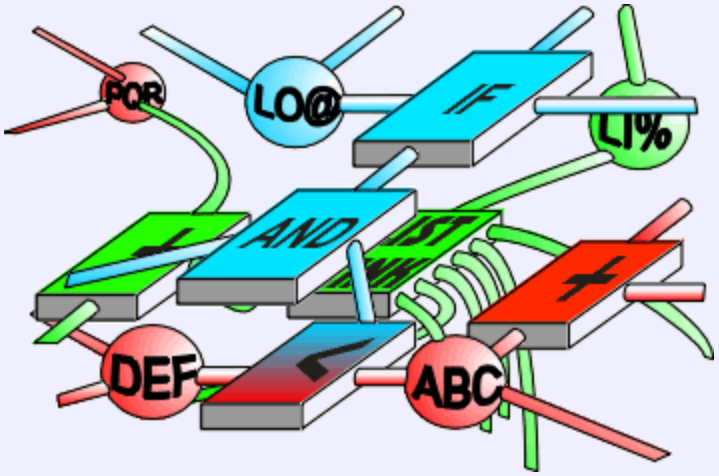
...But is it Scientific?



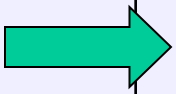
Scientific?

To answer that question, we need to look at the operation of a sequential machine

No Connection

In the stream of instructions, tests are performed and then actions are executed, with the assumption that the actions will not invalidate the tests performed previously.

As each new instruction is executed, the path back to previous instructions is lost.

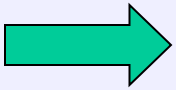


```
IF A > 5
  THEN BEGIN
     ..........
     IF B = 3
       THEN BEGIN
        ..........
        PROC_C
        ..........
```

No Visible State

The state of a sequential machine is bound up with the value in its program counter - what instruction it is about to execute. The result of this execution could take it forward or backward in the sequence of instructions.

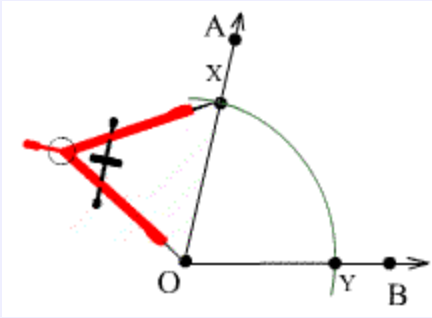
The lack of visible state prevents other parts of the machine autonomously responding to changing states.



```
again:  LOAD    loc_a,1
        LOAD    loc_i,1
        LOAD    reg1,loc_a
        INC     reg_1
        LOAD    loc_a,reg_1
        LOAD    reg_2,loc_i
        CMP     reg_2,101
        JNZ     again
        .......
```

Program as Proof

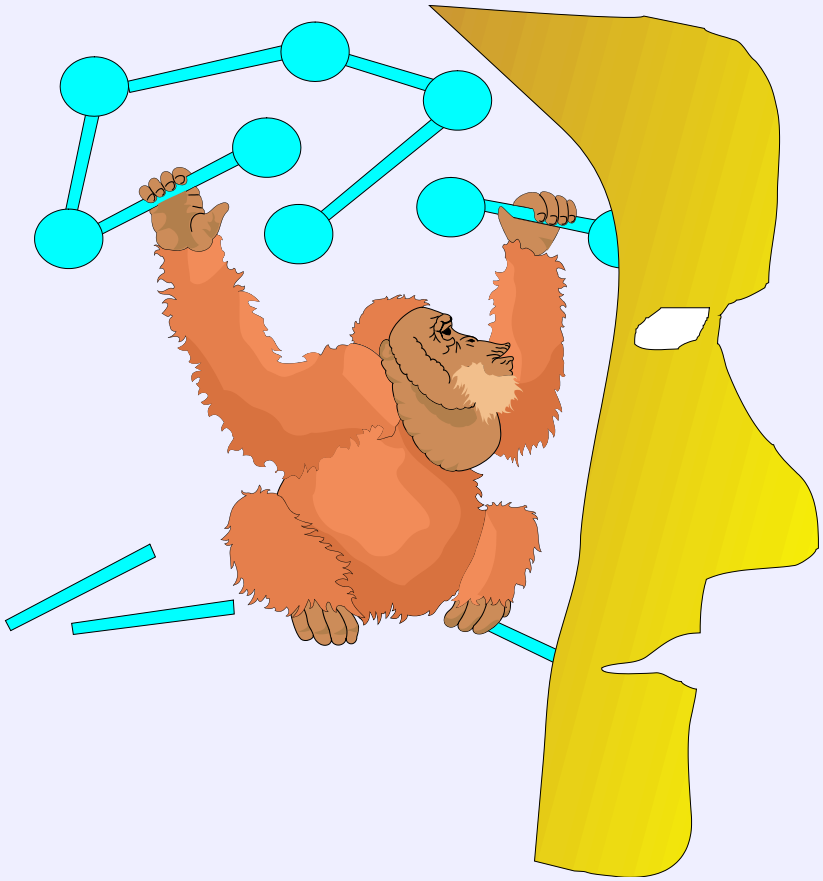
A program is sometimes likened to the proof of a theorem of Euclidean geometry - both seem to share a set of instructions to be carried out to reach a goal.

