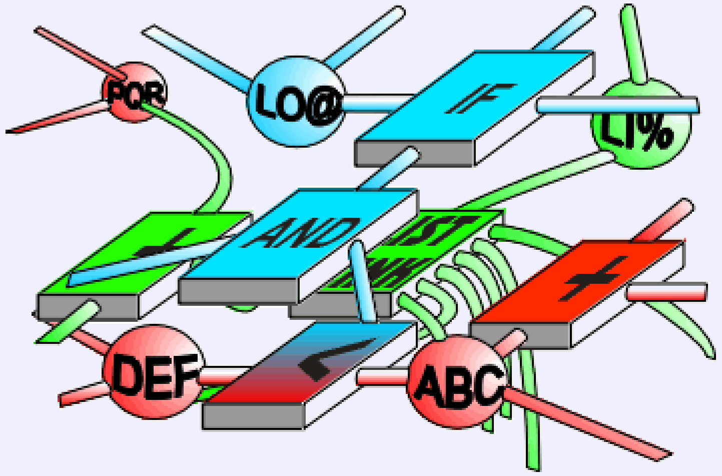
Computational Autonomy



Broadening the Focus

Computational Autonomy is seen as a way of enlarging the narrow focus of a program, which carries out one instruction at a time.

Instead of a program, there are many separate and autonomous agents.

Each agent is assigned one or more relatively atomic roles, and a reward/punishment structure is used to ensure the agent “does what is required”.

With these agents cooperating with each other, it is thought that problems in complex domains will become more tractable.

Assumptions

There are several large assumptions present in the foregoing:

- Autonomy of action is desirable
- A complex interaction is reducible to independent tasks
- The agents communicate with each other, but only with messages at the beginning and end of their tasks
- A stable and static punishment/reward structure can be devised for each agent

Autonomy of Action

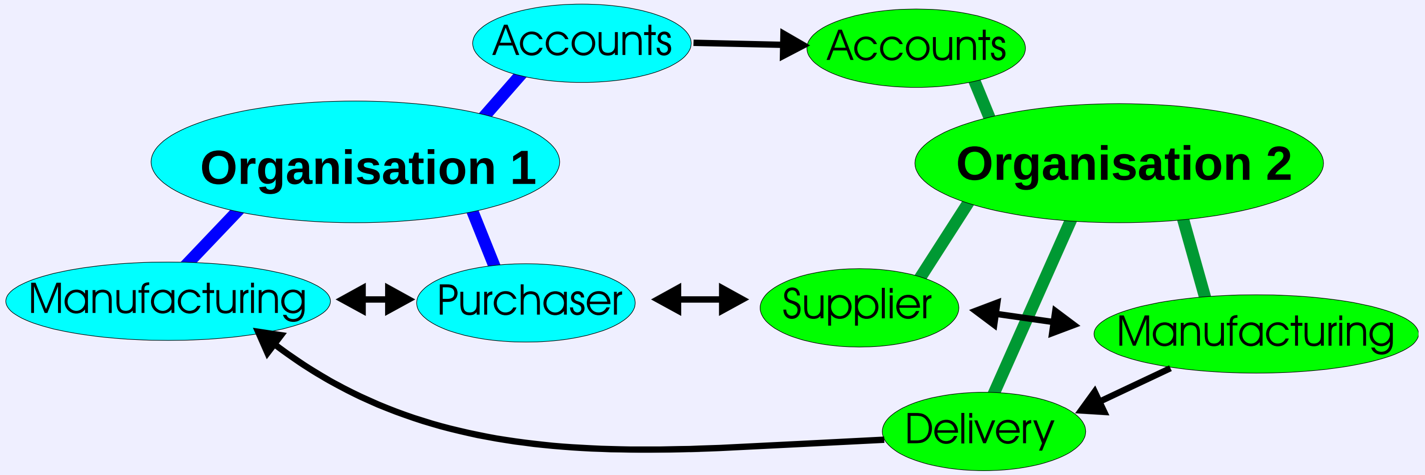
Project Management is all about limiting autonomy of action on the jobsite, because, untrammeled, it results in chaos.

Higher level control is exercised, to ensure a smooth interaction among all the different crafts.

This can’t be exercised at the level of each craft, because not one of them knows what the overall plan is.

A Complex Interaction

eCommerce is often given as an example where autonomy is required.



If we look at the interaction, Manufacturing in Organisation 1 gives Purchaser a requirement. The product may be available, or the Supplier may need to modify what they offer, or Manufacturing may need to modify their specification. Control is shifting back and forth, based on communication of a complex object.

