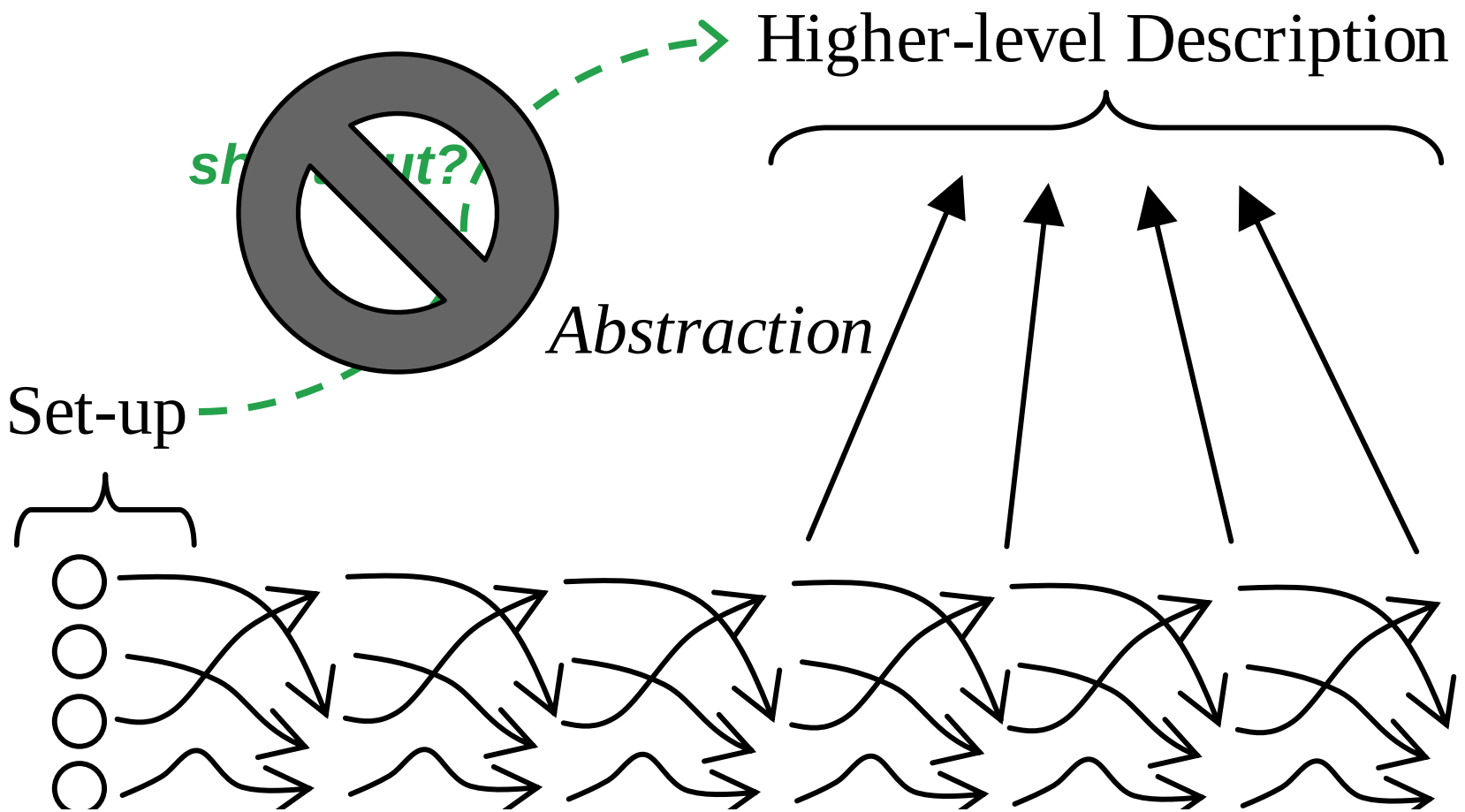
The Necessity of Post-construction Management for Complex Systems

Bruce Edmonds
Centre for Policy Modelling,
Manchester Metropolitan University

Part 1:

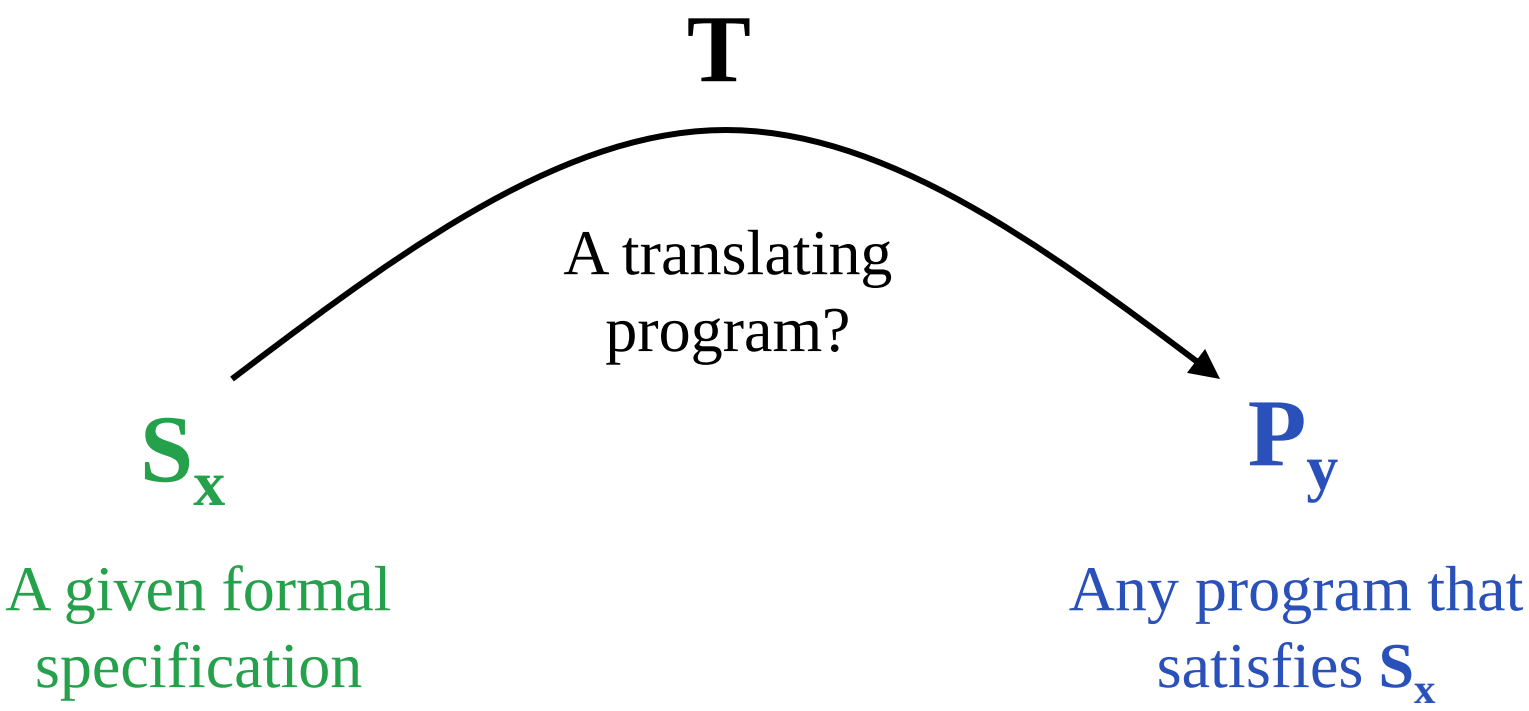
"*It is impossible to ensure that complex systems do not have unforeseen emergent properties*"
Alan Winfield