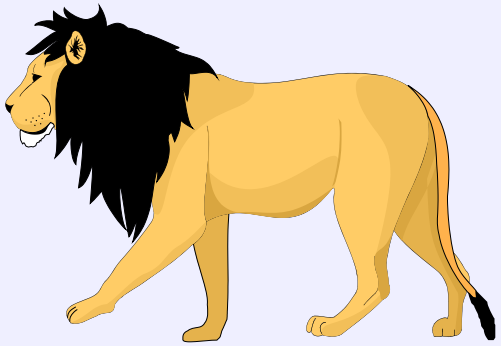


The geometric proof is meant to be taken up by an active system with memory - another person - who then uses the prior instructions to support what follows.

A sequential computer executing instructions does not remember what it did in order to support what follows - it has no connectivity.

Active Structure

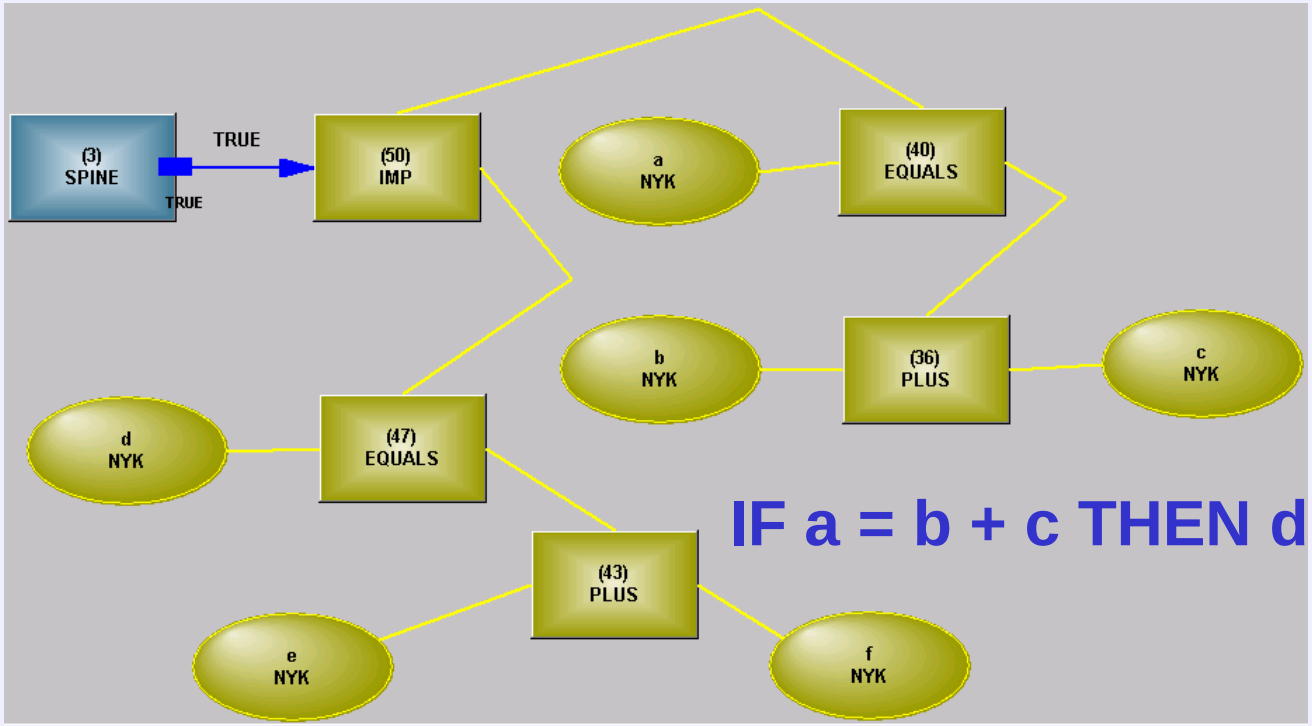




Active Structure is dynamically extensible - that means it is all the same stuff from the root of the structure - no compiled structure with a bit added on as an afterthought

Is it Logical?

Did we mention we use sentential logic as the connection between test and action - if something goes wrong, the system can reason about what must have happened - in the same way that we can.