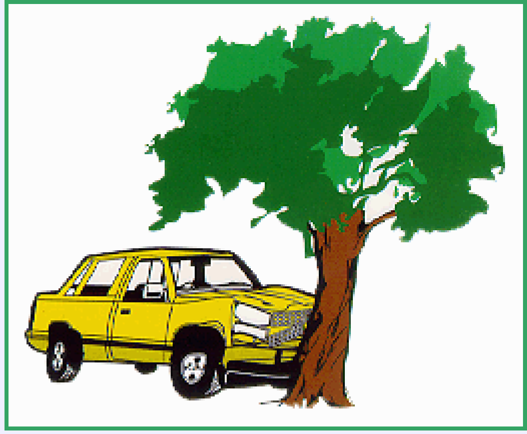
This is Distributed Control, not Autonomy

Punishment/Reward Structure

As any parent will tell you, it is very hard to devise a successful set of punishments and rewards, particularly if the agent can learn how to avoid one and maximise the other.

Simply put, a static structure won’t work. An example is drink-driving.

Drink Driving



People drive while drunk, with disastrous consequences.
Fines are increased, to stop the practice.
The people who can't afford the fines don't pay them, the people who can afford them don't feel constrained by them.
Potential loss of licence is introduced.
People drive while unlicensed.
The only effective deterrent is immediate incarceration.

This is typical of a static punishment structure attempting to control behaviour of autonomous agents.

Alternative Approach

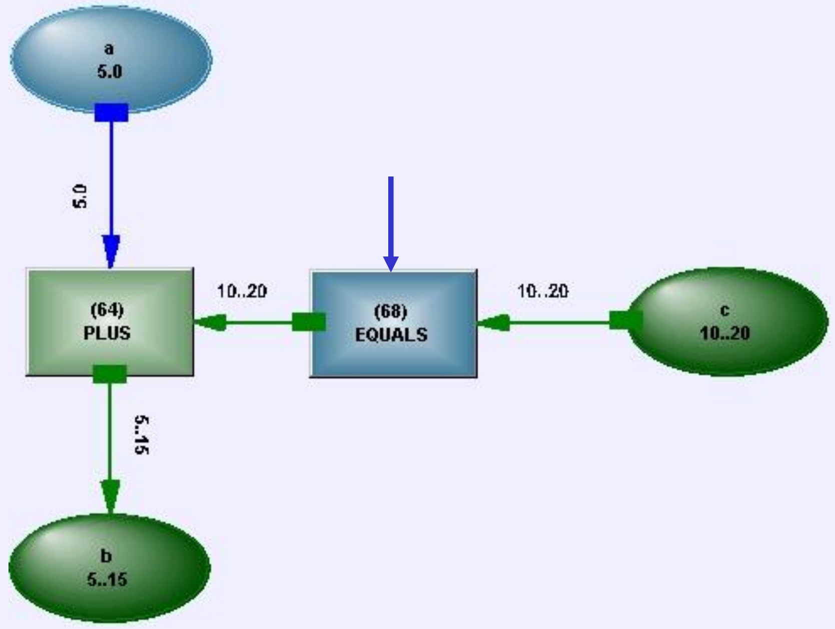
Rather than autonomy as a direct goal, it may be better to pursue distributed control and complex messaging, with computational autonomy emerging as an end-product.

We show a network of quasi-autonomous operators.

A Simple Operator

The PLUS operator determines its response based on the state of its connections - it can propagate a state, a range or a singular value down any connection, including one that functioned as an input.

The operator may appear to be locked in a static framework.