(Weak) Emergence



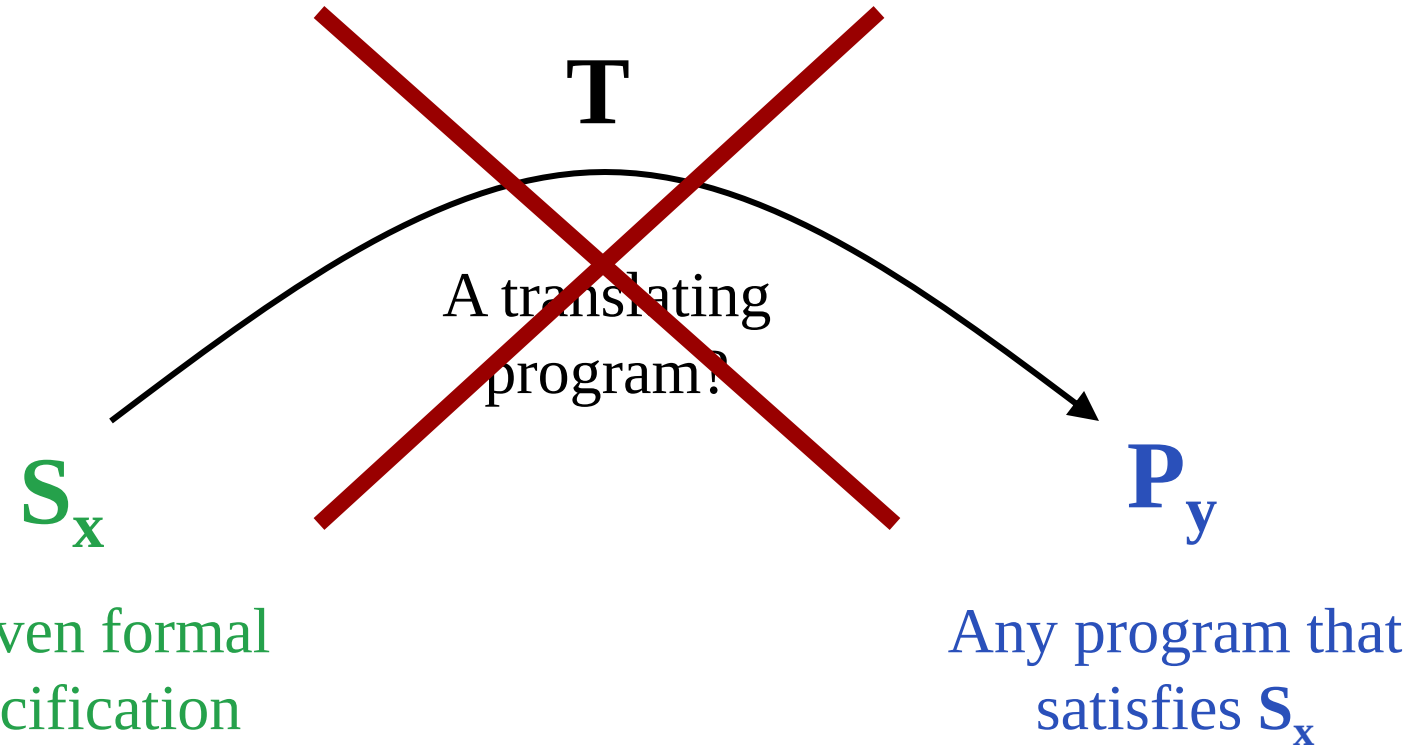
The *Programming Problem* Posed

Is there a general, effective or systematic method of finding a program/system that satisfies a given specification?



