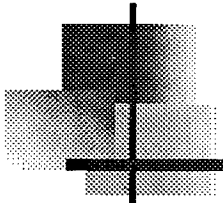


# An Odd Take on Formality and Testing

A Title Best Left Unannounced???



**John A Clark**  
**Senior Lecturer in Critical Systems**  
**Dept. of Computer Science**  
**University of York, UK**  
**[jac@cs.york.ac.uk](mailto:jac@cs.york.ac.uk)**

**Brunel** 26.03.2002



# Aim

---

- **Largely to pose some questions and highlight issues that may be useful in defining the relationship between formality and testing.**
  - **What are formal methods?**
  - **What's a spec?**
  - **What's a Fault?**
- **Some non-standard issues.**



# What are formal methods?

---

- A formal method typically comprises....
  - a notation for describing systems
  - calculus for reasoning about properties of systems (including refinement)
  - guidelines for refining.
- We may not be able to insist of all of the above components.
- The sorts of systems are changing and we will typically need to test them.



# Established Models

---

- **Formal spec + random or directed testing using spec as oracle. Test and check.**
  - **Degree of directness varies a little. E.g. Z specifications or program annotation languages (e.g. Spark annotations).**
  - **You can view formal specifications as being created by some of the evolutionary testing based tools (e.g. for attaining structural coverage).**
- **Specifications generate formal constraints that define (after solution) test cases.**
  - **Z specs or after symbolic execution from programs in the form of augmented path traversal conditions etc.**



# Other Established Models

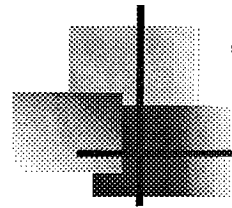
---

- **Worst case timing analysis. Some degree of formality applies here too.**
- **Given specific formal assertions about subpath feasibility a calculus can provide good bounds on worst case execution time.**
  - **In the absence of formal proof it is really hard to see how standard approaches to testing (iother than evolutionary approaches can really hope to exhibit extreme execution times.**



1234: What are we testing for?

---



# Testing for what?

---

- **Academically - concentration on small things**
- **Also emphasis on functional correctness**
- **But there are many other interesting things to be bothered about**
  - **timing, stress testing, resource usage, modifiability (???), security....**



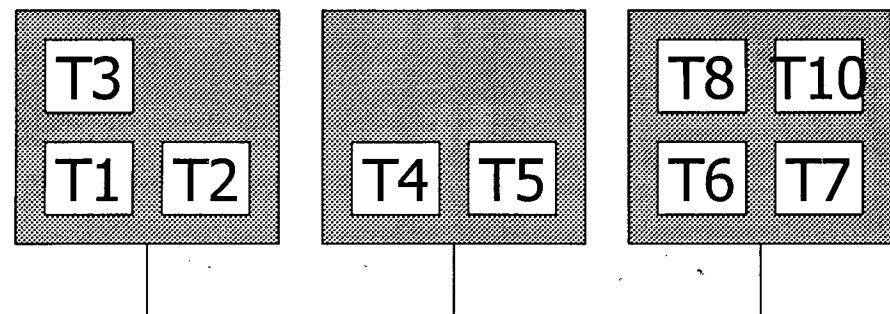
# Where Refinement is NP-hard

---



# NP-Hard Refinements

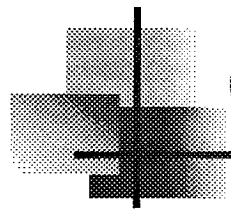
- **Common to say that formal specification is hard and refinement is harder.**
- **But sometimes we may lack a real refinement method.**
- **For example, for distributed systems the some allocations of tasks to processors may well result in tasks missing deadlines**