Sentential Logic

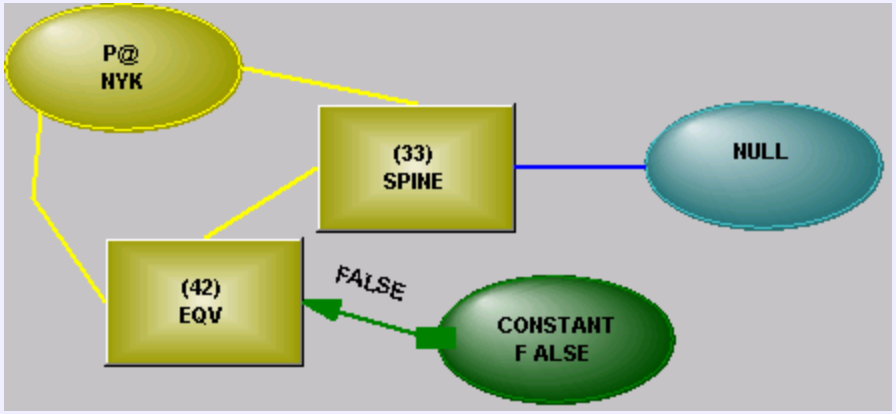
When Plato wrote down the rules for propositional logic, he wasn't inventing something, he was trying to write down how we reason - if we use those rules to drive the logical aspects, then the system works closer to the way our reasoning works, and it should be easier to understand and to interface with.

As Godel showed, what Plato had written down was a simplification and a subset....

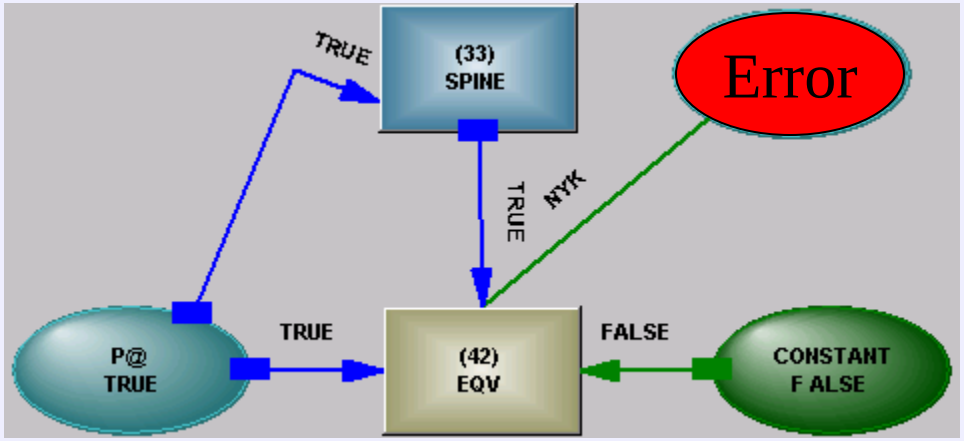
Godel's States

Godel showed that even simple axiomatic systems require a minimum of three states to represent the conditions that can occur. He used the example of the logical variable P standing for the statement P Is False.