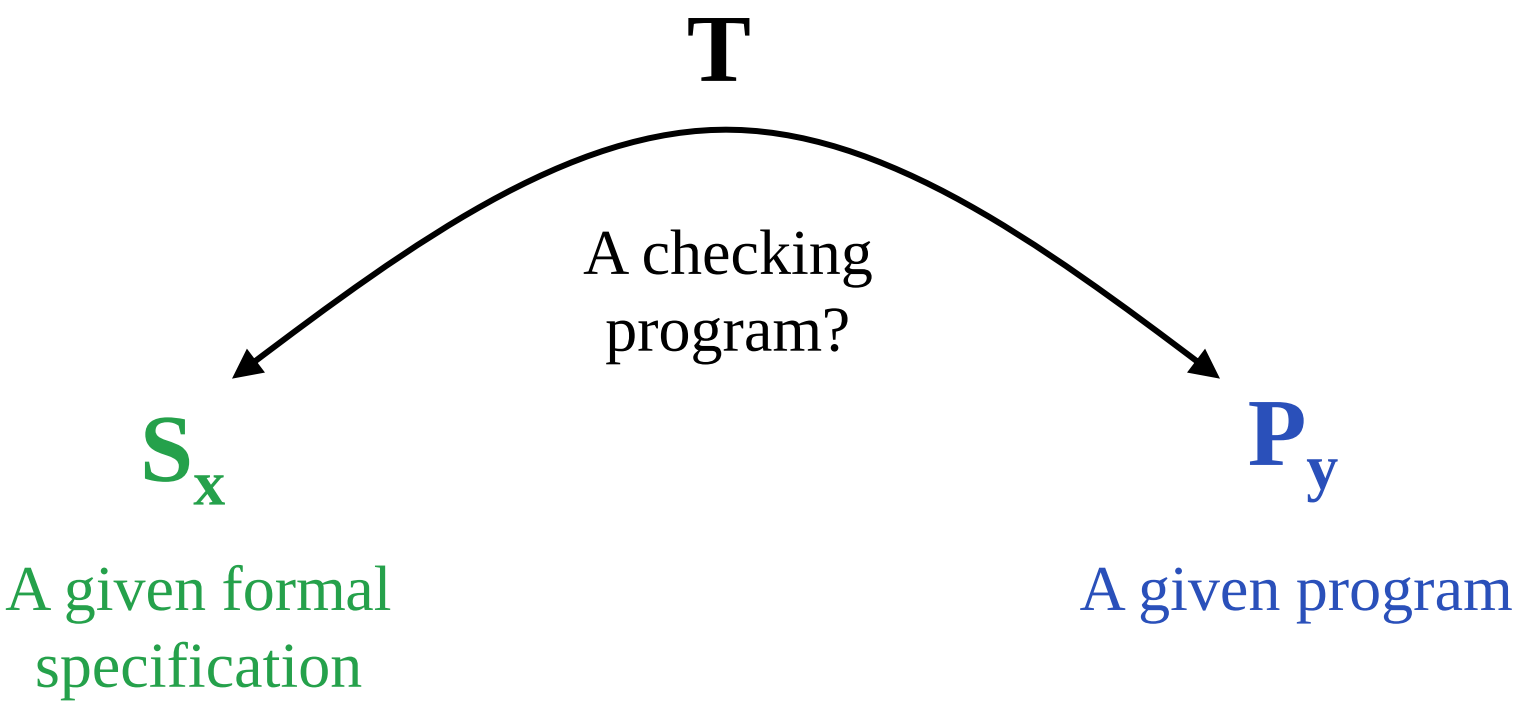
The *Programming Problem* Answered

No, if the class of systems is complex enough



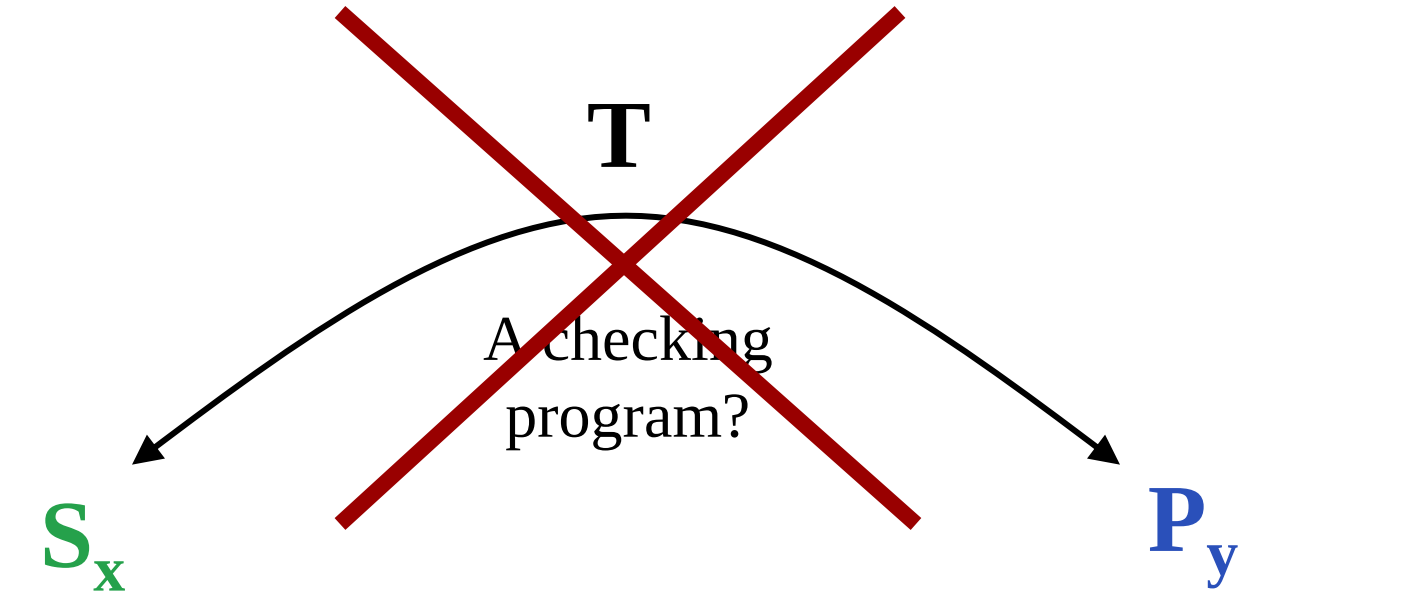
The *Checking Problem* Posed

Is there a general, effective or systematic method of checking whether a given program/system satisfies a given specification?



The *Checking Problem* Answered

No, if the class of systems is complex enough



Proof Sketch
(programming problem answer)