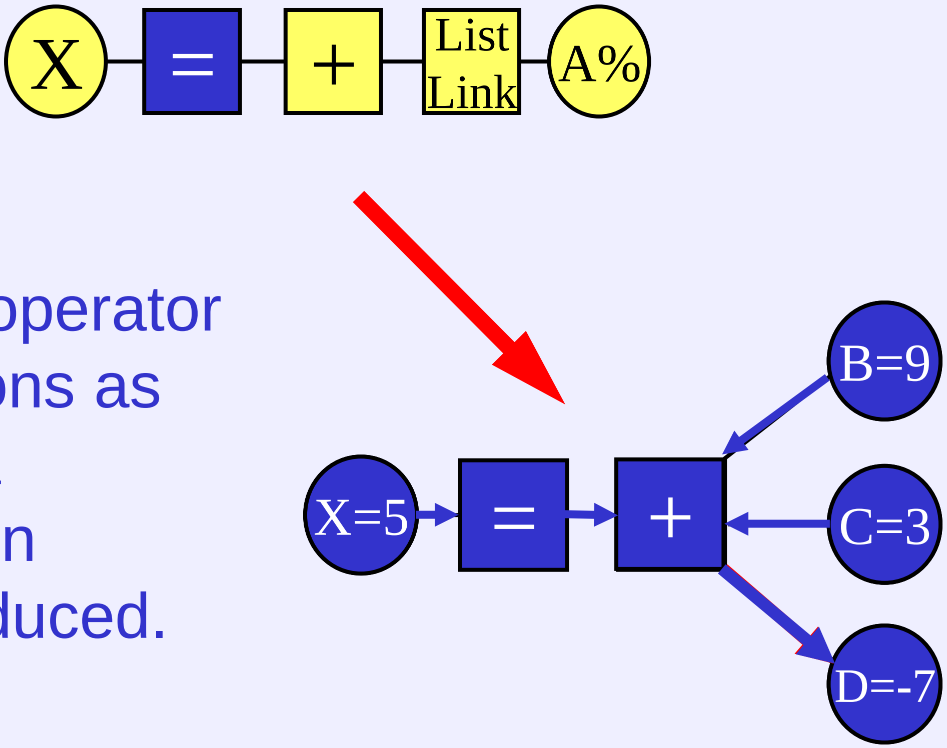


Self Modification

The statement

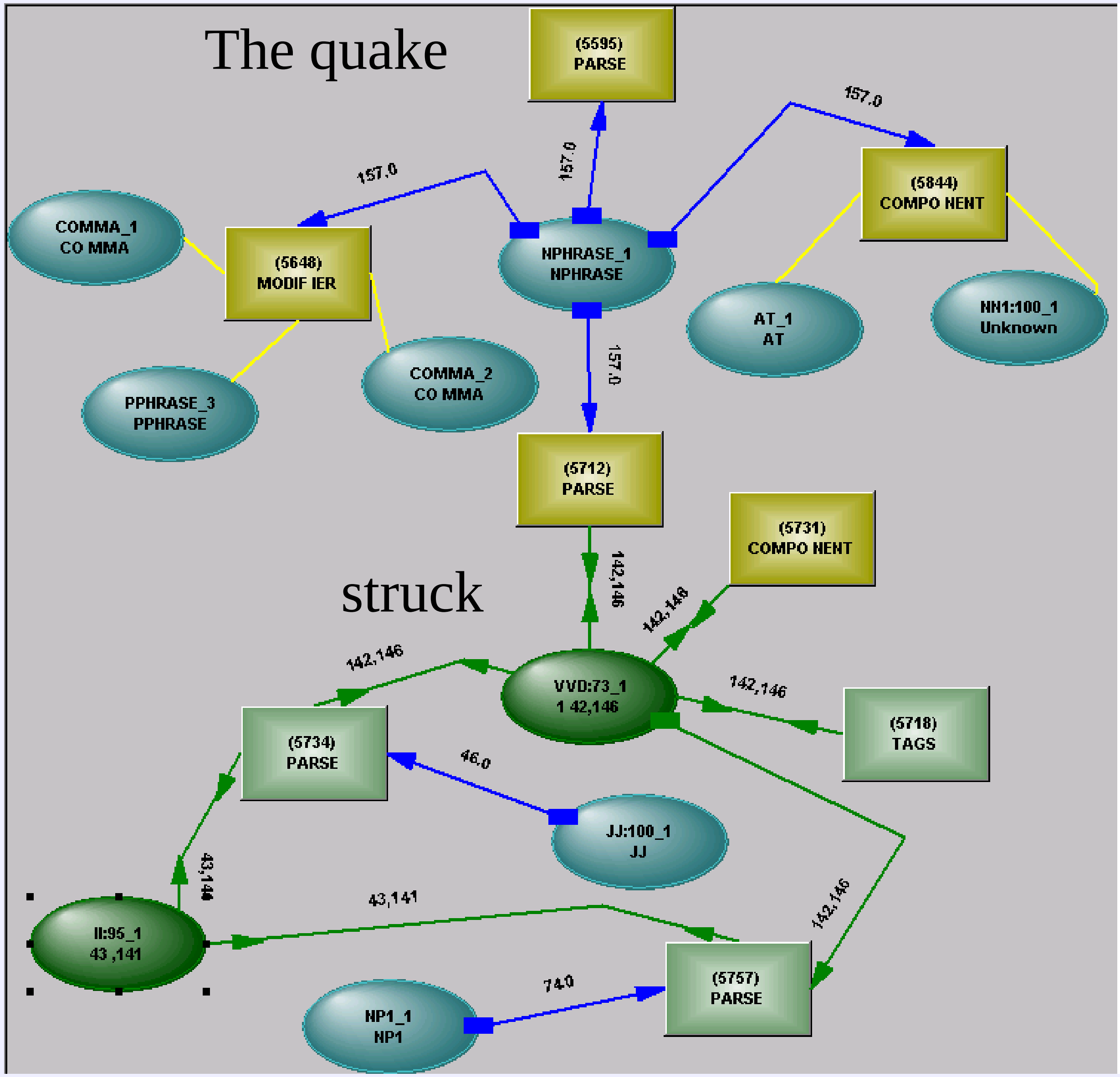
X = SUM(List)

requires that the PLUS operator have as many connections as the list has members - a dynamic self-modification property has been introduced.