In network terms



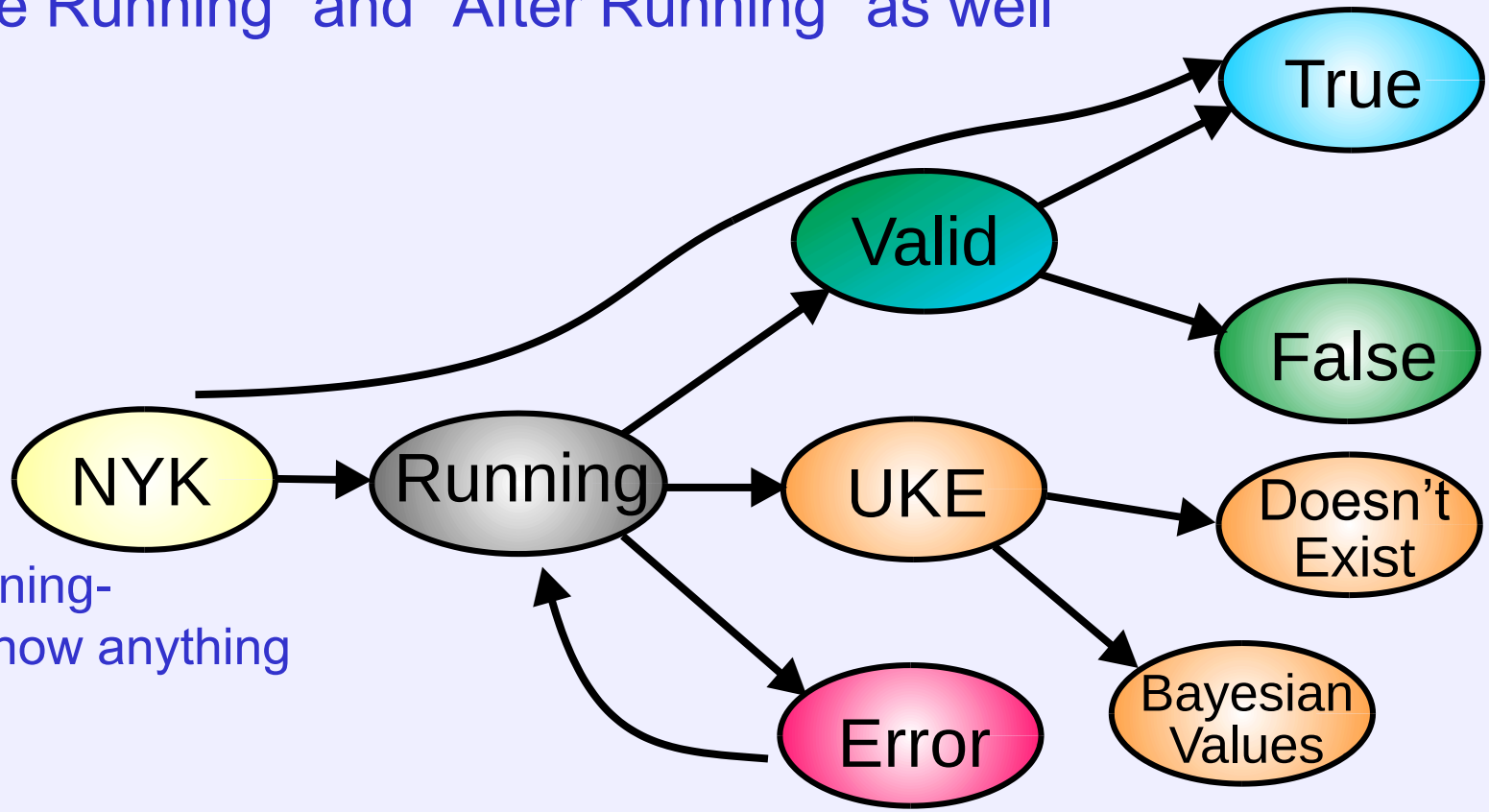
Not True, Not False



If we set P to True, we get an error at the EQV operator.
Similarly, if we set P to False.
Another state is required - ERROR

Are Three States Enough?