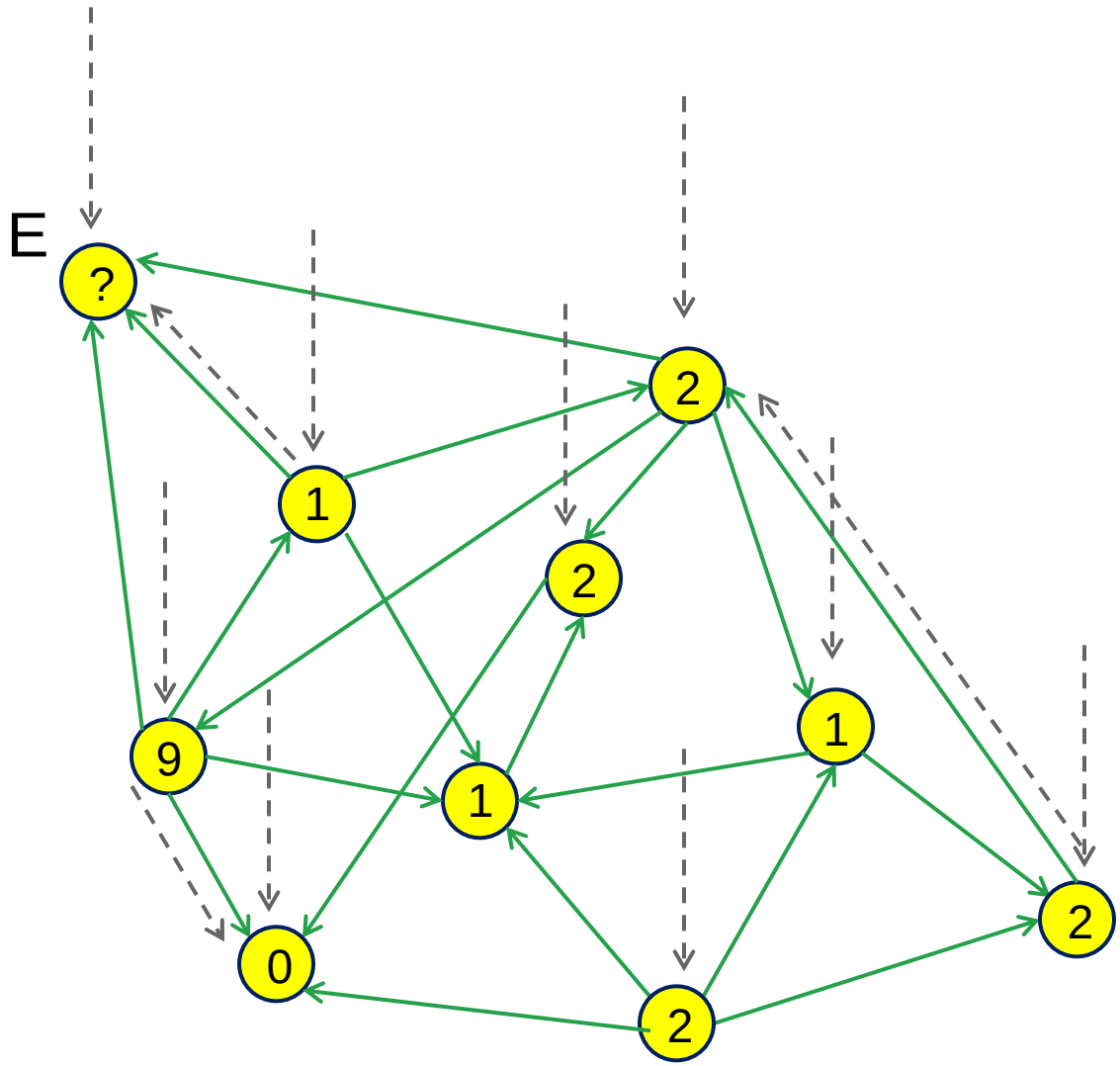
Define the "n^{th} limited halting problem", $LH_n(x,y)$, as:
Does P_x *ever halt with input* y *where both* $x,y \leq n$*?*

Each $LH_n(x,y)$ is computable as a finite lookup table

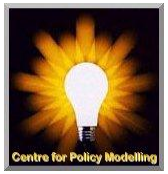
$LH_n(x,y)$ is effectively expressible in a language with arithmetic via the construction in (Gödel 1933) as SH_m - that is, m here is computable from n

Now if there *were* a translating program, **T**, then: given x and y; let z=max(x,y); compute SH_z; use **T** to find a program to compute $LH_z(x,y)$ from SH_m; and use this to find whether $P_x(y)$ halts; but this is impossible (Turing 1936).

An Illustration of what happens in a GASP system



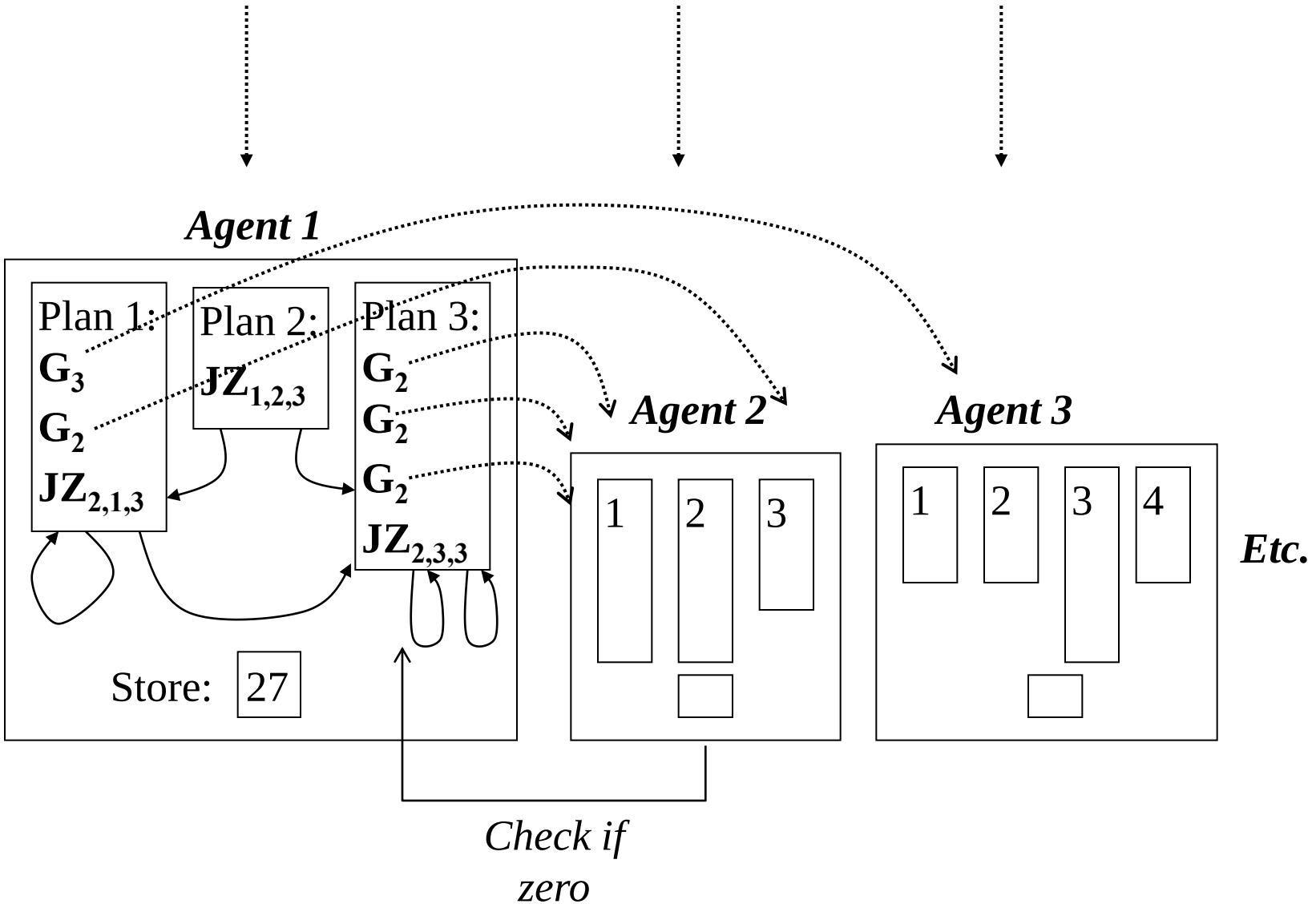
Example: A "Simple" Class of System – "*GASP" systems*



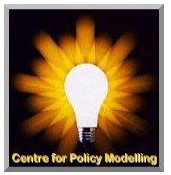
- **G**iving **A**gent **S**ystem with **P**lans
- Fixed number of agents: A_1, A_2, …A_n
- Each agent, A_x, has
  - a store, S_x
  - a fixed number of plans: P_{x1}, P_{x2}, …
- Each Plan, P_{xy}, composed of instructions:
  - A fixed number of "give one to"
  - And one test: If S_i is zero then do plan *j* next, otherwise plan *k* next
- Each time click, all do:

  get 1 unit; use current plan to: [do giving (while they have); test others; note next plan].

