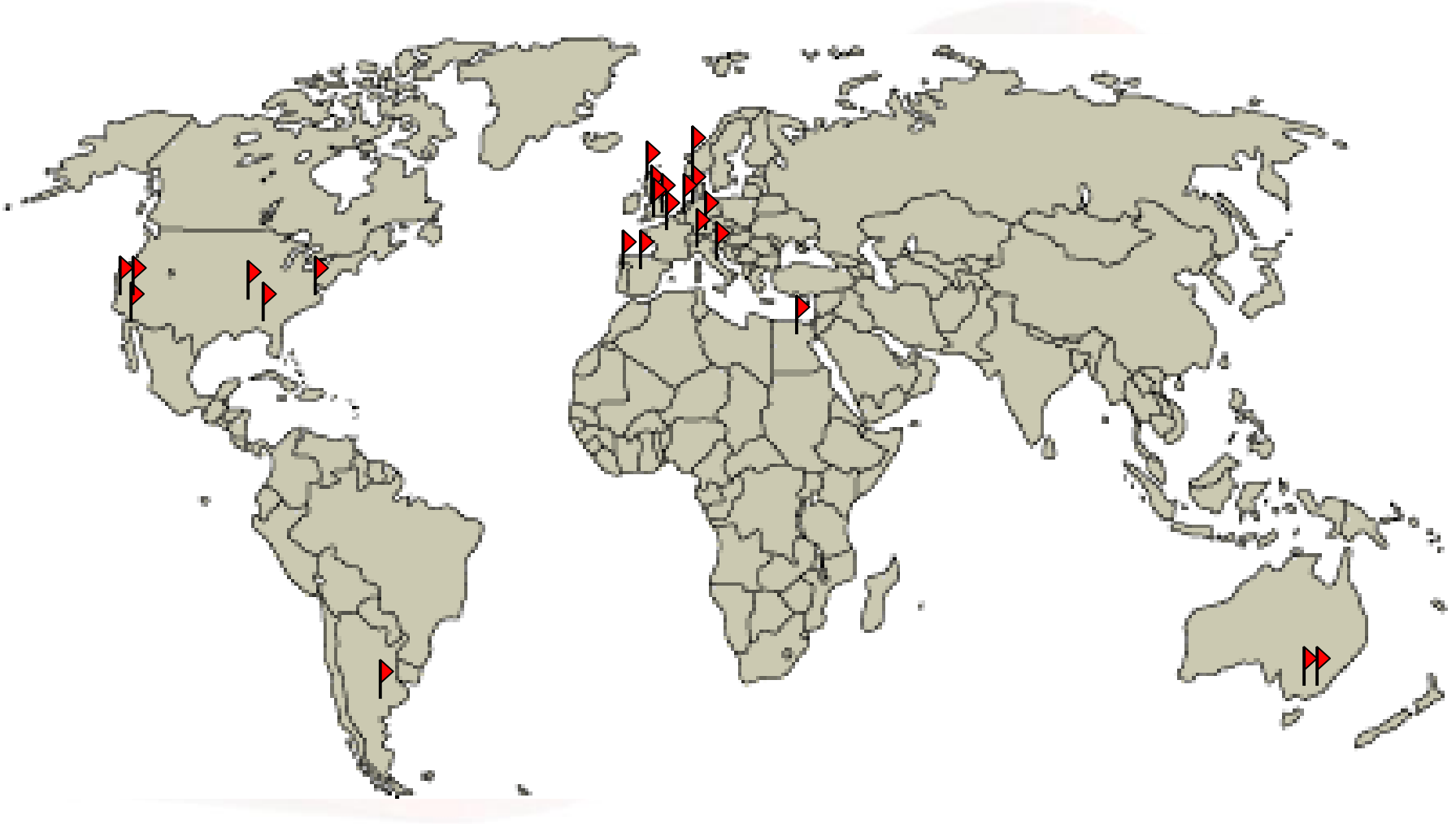
Z. Janiszewski, *On the Needs of Mathematics in Poland*, 1918

Collaborators, contributors – thanks!