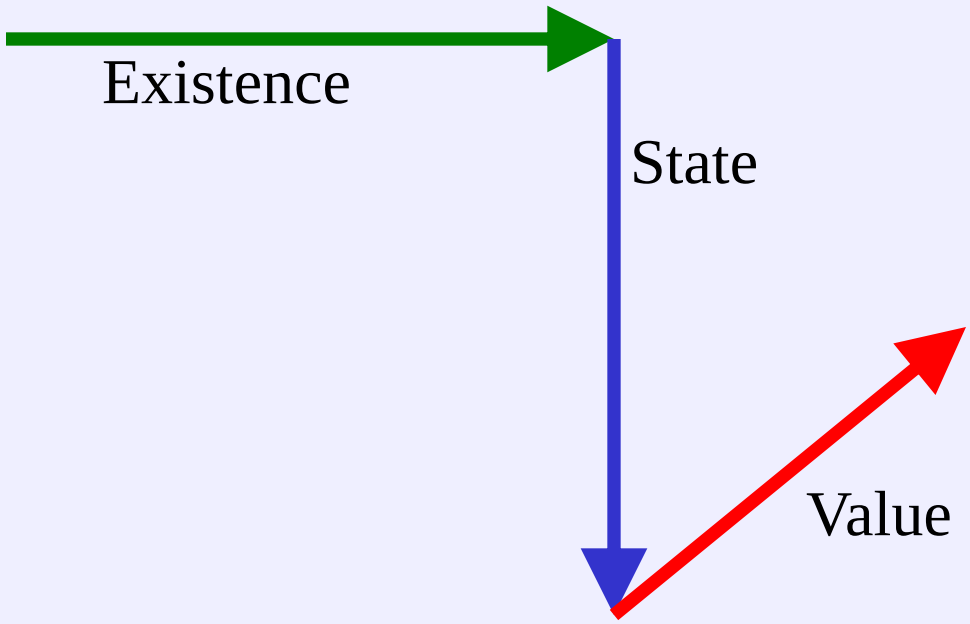
Three states are still not enough - there are the states of “Before Running” and “After Running” as well



Before running-
we don’t know anything

We have to handle the case where the structure is still there, but is dark signifying it doesn’t exist -”the man has no legs”

But There's More...



We need to determine existence before we determine state before we determine value - value on one object may determine existence on another -

if glycosine_level > 0.3 then Antibody is present

Interacting Influences

Consider the tennis player.

- A ball with partially known trajectory and spin
- The position of the opposing player
- A choice of stroke - volley, lob, drop shot
- A collision course with the racquet is plotted, based on the player's position, speed and physical limits

All brought together and processed on the run, with increasing precision until the moment of impact, and then accuracy at the 99.9% level. Putting a single number in a register or using a resistor network (ANN) sound slightly inadequate in comparison.