An illustration of a GASP system

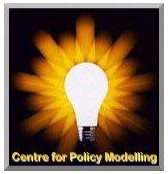


Thus *all* that happens in GASP systems is:



- That agents have a fixed set of *very simple* plans/programs
- Their state is the amount in their store and the index of the current plan
- All they do is give fixed amounts to other agents accorind to their current plan
- All they can perceive is whether an other agent's store is zero or not…
- …which determines the index of the next plan in a fixed way

Lessons from GASP systems



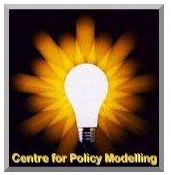
- They are Turing Complete, that is they can simulate *any* computer program
- Hence many questions about their behaviour are (in general) *undecidable*
- E.g. one will *not* (in general) be able to check which GASP systems satisfy some specifications from its set-up
- In other words one cannot hope to rely on *design* for some kinds GASP systems
- Many systems we are considering are much more complicated than GASP systems
- So we cannot rely on design with them either, unless we *establish that* we are considering a special case in some way

Part 2:

"*Is there a meta-theory of emergence?*"

Rebecca Mancy

Some (roughly evolutionary) Examples



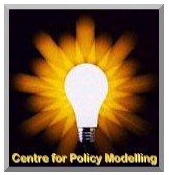
1. Biological Evolution
2. Development of theories/models in science
3. Children learning to walk, etc.
4. Artistic Creativity

- Whilst not pure BVSR (Blind Variation and Selective Retention)...
- ...all include the ad-hoc generation of hypotheses/possibilities...
- ...which are later checked/validated/reinforced
- None have a “mechanism” for rule discovery
- All are open to criticism/review/deletion (c.f. Fiona Polack’s talk)

but evolution does *not* necessarily produce systems that are...

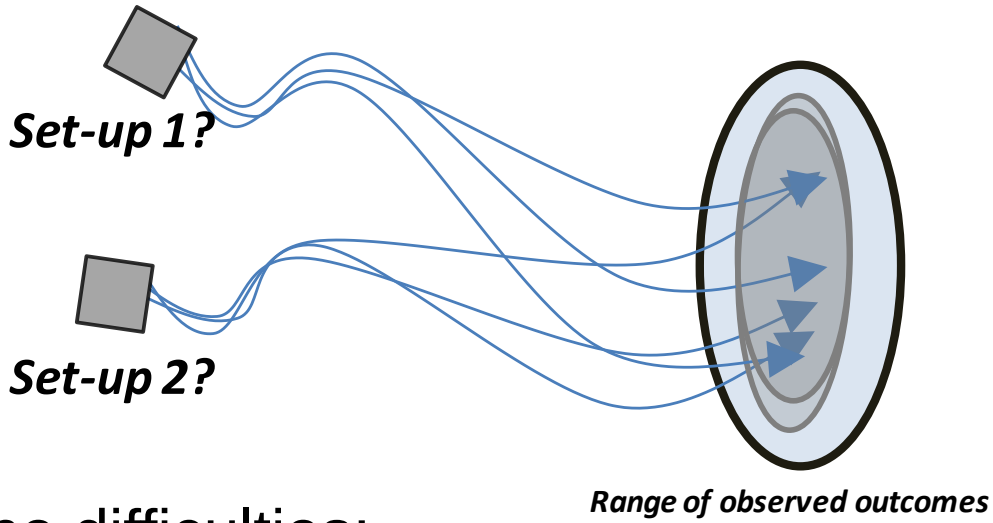
- comprehensible
- verifiable
- safe
- robust
- scalable
- distributed
- generally applicable
- adaptable
- have simple parts

Easy and *Hard* Complexity Problems



- The "easy" problem: producing examples of emergence (complex behaviour) from the interaction of simple parts
- The "hard" problem: finding how to set-up a system to produce a given kind of emergent behaviour using the interaction of some distributed parts
- Progress with the easy problem only helps with the hard problem if we can solve the *reverse engineering* problem

