

Logical Problems in Cognitive Modelling

Catharina Kennedy - Internal Report

Technical University of Dresden, June 2, 1995

1 Introduction and Background

In this paper, the word "cognition" refers specifically to the process of **living** as described in the theory of *autopoiesis* introduced by Maturana [14] und Varela [17]. The theory is a subfield of "Second Order Cybernetics"(SOC) (see, for example, von Foerster [2]). The concepts of SOC have potential applications in the areas of artificial intelligence and robotics as well as in other sciences, for example sociology and medicine. There remains however the problem that they have not yet been successfully formalised due to the requirement for total self-reference.

An attempt will be made to answer the following questions:

1. What are the main concepts of Second Order Cybernetics and how do they differ from those of "classical" cybernetics?
2. Why are these concepts necessary in artificial intelligence and particularly in machine learning?
3. Which problems occur when attempts are made to formalise these concepts with existing logics and why do they occur?
4. How could the problems be overcome? (An overview of polycontextural logic)

1.1 Key Concepts

The following concepts play a particularly important role:

1. Theory of Autopoietic Systems (TAS)
2. Biological Cognition (= Autonomy)
3. System/Environment Boundary
4. Inclusion of the Observer
5. Ontological Location

- 6. Contexture, Polycontextural Logic (PCL)
- 7. Simultaneous Networks

The following distinctions are very important and will be clarified later:

- 1. Autopoiesis / Homeostasis
- 2. Perturbation / Input,Output
- 3. Operationally Closed System / Operationally Open System
- 4. Heterarchy / Hierarchy

2 Concepts of Second Order Cybernetics

2.1 Distinction:Autopoiesis / Homeostasis

Homeostasis is a form of "stability" and occurs very often in biological and engineering control systems. One can say that certain activities "A" regulate objects (or variables) "B". A = **Operator**, B = **Operand**. this can be written as $A \longrightarrow B$. For example: *heatingsystem* \longrightarrow *temperature*.

Homeostatic systems and their variants are sometimes called "self-referential". This however, does not correspond to the term "self-reference" as it is used in second order cybernetics. Examples of such systems are

- 1. **Computational Reflection:** This approach is to be found in Maes [13] and can be represented as follows: $C, B \longrightarrow B$ where $C, B = A$ and C is called the *metaprogram* or metalevel). Clearly, these are configurations in which a system reflects over *parts* of itself
- 2. **Sequential "self-reference":** These are situations where A determines a future state of itself and can be represented as follows:

$$A_t \longrightarrow P \longrightarrow Q \longrightarrow \dots \longrightarrow A_{t+1}$$

or: $A_{t+1} = f(A_t)$; $B = A_{t+1}$. An example is the biological "hypercycle" of Eigen und Schuster.

Autopoiesis means "self-production" ("Auto" = Self, "poiesis" = make) and corresponds to homeostasis with $B = A$. (See Maturana[14], page 48). This can be represented as: $A \longrightarrow A$. Written as a function, this is:

$$A = f(A) = f(f(A)) = f(f(f(\dots)..))$$

The resulting circularity is shown in figure 1. Because of the infinities and circularities which result, it is already clear that there are serious problems when a formal definition of autopoiesis is attempted within the classical framework. This problem will be discussed in detail later.

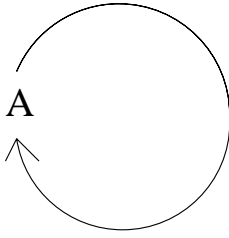


Figure 1: Circularity associated with autopoiesis

2.2 System/Environment Boundary

Autopoiesis requires complete and simultaneous self-reference. The concept of simultaneity which is required here is important but difficult and will be considered in detail later. The most important principle is that the system is not reducible to a sequence or a hierarchy. One can imagine a network of processes where each process interferes with the actions of every other process. All these actions and interferences together form a totality which is qualitatively different from the sum of its components and which constitutes the actions necessary to ensure the survival of the system. As a whole, the situation can be represented as: $A \longleftrightarrow A$. This also excludes $A \longrightarrow A, B$.

The above representation can be interpreted as "A preserves exactly A" while A is continually being "perturbed" by independent processes U . One can say that *selective preservation* is taking place. A **distinguishes** between A and U without any outside assistance. The result is an autonomous generation of a **system/environment boundary**.

To summarize, the following redefinition of existing concepts is required:

A(as a totality) = **system**

U (processes not in A) = **environment**

A distinguishes between A und U = **cognition**

2.3 Distinction: Perturbations / Data

The term *perturbation* was introduced by Maturana to demonstrate that living systems do not process "information" or "data" in the way that artificial intelligence systems do. The environmental events to which the system responds (and the events generated independently by the system itself - sometimes called "compensations") must be treated differently.

Data ("input" and "output") have the following properties:

1. They are always part of the design of the system itself.
2. They have a predetermined interpretation (a "purpose").
3. They do **not** constitute an environment (as required by TAS).