Abstract State Machines

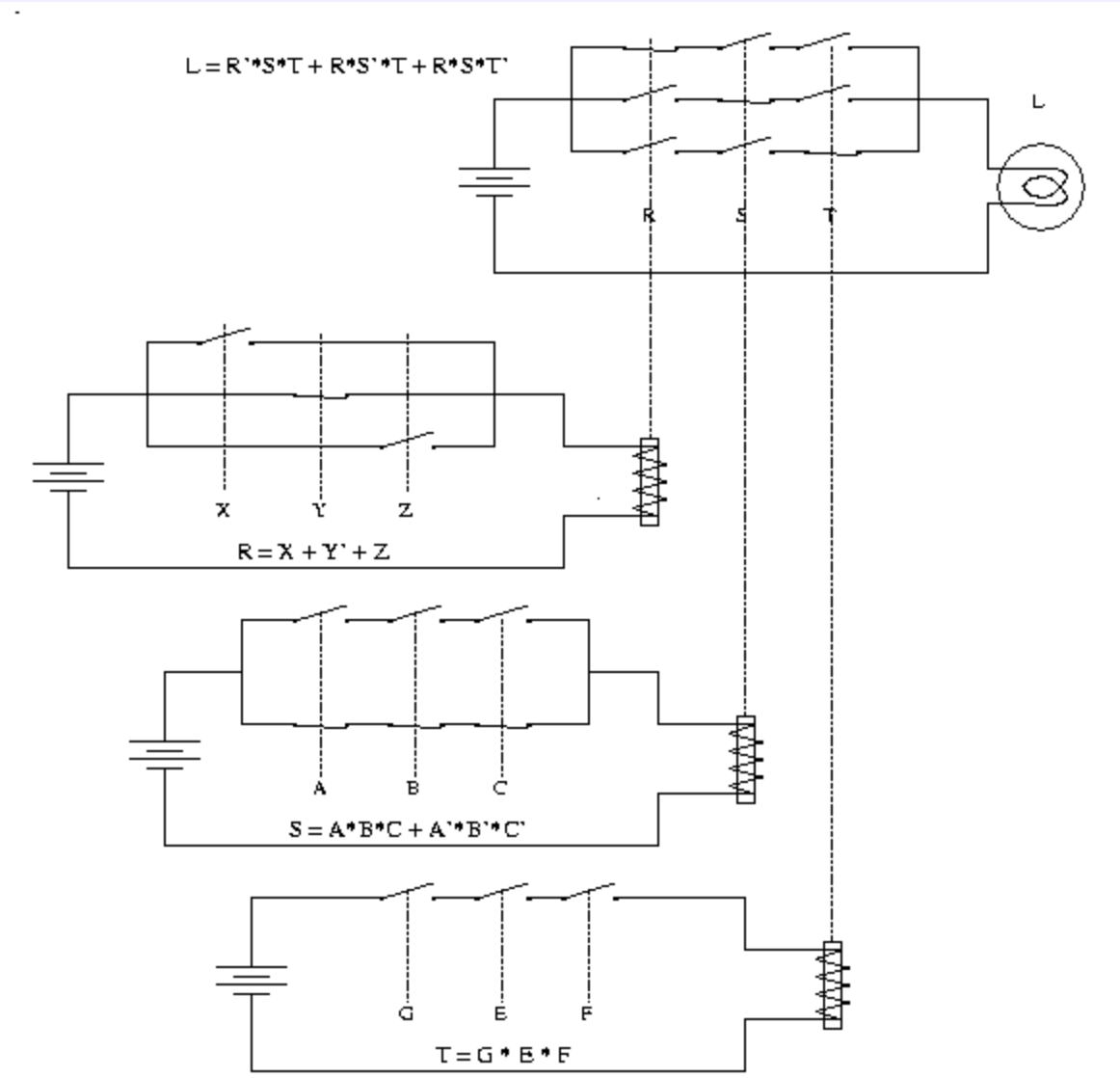
"A formal method for specifying and verifying algorithms"

That's fine, assuming an algorithm is valid in the first place. An algorithm for calculating prime numbers is justified because it will be true forever, a fixed algorithm for interacting with a changing world is not.

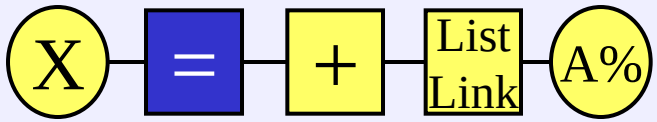
Why not use Realised State Machines?