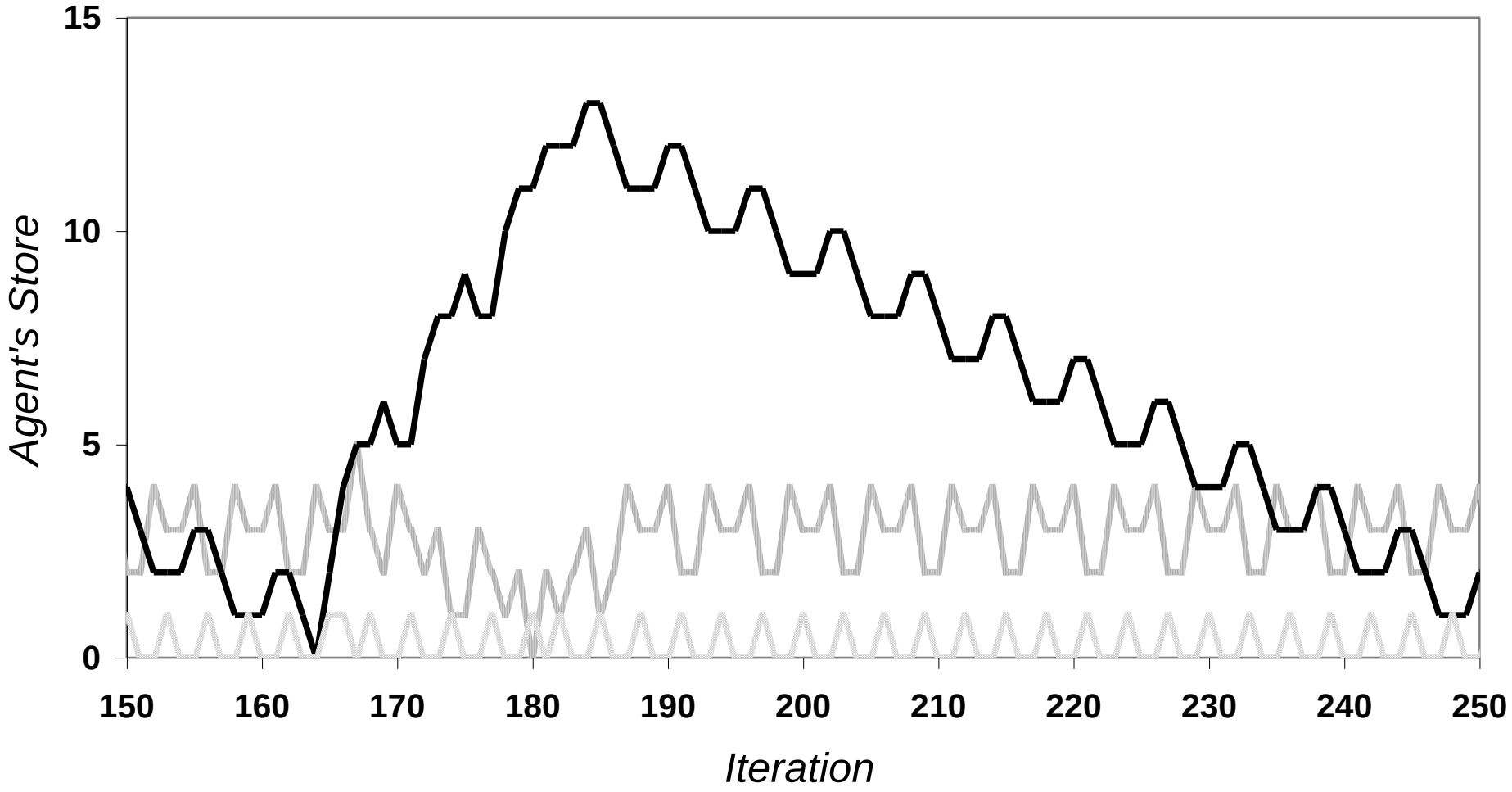
The difficulties of reverse engineering



Some of the difficulties:

- The space of possible set-ups is (almost always) HUGE
- You cannot derive the set-up from the results
- Any set-up → result connection is an empirical matter
- You often cannot be sure you even when you find one
- You the stability of a setup is uncertain
- The role of context is unending

Example evolved GASP which *counts* an 89-cycle (max: 5 nodes, 5 plans each, 3 "give" instructions per plan)

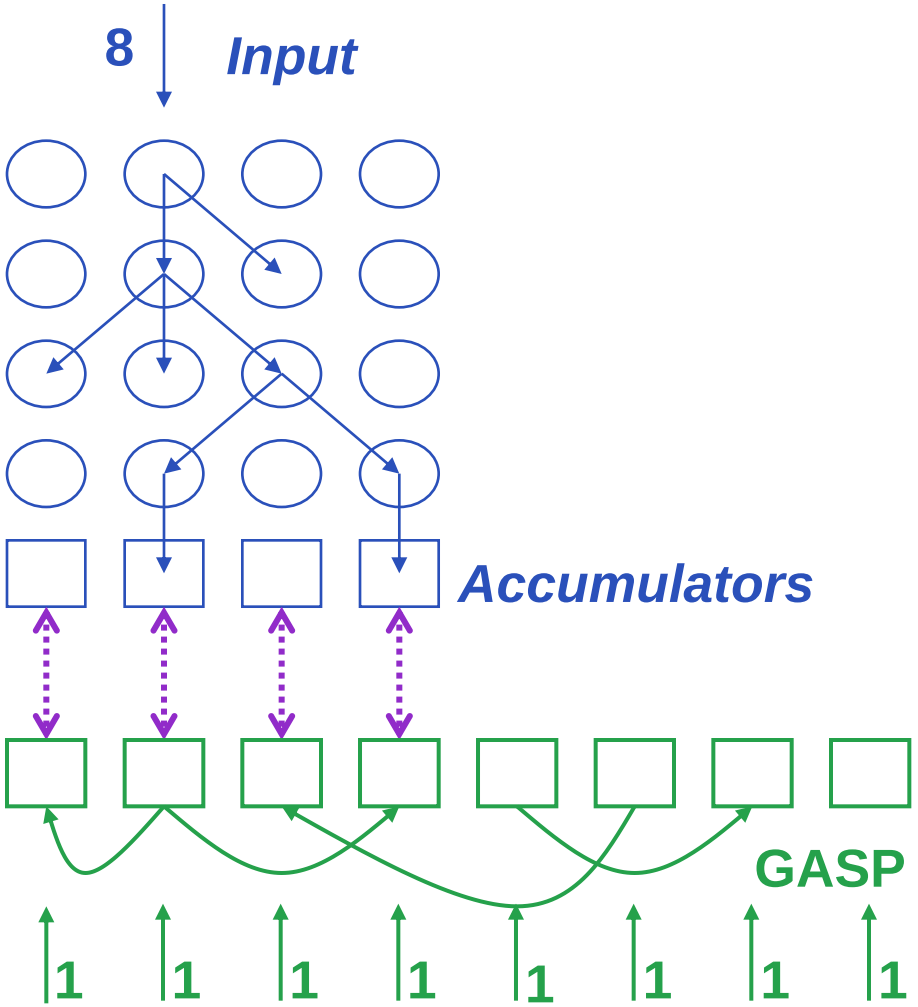


What this GASP looks like

- 1, 1, [], 5, 3, 2
  1, 2, [], 4, 2, 1
  1, 3, [], 5, 2, 4
  1, 4, [1], 3, 2, 4
  1, 5, [3 4], 2, 1, 5
  2, 1, [], 5, 4, 3
  2, 2, [], 4, 2, 4
  2, 3, [3 6 6], 1, 5, 1
  2, 4, [6 5 4], 2, 2, 3
  2, 5, [6 3 3], 3, 3, 2
  3, 1, [], 4, 3, 1
  3, 2, [6], 5, 3, 4
  3, 3, [3 4 2], 3, 3, 4