And what planet are you on?  
**What quantum world are you in?**

---



# Quantum formalisms

---

- **We have a good understanding of what programs do.**
- **Matrix algebra**
  - **unitary transformations**
  - **projections (state collapse)**
- **Unfortunately given a specification it seems very hard to do the refinement!**
  - **We can map down certain constructs onto gate level but general refinement is hard.**
  - **But qQuantum progs are probabilistic and so 'correctness' needs interpretation.**
- **Need to consider faults in quantum world.**



# When the Refinement Isn't

---

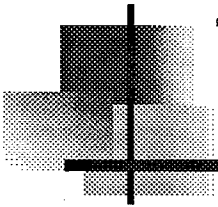


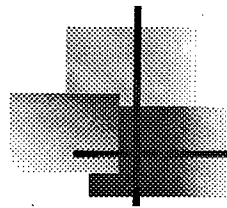
# Relaxing Rigidity

---

- **You don't want too much detail in the specification.**
- **You may be prepared to deviate from the specification in the name of practicality**
  - **Formal spec might have reals.**
  - **Implementation might have fixed point.**
- **What are the implications for testing?**

Where we can specify the  
individual behaviours but not  
their sum

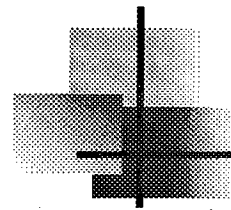




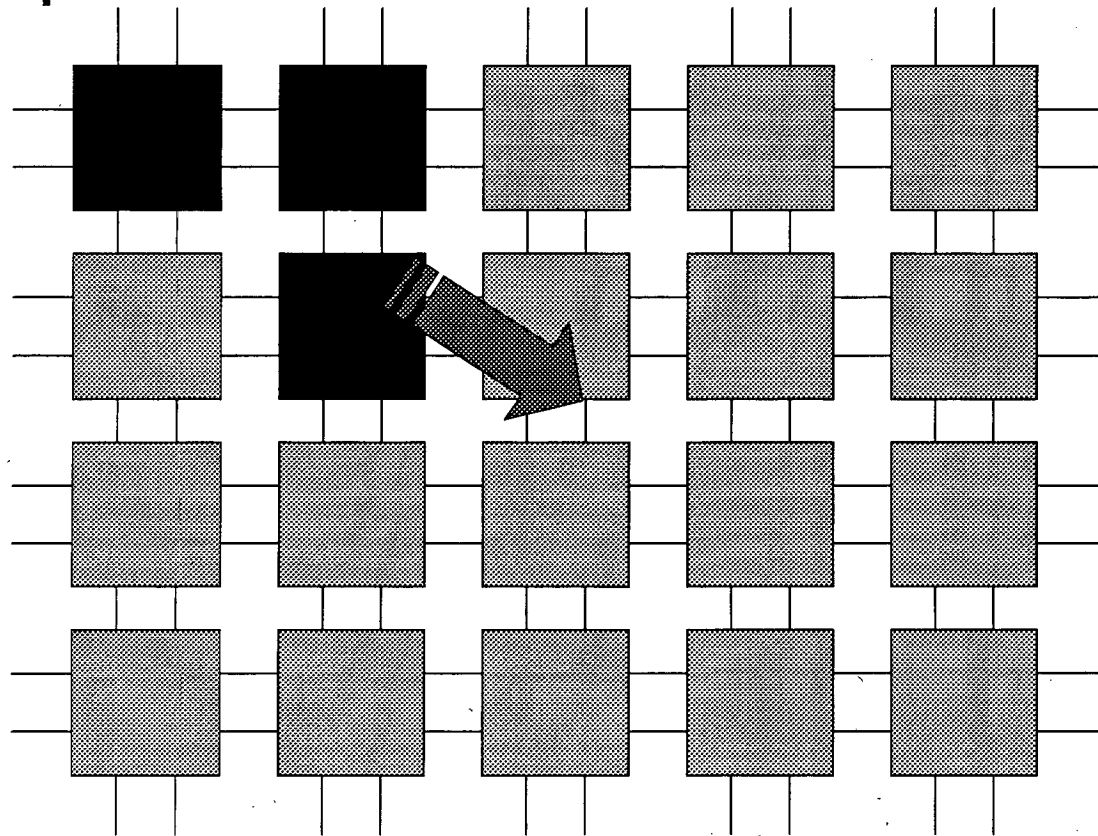
# Systems with Emergent Properties

---

- **There are some systems we can define formally but not refine well.**
- **Systems comprising many interacting components the behaviour of each defined typically by some set of rules**
  - **For example Conway's 'Game of Life'**
  - **Gives rise to the concept of 'gliders' passing across the screen**
  - **BUT - given the concept of gliders we lack a refinement approach to generate the rules for each block**



# Systems with Emergent Properties



Each node  
is a simple  
state machine  
with state and  
outputs  
defined by state  
and  
NSEW inputs



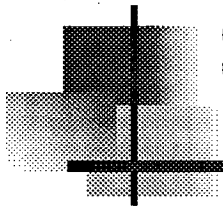


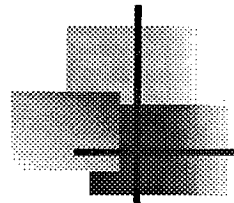
# Emergent properties

---

- Reasoning and testing these sorts of systems and their emergent properties will become increasingly important.
- Game of life is a toy example but it illustrates a point.
  - More interesting examples of emergent properties might include load balancing, provision of quality of services etc.
- What properties would we actually want to specify, how might we test these, and what are the interactions between specification and testing?

It just looks a bit non-  
standard





# Non-standard Architectures

---

- **What work is being done in FPGA testing and links with formality?**
- **FPGAs are industrially very important. Some work on high level formalisms and their refinement/compilation to FPGA hardware.**