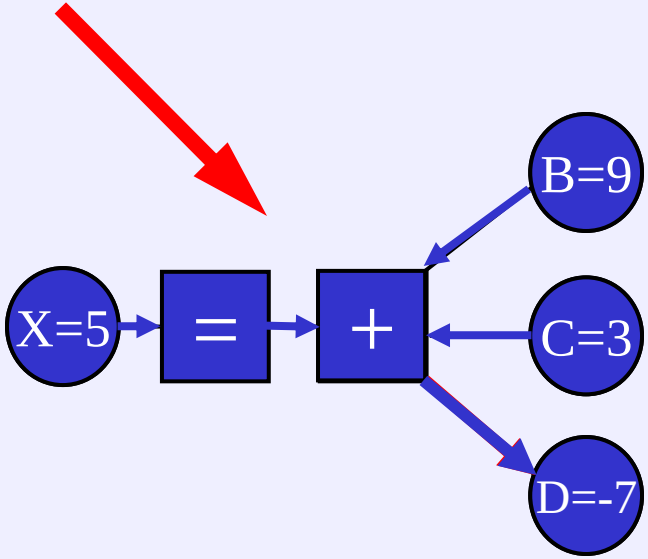
Graph or Structure?

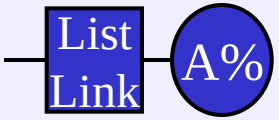
A wiring diagram is a graph - if relays are involved, we understand that other states, not shown on the graph, are possible - the graph is used as a model of a structure



Structure is Invariant





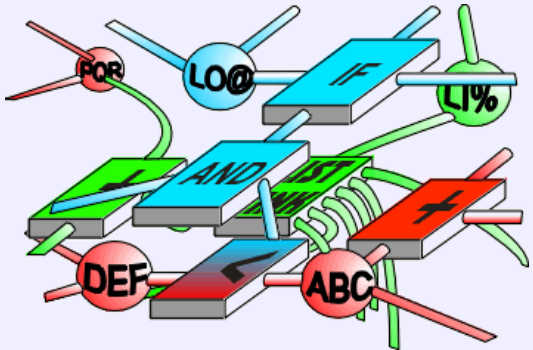


Here is the same structure but two different graphs - the structure has changed state, but it is the same structure and can recover the other state

Simple or Simplistic?

A graph is a simplified model, and a directed acyclic graph is even more simplified, but Ohm's Law doesn't seem like a DAG until you know how you want to use it - it seems more like knowledge.

An active structure can propagate objects through its connections and can respond to propagation by changing its connections. A graph can't.



Stream of Instructions

Simply put, there is no scientific justification for using a fixed stream of instructions to interact with the world.

Plants do this, but if the instructions in their DNA don't match their environment, they die.

Animals don't do this - they change their environment if their instructions don't match.

We shouldn't use programs to interact with a changing world, and as far as our businesses go, it is changing every minute.

Scientific, No?

If we are to represent proofs, we need to maintain the connection between earlier and later statements - we need to ***remember*** states

If we are to model the external world so we can successfully interact with it, we need a modelling system that can change its structure as fast as the external world changes

Active Structures are structures that can do all these things - using connections to remember where states came from, and changes of state to drive activity and change topology

Jacquard Loom

The Jacquard loom of the 1880s - a machine that uses a program to weave a structure