- 3, 4, [4 4 5], 1, 3, 5
  3, 5, [3 6], 1, 2, 1
  4, 1, [], 1, 3, 3
  4, 2, [], 1, 5, 5
  4, 3, [3 3], 3, 3, 5
  4, 4, [2], 1, 3, 1
  4, 5, [3 2], 5, 5, 4
  5, 1, [3 2 2], 5, 3, 5
  5, 2, [1 6], 2, 3, 1
  5, 3, [3 1 5], 2, 2, 4
  5, 4, [1 2 1], 5, 1, 4
  5, 5, [4 4], 4, 4, 4

Example (toy) Problem for a GASP: shoring up (artificial) avalances

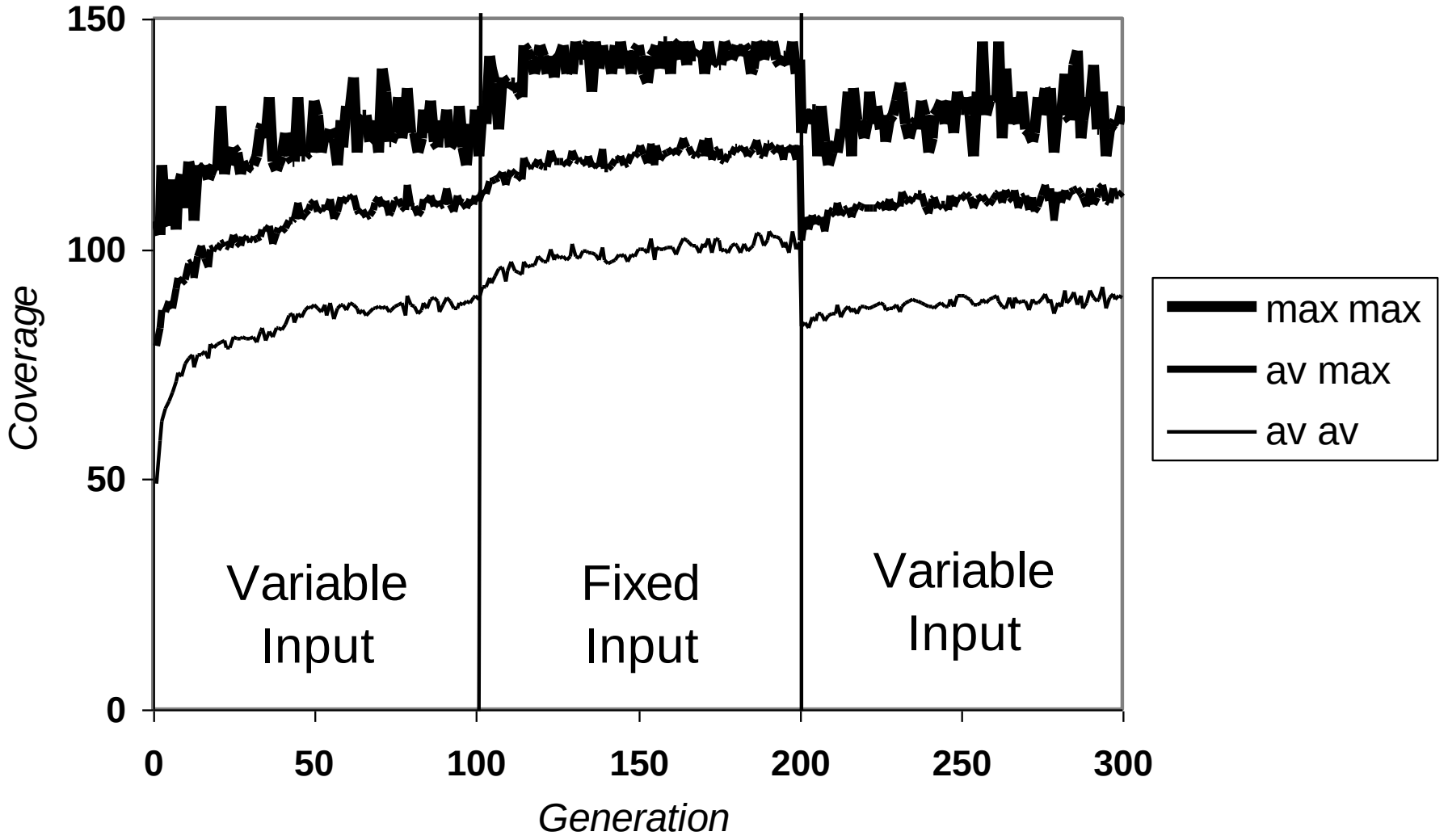


- Input position can be fixed or variable

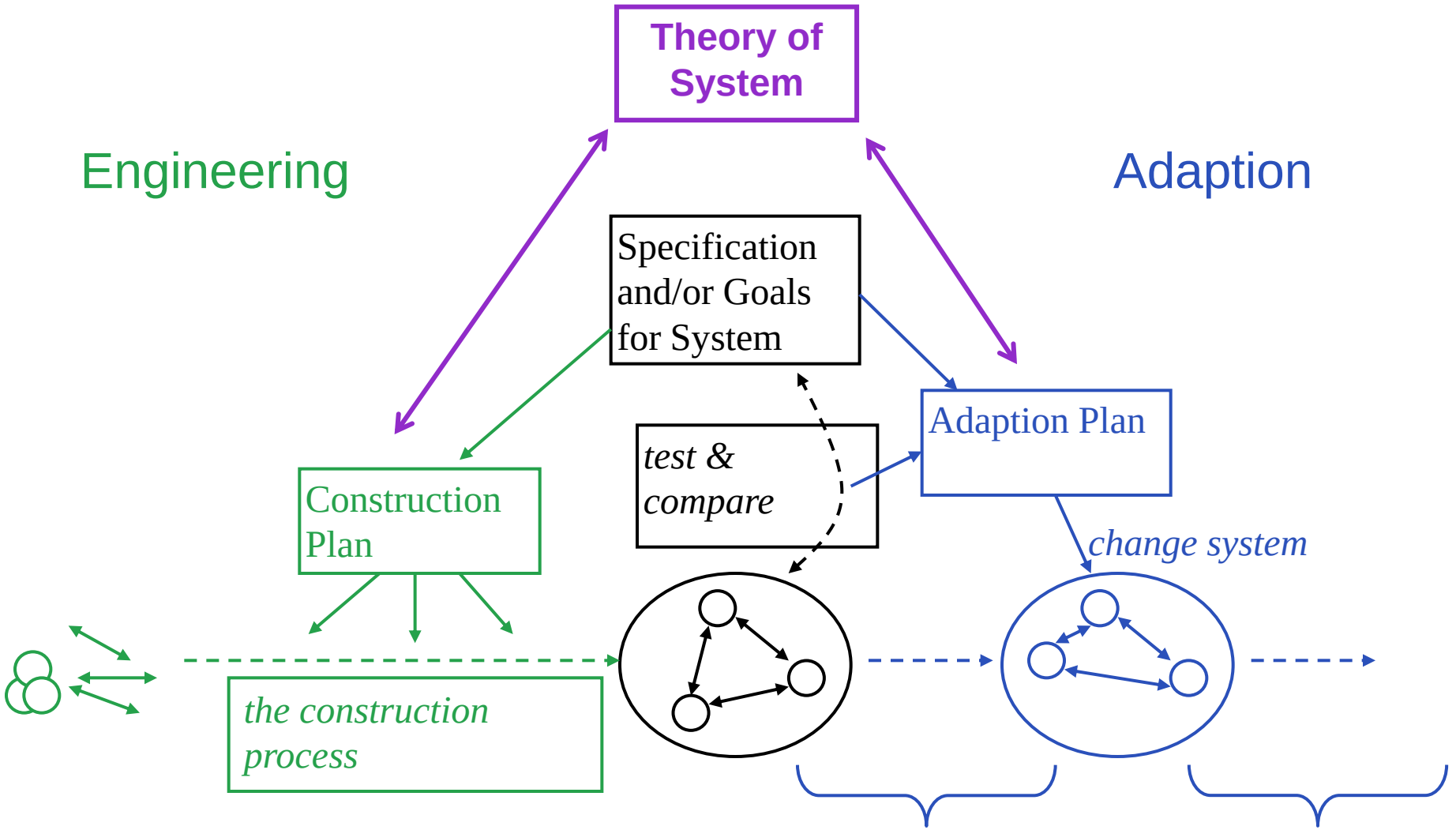
- Avalances generated by self-organised critcal system which is inherently difficult to predict

