Alphabetic List of Physical Information Systems

Standard Dictionary for Path Semantics

by Michael Loye, Sven Nilsen, 2025

Table of Contents

Actuator....1
Anomaly Detector....1
A-Sym – Algebraic Symmetry....2
Coh – Coherence....2
Cor – Correspondence....2
C-Sym – Computational Symmetry....2
DeCoh – Decoherence (heat)....3
GIM – General Information Matter correspondence....3
IB – Information Boundary....3
IO – Input/Output....4
ITC – Information Theoretical Construct....4
ITC-IO – Information Theoretical Construct Input/Output....4
PI – Physical Information....5
RS – Real Space....5
RSC – Real Space Construct....5
Sensor....5
SS – Simulated Space....6
SSC – Simulated Space Construct....6

Actuator

An Actuator is an ITC-IO machine where we can might ignore the output. In a theoretical sense, the Actuator ITC-IO machine must be a composition of two ITC-IO machines in a serial connection. The first produces less information out than in, performing work in the environment. The second might be the identity ITC-IO machine, or it could be a source of information, producing more information out than we put in. In the latter case, the Actuator can be thought of as a mixed Actuator-Sensor, which requires action first before receiving new data.

Anomaly Detector

An Anomaly Detector is a Sensor ITC-IO machine. Like all Sensors, the Anomaly Detector uses two ITC-IO machines that measures deviations relative to each other. One ITC-IO machine might be theoretical, a predictive model of the other machine in absence of anomalies. The Anomaly Detector is constructed in a such way that it breaks down in the presence of sinks or sources of information. The input to an Anomaly Detector is scrambled, yet predicted precisely, to scan as many configurations as possible. If the input is repeated, then the search gets constrained to anomalies that react on periodic signals. There is no well defined Anomaly Detector, but various detectors might be constructed to break down when experiments do not match theory.

A-Sym – Algebraic Symmetry

Laws:

A-Sym => C-Sym

A-Sym is a particular kind of C-Sym that is restricted to binary operators or higher operators. This is because unary operators can not have A-Sym. In the case of binary operators, A-Sym is implicitly defined, but for higher operators it needs to be explicitly defined. The following definition holds for binary operators:

If \`f\` **has** A-Sym, then \`f(x, y) == f(y, x)\` for some \`x, y\`.
If \`f\` **is** A-Sym, then \`f(x, y) == f(y, x)\` for all \`x, y\`.

Coh – Coherence

Laws:

C-Sym => Coh | DeCoh

Coh is a quantum property of ITC-IO machines which is only enabled by parallel inputs. This is because the input can not be in an indeterminate state if we know which input gate it goes through. Therefore, there must be at least two input gates to enable an indeterminate state. In a quantum computer, a single input gate might have indeterminate state, so it counts as parallel input. Since we do not know which input gate the indeterminate state goes through, this is a form of C-Sym. Coh might not require usable energy to be preserved, but this is only possible in the absence of DeCoh. In quantum computers we use error correction codes to maintain Coh. This both produces Coh and DeCoh (in the form of heat) and hence error correction codes needs usable energy.

Cor – Correspondence

A Cor is a shared property between two systems, that might be of the same kind or of different kinds. This property is not restricted to ITC-IO machines. For example, in Category Theory, a functor is a correspondence between two categories. With other words, a correspondence might be physical or abstract. A physical correspondence usually does not require physical interaction, because when we speak about physical interactions, we tend to use other terminology. The way Cor happens is usually due to internal configurations in systems, causing a Cor not through direct interaction, but by being mathematically related to each other through composition.

C-Sym – Computational Symmetry

Laws:

A-Sym => C-Sym
C-Sym => Coh | DeCoh

A C-Sym is a property of two functional identical ITC-IO machines, which are given two inputs with two states, \`x, y\`, such that they commute with respect to their outputs. If the input is serial, then the ITC-IO machine will produce DeCoh in the form of heat. This is because the information about the order of \`x, y\` vs \`y, x\` is lost. Hence, since ITC-IO is a particular form of IB, this lost information must be preserved, somehow. The default way of interpreting lost information in any IB is in the form of heat, radiating from the machine. If the input is parallel, Coh might be produced.

DeCoh – Decoherence (heat)

Laws:

C-Sym => Coh | DeCoh

A DeCoh is a particular form of C-Sym. Traditionally, this connection has not been very well understood in computer science. DeCoh happens in all ITC-IO machines with serial inputs that produce less information out than they receive in. This property might be thought of as a C-Sym between two functional identical ITC-IO machines, where `x, y` sent to one is `y, x` for another. Here, there can be an abtrary number of states between `x` and `y`. Also, `x, y` must at least be possibly symbolic distinct in some possible world, otherwise it would be possible to perform a homotopic contraction of the two ITC-IO machines into one ITC-IO machine. This means that symbolic distinction is fundamental for DeCoh (regardless of the idea of using symbolic distinction to explain consciousness in Wolfram models using Avatar Hypergraph Rewriting). When inputs are parallel, DeCoh might still be produced, when the environment IB stores the information of which input gate `x` goes versus the input gate where `y` goes.