Two Paths

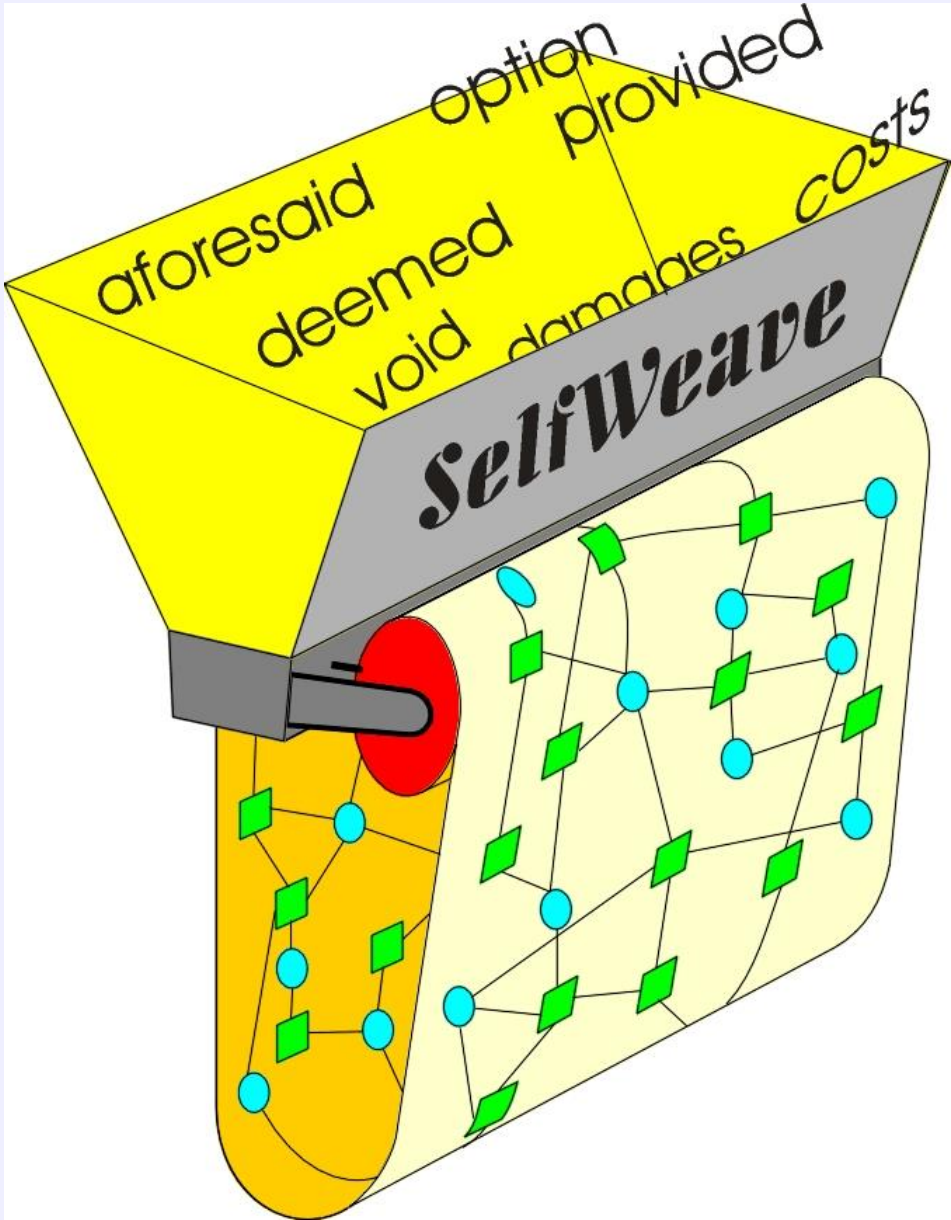
We had a model of a machine that used a program to weave a structure

We could choose to have:

- A machine that used a program to write a program
- A machine that used a structure to weave a structure

Using Structure to Weave Structure

A machine that weaves words into structure can be self-extensible - the structure that weaves the words becomes augmented by the structure it has woven



Battle Lines

Here are the battle lines - a structure which is:

- Active
- Undirected
- Highly Visible and Connectable
- Dynamically Self Extensible
- Self Modifying
- Backtrackable Structurally
- Propagating Complex messages

If we show that a computer-based system can support all these properties, it is up to others to show that any one or all of these can be dispensed with and similar performance attained in a complex dynamic environment.

But Doesn’t Turing’s Proof Show...

Turing’s proof shows that people don’t think much.

It is of the same order as “Have you stopped beating your wife?’ - a single bit of information is not sufficient to resolve the logical problem.

“if it halts it doesn’t halt”

Allow a machine to display two logical states - say a light that glows red or green - and the problem in the Turing proof that revolves around an overloaded single state vanishes. We wanted to show we were cleverer than machines, and failed dismally.

ORION