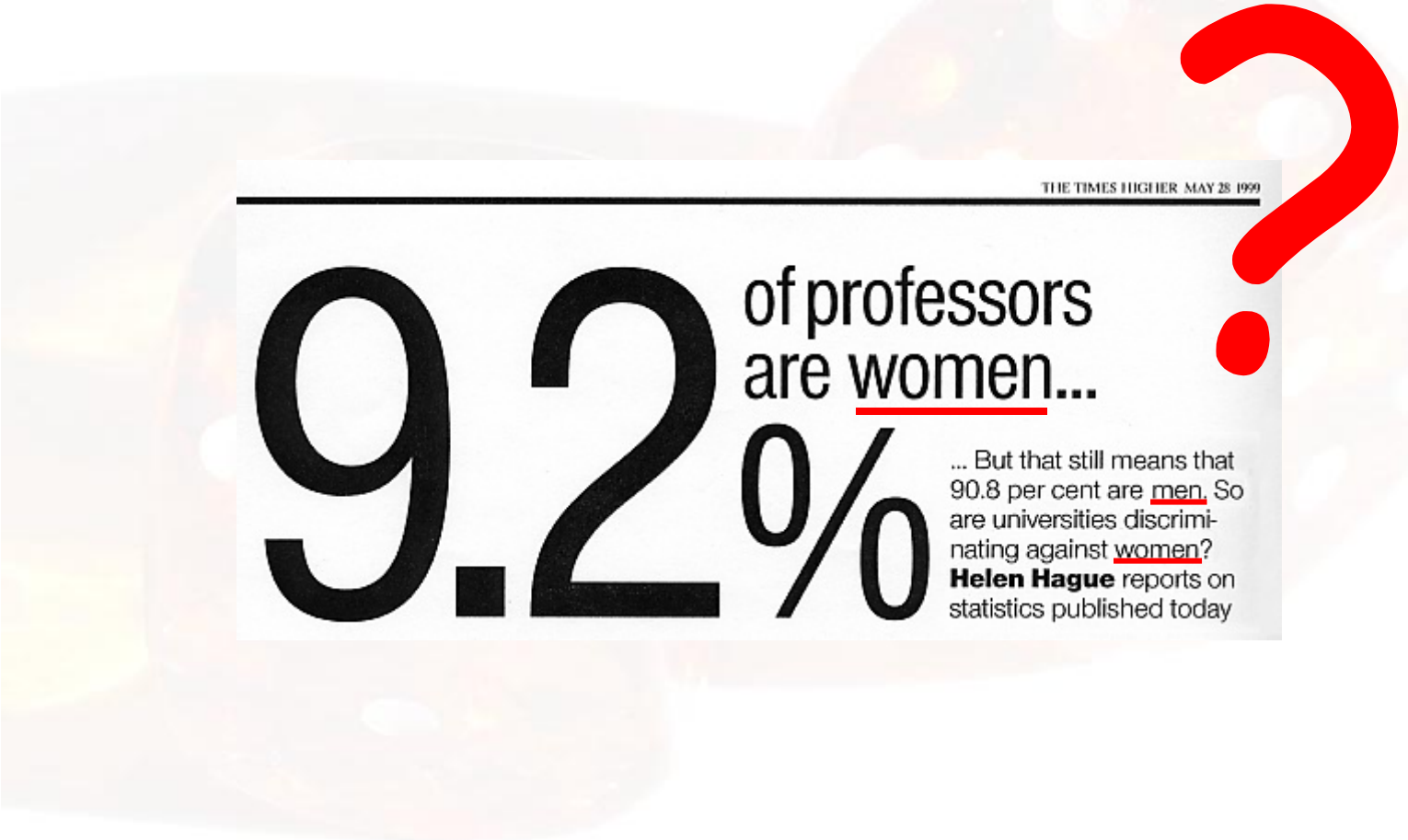
Samson Abramsky, Rajeev Alur, Simon Ambler, Christel Baier, Bard Bloom, Marcello Bonsangue, Stefano Cattani, Andrew Chan, Sibusisiwe Chiyangwa, Ed Clarke, Costas Constantinou, Sadie Creese, Pedro D'Argenio, Conrado Daws, Luca de Alfaro, Jaco de Bakker, Abbas Edalat, Doug Edwards, Amani El-Rayes, Stephen Gilmore, Michael Goldsmith, Dimitar Guelev, Rajeesh Gupta, Chrysafis Hartonas, Vicky Hartonas-Garmhausen, Boudewijn Haverkort, Holger Hermanns, Joe Hurd, Michael Huth, Jane Hillston, Bertrand Jeannet, Joost-Pieter Katoen, Klaus Keimel, Peter Knowles, Kim Larsen, Zhenyu Liu, Annabelle McIver, Nick Measor, Rashid Mehmood, Christoph Meinel, Carroll Morgan, Gethin Norman, Colin O'Halloran, Antonio Pacheco, Dave Parker, Doron Peled, Wojtek Penczek, Sylvain Peyronnet, Iain Phillips, Mark Ryan, Roberto Segala, Mike Shields, Vitaly Shmatikov, Sandeep Shukla, Markus Siegle, Jeremy Sproston, Georgios Theodoropoulos, Rick Thomas, Moshe Vardi, Fuzhi Wang, Pete Watkins, Irfan Zakiuddin, Bohumir Zoubek