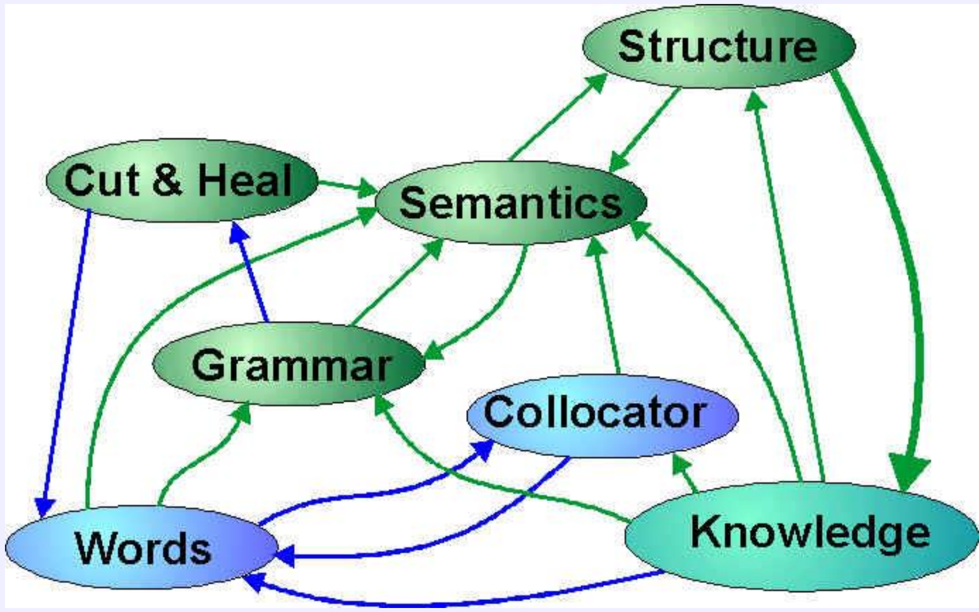
Information Extraction

The PARSE operators here analyse their local environment, and then change the structure, destroying themselves in the process, allowing a more complex structure to emerge. Punishment/reward is irrelevant if altruism is required.



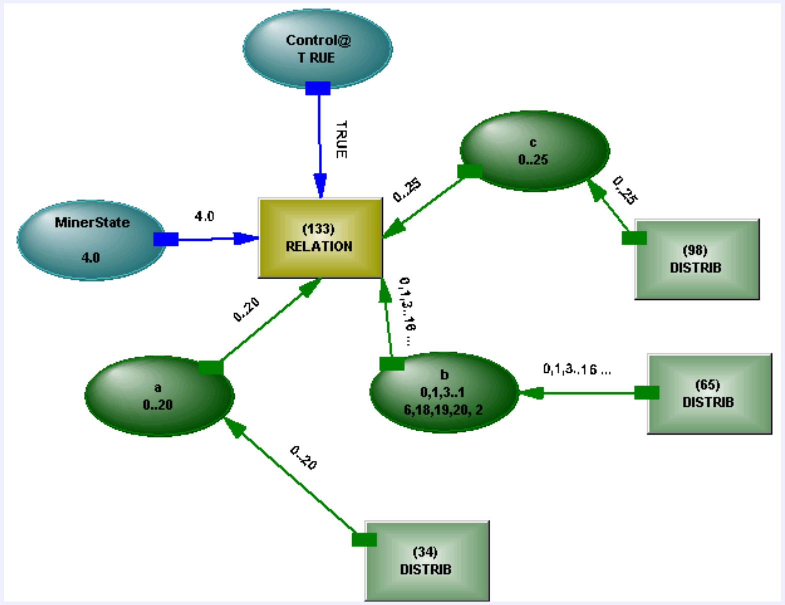
A Complex System

Information Extraction relies on knowledge in the area, and the knowledge is enhanced by the information extracted.
This means that processes interact - grammar helping to resolve POS tags, semantics helping to resolve grammar, and so on. Distributed control and complex messaging are essential for complex applications.



Learning

If we are to justify the label of autonomy for a system, then we should expect it to learn from its interactions. The diagram shows operators connected together that store occurrences in related dimensions, then make those occurrences available as probability distributions, relying on the ability of the network to propagate ranges.



Autonomy

Autonomy is a natural result of building systems with uncommitted structure, distributed control and complex messaging. These same attributes are needed for many applications in operations, engineering, finance, management.

ORION Technology