GIM – General Information Matter correspondence

Laws:

GIM => ITC

A GIM is produced by performing particular physical ITC experiments. This gives an ontology of that particular physical universe. For example, in the double slit experiment, one can derive the properties of quantum mechanics. There is no well defined GIM prior to performing the actual experiments. Different universes with different physical laws will produce different GIMs. In Path Semantics, we tend to blur the language boundaries between GIM and PI, because the GIM has largely already been established. However, in principle, PI does not need a specific ontology. This means, that one can talk about different GIMs when it is not implied from context that the default GIM is used, that we get from our physical experiments. For example, when somebody are simulating another possible universe with different GIM, or predicting extensions of our GIM.

IB – Information Boundary

Laws:

IO => IB
IB => PI

An IB is any PI system that maintains information. This does not imply that there is IO, nor does it imply that the system is open or closed. Maintaining information can require useful energy, such as error correction codes, or it might not require useful energy, for example in the state of photon. The general idea is whenever information goes in or out, there is some associated physical phenomena.

IO – Input/Output

Laws:

IO => IB

An IO is an IB where one can control information going in or out. This does not imply total control. There might be side effects that are beyond the control of any observer. Also, the sense of control is kind of a joke, because the observer is also an object subject to laws of physics. This undermines the very idea of IO. Despite being a wrong idea, IO makes it easier for people to wrap their heads around various physical systems. This means, some control is required, but there are different philosophical perspectives on this kind of control. There is no way to tell what a “correct” philosophical position would be like. People just define different kinds of control. Sticking to a definition is therefore not a valid claim on ontological truth, but merely a practical language tool. This also produces a kind of paradox, where a GIM produces physical ontology, but is built on IO.

ITC – Information Theoretical Construct

Laws:

GIM => ITC
ITC => ITC-IO

An ITC is a particular physical experiment that simulates an ITC-IO machine. ITCs are much more complex to wrap your head around than ITC-IO machines. It is because an ITC is the analogue of a real computer, while an ITC-IO machine is the analogue of a Turing computer. Now, this analogue is very good, since a real computer can be thought of as an example of as an ITC experiment. Also, a finite Turing machine can be thought of as an ITC-IO machine. From ITC we get GIM which is used to give an ontology to physical laws in our universe.

ITC-IO – Information Theoretical Construct Input/Output

Laws:

ITC => ITC-IO
ITC-IO => IO

An ITC-IO machine is an abstract model of an ITC experiment. It is also an IO system. The IO is dropped when talking about ITC experiments, because it is not philosophically sound to think about what happens in the physical world as observers having arbitrary choice over input and output. This is an illusion of control, which is very useful, but that is only due to the concept of an ITC-IO machine. ITC-IO machines span a wide range of physical phenomena, for example, both classical and quantum computers, computer networks and most physical experiments. An ITC-IO machine is particular useful because it provides semantics for when some physical system produces Coh or DeCoh by C-Sym. C-Sym is a particular property of some inputs with respect to some outputs.

PI – Physical Information

Laws:

IB => PI

PI is the highest abstraction when considering physical systems from the perspective of information. From this perspective, it might not be necessarily true that information is maintained as an IB. However all IBs are PIs.

RS – Real Space

Laws:

RSC => RS
RS => IB

RS refers to physical space or space-time, in the sense of the real world. RS is an IB, or at least thought to be among most physicists. It means that RS preserves information, such that it can not be destroyed nor created without the information going somewhere.

RSC – Real Space Construct

Laws:

RSC => RS

RSC refers to some construct in RS, just like an ITC is an ITC-IO, or an SSC is an SS.

Sensor

A Sensor is an ITC-IO machine where we can might ignore the input. In a theoretical sense, the Sensor ITC-IO machine must be a composition of two ITC-IO machines in a serial connection. The first produces less information out than in, performing work in the environment. The first might be the unit ITC-IO machine, or it could be a sink of information, sending more information in than we get out. In the latter case, the Sensor can be thought of as a mixed Actuator-Sensor, which requires action first before receiving new data. All Sensors use two ITC-IO machines (side by side) that measure deviations relative to each other. One ITC-IO machine might be theoretical, a predictive model of the other machine in absence of sensory input. The Sensor is constructed in a such way that it breaks down in the presence of sources of information. Therefore, a Sensor is kind of grounded in an ideal ITC, but also, it deviates occationally from that description.

SS – Simulated Space

Laws:

SSC => SS
SS => PI

SS refers to simulated space, in the sense of a computer simulation. Information might be created or lost in a computer simulation, so an SS is not necessarily an IB. However, this can also depend on perspective. Since SS needs a real computer, a RSC, there is perspective where an SS is an IB.

SSC – Simulated Space Construct

Laws:

SSC => SS

SSC refers to some construct in SS, just like an ITC is an ITC-IO, or an RSC is an RS.