EPSRC QinetiQ The Nuffield Foundation www.nuffieldfoundation.org BRITISH COUNCIL

PriSM Contributors



Women and Career Ladder



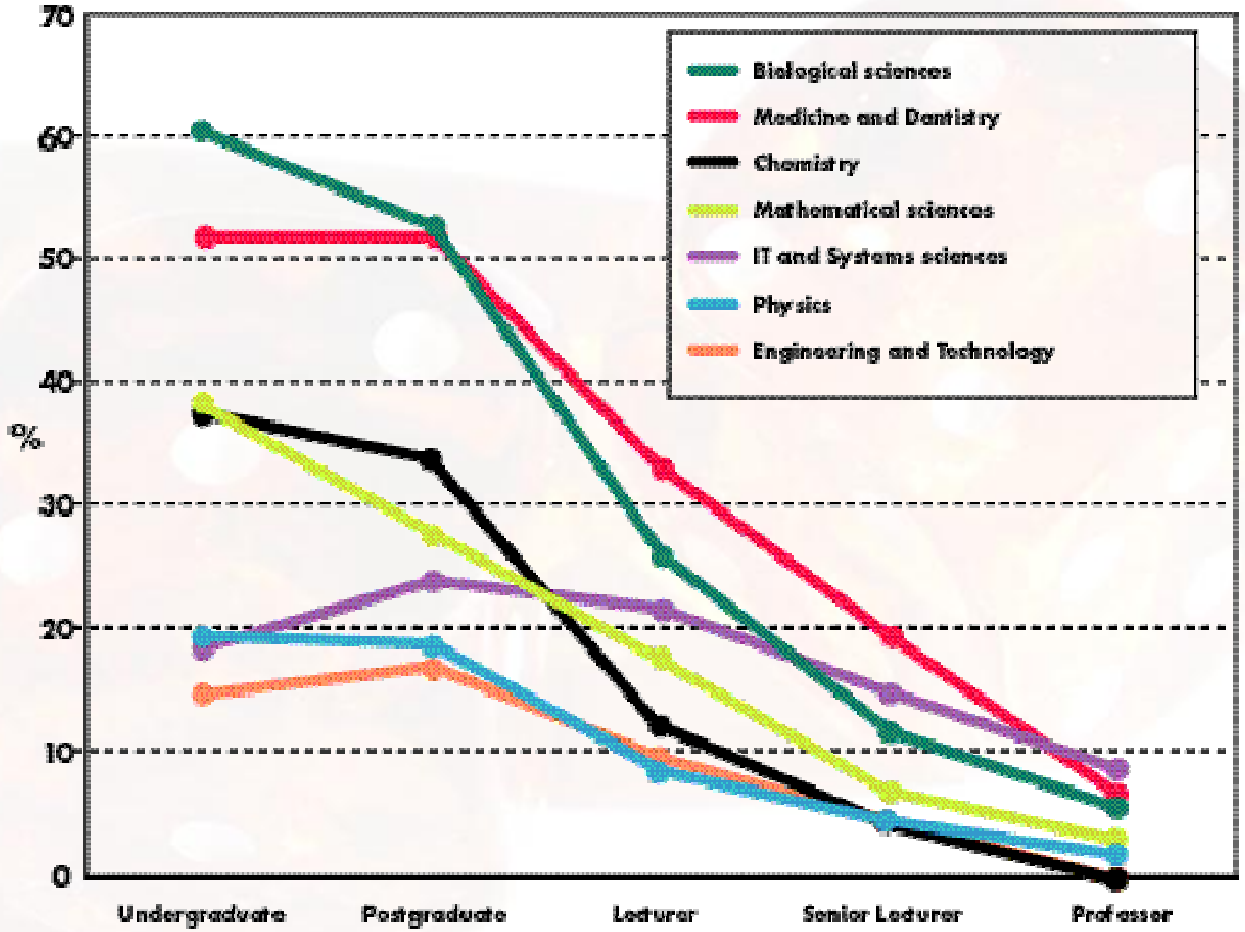
THE TIMES HIGHER MAY 28 1999

9.2% of professors are women...

... But that still means that 90.8 per cent are men. So are universities discriminating against women? **Helen Hague** reports on statistics published today

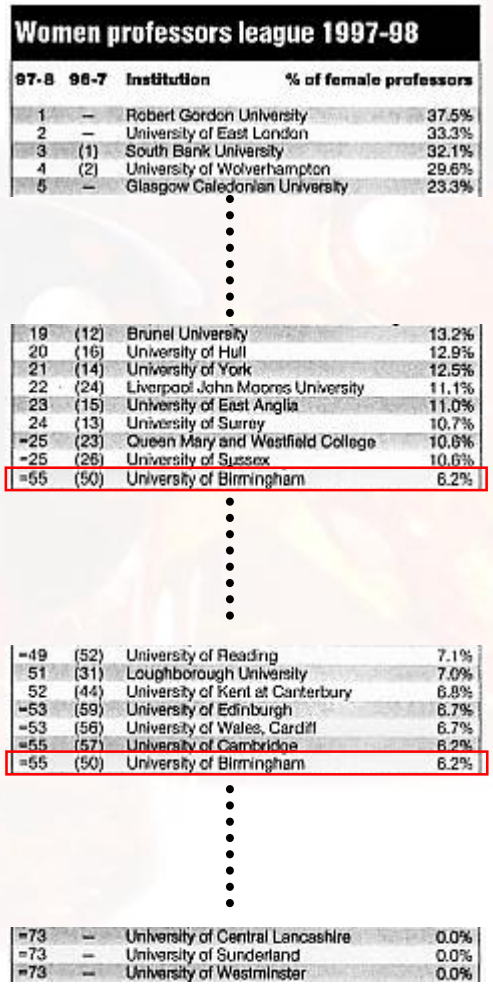
Source:The Times Higher May 25 1999

Women in UK Universities



Source:http://www.shu.ac.uk/witec/ukprojects/documents/conferencereport.pdf

Women Professors League Table



Women professors league 1997-98

97-8	96-7	Institution	% of female professors
1	–	Robert Gordon University	37.5%
2	–	University of East London	33.3%
3	(1)	South Bank University	32.1%
4	(2)	University of Wolverhampton	29.6%
5	–	Glasgow Caledonian University	23.3%
19	(12)	Brunel University	13.2%
20	(16)	University of Hull	12.9%
21	(14)	University of York	12.5%
22	(24)	Liverpool John Moores University	11.1%
23	(15)	University of East Anglia	11.0%
24	(13)	University of Surrey	10.7%
=25	(23)	Queen Mary and Westfield College	10.6%
=25	(26)	University of Sussex	10.6%
=55	(50)	University of Birmingham	6.2%
=49	(52)	University of Reading	7.1%
51	(31)	Loughborough University	7.0%
52	(44)	University of Kent at Canterbury	6.8%
=53	(59)	University of Edinburgh	6.7%
=53	(56)	University of Wales, Cardiff	6.7%
=55	(57)	University of Cambridge	6.2%
=55	(50)	University of Birmingham	6.2%
=73	–	University of Central Lancashire	0.0%
=73	–	University of Sunderland	0.0%
=73	–	University of Westminster	0.0%

Source:The Times Higher May 25 1999

It's about role models

MARIE SKLODOWSKA CURIE opened up the science of radioactivity. She is best known as the discoverer of the radioactive elements polonium and radium and as the first person to win two Nobel prizes. For scientists and the public, her radium was a key to a basic change in our understanding of matter and energy. Her work not only influenced the development of fundamental science but also ushered in a new era in medical research and treatment.

Source: http://www.aip.org/history/curie/contents.htm

Thank you for your attention