In contrast, the term "perturbation" is an attempt to describe the following:

1. disturbances of the system boundary by an environmental event (perturbations),
2. independently generated disturbances of the environment by the system (compensations),
3. **uninterpreted** events (i.e. contain **no information**),
4. the relevance of an event for an independent living system which is not the observer.

2.4 Distinction: Closed Systems / Open Systems

It is important to distinguish between "open" and "closed" systems in the sense of *information* and "flow of control". The term *operational closure* was introduced by Varela to describe this "closure of information" and is independent of closure in the physical sense. In information-processing terminology, one normally talks about a "system" *and* its inputs (sometimes called its "environment"). In the terminology of TAS, such a "system+environment" is simply a larger system *without* an environment and it is always an open system, even if it is physically "closed".

It is important to emphasise here that "open" and "closed" are two different **points of view** from which a living system may be described. A complex autopoietic network (e.g. immune system, nervous system) can be described as *open* if one is observing particular processes from within the system. If one is describing the *whole* system however (and this must be from "outside" of it), it is more accurate to describe the system as *closed*. The contrast between open and closed is summarised below.

Open Systems: all systems in the natural and information sciences are *open*. Their properties are:

1. *Transparency:* operators and operands are clearly identifiable, with inputs and outputs.
2. *Reducibility:* "reductionism" - piecewise breakdown is possible, resulting in hierarchies and/or sequences.
3. **No environment** exists in the sense of TAS.
4. **Unbroken flow of information** into and out of the information-processing part of the system (classically described as "the system").

An autopoietic (living) system can be described as open if its self-reference is not taken into account. An open system operates on *data*. A closed system is *perturbed*.

Closed Systems are totally self-referential and have the following properties:

1. *Non-transparency:* Operators und Operands are not identifiable.
2. *Non-reducibility* - it is not possible to "cross" the boundary of the system. An infinite regress results and an undefined "self" remains - "holism".

3. **No information** crosses the system boundary - its flow is broken.
4. The system has an **environment** in the form of perturbations.

Volition: Due to the non-transparency of the internal operation of a closed system, it is to be expected that these operations will have non-deterministic "side effects" on the external environment. In effect, the environment is "perturbed" by the system. This form of description (called "volition") corresponds to the observed **autonomy** of living systems. Due to the circularity encountered in the formal definition of closed systems, there is no way of formalising this essential element of autonomy within the classical framework. More will be said about this later.

3 Second Order Cybernetics and Machine Learning

The concepts of SOC have considerable importance in artificial intelligence (AI) and machine learning (ML). To demonstrate this, it is first necessary to identify the relevant properties of existing learning strategies and to show their limitations. These can be listed as follows:

1. All existing learning systems are "open" systems based on information-processing. The programs are specially designed to handle the different types of data, and the data in turn fits in with the program design. The same applies to "fuzzy" or "noisy" data. The system cannot handle events which signify a new situation completely different from anything which has been programmed.
2. The contexts (within which learning takes place) are predefined branches of a universal tree structure. There is no capability to generate a new context when an unexpected event occurs.
3. The system cannot monitor the whole of its current operation.

3.1 Neoconnectionism

The approach known as "neoconnectionism" plays a very important role in current learning systems and it is therefore necessary to investigate its limitations in more detail. The main examples of "neoconnectionism" are neural networks, fuzzy systems and genetic algorithms. The learning strategies can be divided into two main groups:

1. **Supervised learning:**
 - Selected parameters are repeatedly modified (e.g. weights on the connections of a neural network) until the system converges to a "useful" behaviour.
 - The categories of stimuli to which the system should give separate behaviours (or the "fitness" or "correctness" of a particular behaviour) is pre-determined.
2. **Unsupervised Learning** (e.g. Kohonen nets [12]):
 - The "useful behaviour" is a categorization of the stimuli according to the similarities and differences between them. An example is a topological map.

- The "usefulness" of such a map (or other behaviour pattern) is predetermined through selection of particular parameters by the programmer. Examples of such parameters are the frequency of presentation of particular kinds of stimuli or initial values chosen for the connection weights.

One can summarize by saying that such systems do not autonomously determine the usefulness of their own behaviour patterns according to the situation they are in at a given time. A partial solution is achieved by programming for all the "possible" situations along with the "useful" behaviour that the system should converge to in each of these situations. The disadvantage of this approach is that the programmer or designer cannot possibly imagine all possible combinations of events that can happen in the real world.

3.2 Learning in Biological Systems

The main contrasts between biological (autonomous) learning and neoconnectionism can be listed as follows:

1. The **contexts** in which learning takes place are generated by the system itself and not by an outside programmer, i.e. the system determines whether a particular combination of events should constitute a different "situation" (context) or whether it should be regarded as part of the same situation.
2. A different context means that "features" of stimuli should be interpreted differently. A decision is made on **what** this interpretation should be, i.e. information is "constructed" by the system.
3. The system continually evaluates its total current operation in the presence of actual environmental events. i.e. *cognition* takes place.

To summarize, one can say that a second level of learning exists, namely learning to find "useful" contexts in unexpected situations. This is in addition to the development of "useful" behaviour within a particular context. This division into two levels is particularly emphasised in [19] and [22].