- **What else can be done?**
  - **FPGAs are built for fault injection. A program is a BIG fault.**
  - **Break free of the standard computational paradigm.**



Where the spec comes last!

---



## **Data-Tests-Data-Tests-Data....**

---

- **Reverse engineering seeks to impose a snappy description on test data obtained.**
- **Good example here is Michael Ernst's work on specification generation**
  - **start with loads and loads of predicates that could be invariants and remove those that are inconsistent with the test results.**
  - **What's left is a 'spec'.**
- **But can use directed testing to attack those that remain and so improve matters.**



# Neural Network Descriptions

---

- **Neural nets-work! But what are they doing.**
- **For certain types of system their use is arguably advantageous technically but reasoning about them and testing them is difficult.**
- **Work at present aims to get NNs to provide an 'explanation' of their decision making - e.g. a formal characterisation of classification groupings.**
  - **These are 'specs' or 'assertions' and opportunities for falsification testing.**
  - **Or actually just testing!**



# When Nine out of Ten Testers Agree (It must be wrong)

---

**The testing of statistical properties**

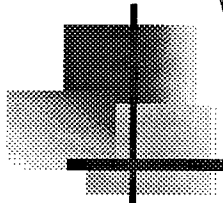


## What is a Test of a property?

---

- **For systems where formal requirements are specified in *statistical terms* no one 'test' will suffice.**
  - **Need a test sequence**
  - **Actually sets of test sequences, since we need to allow for oddities such as initial conditions and lack of strong connection in behavioural graphs (e.g. initial operations may preclude interesting states ever being reached).**
- **How do we go about getting such test sequences?**
  - **Typical approaches would be random from some distribution (e.g. operational). What else can be done?**





# What is a Fault?

---



# What is a Fault?

---

- **This varies.**
  - **For statistical requirements we need some form of confidence interval approach (or similar)**
  - **for rule based networks, a fault could be a node with a disturbed rule set.**
  - **Or possibly a perturbation in the input distribution or initial state assumptions?**
  - **Other interpretations?**



# Summary

---

- **Aimed simply to indicative some ways of deviating from standard views on formal methods and testing.**
- **Some areas are not well pinned down yet.**
- **How far should we limit ourselves?**