- GASP evolved to distribute units to equal units in avalance accumulators - it races the SOC

Results of Evolution (31 runs of the evolution of GASP populations of 12)



The place of a system theory



Such system theory…

…will necessarily be to a large extent inductive rather than deductive.

- It may orginate from different sources (the design, analogies with natural systems, by observation of the system etc.)
- but it will need to be tested using the classic experimental methods
- involving repeated and independent attempts at disproving theories
- and explicitly recording/revising the conditions of application.

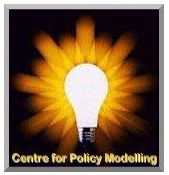
The Classic Experimental Method

- Theory developed empirically (not *a priori*)
- Applied using well-validated processes and tools
- How the theory can be applied using what approximations is empirically established
- The conditions under which a theory can be safely applied (and how) developed over time
- Useful properties can only be *deduced* after theory has been validated
- Has worked in messy systems (in science) where there is little that can be completely generalised

What this might give us…

- Explicit, relevant and testable models/hypotheses concerning the properties of certain CDS
- With sets of conditions under which it has survived trials/testing (and the extent of that success)
- Also a set of situations where the hypotheses failed, to indicate the *limits* of its applicability
- Giving confidence and guidance to those who wish to use these systems
- From which inferences can be made
- A *scientific* basis on which to build sound engineering practices

E.g. Engineering a bridge

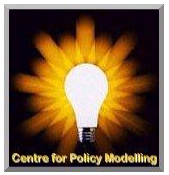


- Use of well-validated general designs and strategies (e.g. arches, columns, suspension)
- Multiple approximate calculations (maximum stress, weight, compression)
- Use of well-validated components or components made using well-validated techniques (e.g. standard girders or cable)
- Simulations of the set-up (e.g. oscillations)

…still the unexpected may occur - no illusion that *design proof* can be used on whole systems to achieve reliability

A Comparison of Approaches

(at present time)



Formal/Inferential

- Optimality criteria for success
- Well-developed methodology/tools
- Taken from formal sciences
- Limited applicability
- works for small components
- As a check with simplified models

Inductive/Experimental

- Sufficiency criteria of success
- Methodology/tools need improving
- From natural sciences
- Potentially wide applicability
- On real systems
- No certainty
- An eternal task

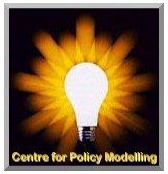
Part 3:

"*It's not a safe system – it's that it's operated safely*"
Fiona Polack

System Farming

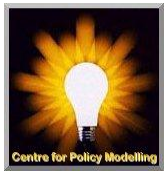
- That one has (through happenstance, evolution, or complexity) systems one does not fully understand
- What one has to do is grow and tend these systems so they don't die
- Continually monitoring them; fixing them; re-modelling them
- Know when you have to put a system down and start afresh
- Lots of crisis management
- Not much engineering status – its *mundane*

System Farming – *a shift of emphasis*



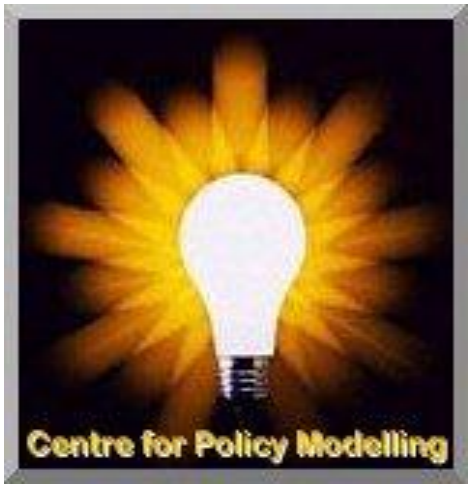
1. Reliability from experience *rather than* control and planning of construction
2. Post-construction care *rather than* prior effort
3. Continual management *rather than* one-off effort (*c.f. co-development of simulations – Andrews et al.*)
4. Multiple fallible mechanisms *rather than* one reliable mechanism
5. Monitoring *rather than* prediction
6. Disaster aversion *rather than* optimal performance
7. Partial *rather than* full understanding
8. Specific *rather than* abstract modelling
9. Many models *rather than* one model
10. A community effort *rather than* individual effort

with Complex Distributed Systems...



- formal proof will *not* play a *major* role in the discovery of patterns in emergent systems;
- there will *not* be any "magic bullet" techniques with universal applicability for engineering CDS;
- the validation, management and adaptation of CDS can *not* be treated as secondary matters that can be eliminated by good engineering;
- engineering will *not* sufficient on its own but rather a post-construction observation, modelling and intervention approach will be needed
- we may end up more like farmers than engineers, *system farmers* which may not have as much status but might be a lot more effective

The End



Bruce Edmonds

bruce.edmonds.name

Centre for Policy Modelling

cfpm.org