Example: a letter-recognition system encounters a 3-dimensional object covering the paper on which the letters are printed. The system must perform the following:

1. The fact that the new event indicates something **different** must first be detected, i.e. it should **not** continue to interpret the features of the 3-D object as candidates for letters or non-letters.
2. A decision is now required: which actions are now useful? There should be a new context, but what should it be? e.g. removal of the object.
3. A new context is established: different learning strategies are now relevant and a different interpretation of the features takes place.

Because cognition is required it is clear that the operational closure of the system must be brought into the design. However, in order to design the system at all, one must have access to its internal operations. It follows that both descriptions of the

system are necessary. A method is required which enables them to be linked in such a way that the identity of the system is preserved, i.e. the operation of the system must be regarded from two different points of view which are both valid.

4 Logical Antinomies

Logical antinomies are paradoxes which occur in self-referential situations. Examples are Russel's Paradox, Gödel's Incompleteness Theorem and the Turing Machine Halting Problem. The Turing Halting Problem will be considered here in detail.

4.1 Turing Machines: Background

A very clear formal definition of a Turing Machine is given in [15]. A Turing Machine is defined by a set of **quintuples**, where each quintuple is of the form:

$$(q_i, \alpha, \beta, q_j, D).$$

The above quintuple describes a step in the execution of a TM and has the form of an "if..then.." rule, i.e. if the current state is q_i and symbol α is read from the tape then write the symbol β in the current position (replacing the symbol just read), enter state q_j and move in direction D (where D indicates left(L), right(R) or no movement(Φ)). If no symbol is to be written, this is also indicated by Φ . q_j can also be the halt-state H. Basically a TM is a program, where the tape acts as unlimited storage capacity.

Of most importance in this discussion is a universal Turing Machine (UTM), which acts like a stored program computer. i.e. it takes on its tape the description (quintuples) of any TM (call it M) in encoded form and simulates the operation of M. The precise details of the coding are not important here, but the literature can be consulted (e.g.[15] and [16] for different coding methods. Not only the quintuples of M are encoded on the tape of the UTM; the tape of M (with its initial symbols) are also encoded on a segment of the tape of the UTM.

The quintuples of a UTM represent the operation of a normal computer, i.e. they describe a program of the form:

```

Initialise current state of M;
Initialise current position of M on its tape;
while state of M is not halt do
begin
  Fetch instruction:
  - search for quintuple corresponding to the current state
    and input symbol for M;
  Execute instruction:
  - record the new state of M;
  - write the new symbol in the current position in the tape of M;
  - update the current position of M according to the direction stated;
end

```

4.2 The Turing Machine Halting Problem

It is well-known that the Halting Problem for Turing Machines (TMs) is unsolvable. A clearly explained proof can be found in [15] and will not be repeated in detail here. Basically the problem involves determining whether an arbitrary machine with an arbitrary sequence of symbols on its tape will halt or not. The only way to do this is for a universal Turing machine (call it A) to execute the encoded description of the "operand" TM, B (along with its initial symbol string) and determine whether this TM will halt when processing the given string. This can be specified as:

$$A(d_B, t)$$

where t is the tape of B. A problem occurs when the "operand" TM, B, is the description of A itself and its input string is also the description of A. i.e., we have the situation:

$$A(d_A, d_A)$$

This means that A must determine whether the process representing its own operation will halt when processing its own description. It must then take a certain action according to its decision. Since the action can involve the opposite of what B does (i.e. the decision can be: if B halts then A goes into a loop and vice versa) and $B = A$ then the result is an absurdity and therefore we must reject the hypothesis that such a machine exists.

4.3 Cognition and the Self-halting Problem

To be cognitive, a universal Turing machine (UTM) would have to examine its own description **as a running program** and distinguish between the actions of this program and other events. Effectively, the UTM must have a label for the process representing its own execution (at the current time) and a different label for events that do not belong to the execution of this program (i.e. symbols that appear independently on its tape).

The requirements for cognition will lead to a similar logical antinomy to that of the halting problem because the process being examined and labelled must itself be amended to include the labelling action, in which case the labelling would no longer apply to the whole process but only a part of it. This means an additional labelling action must be added and so on. It is easy to see that "labelling" can mean making any statement about the process being examined or taking any action which contradicts it.

It is important to note at this point that simply running its own description with just any (other) tape is not going to reproduce the problem. i.e. the situation:

$$A(d_A, d_M)$$

where M is any other machine, does not represent the situation of cognition because the decision that A makes would not refer to its situation *at that instant* (where it operates on its own description); instead it would refer to a different situation where it operates on another description, i.e. the UTM A would refer to its normal (non-cognitive) operation and not to its current (cognitive) operation.

This situation of interest is precisely that of the "self-halting" problem, i.e:

$$A(d_A, d_A)$$

Actually in reality, the description of A could only be written on the tape once and slightly modified so that the "operand" program whose instructions it "fetches" begins on the same tape segment as that of the quintuples of U itself (otherwise, an infinite description would be required since the second occurrence of d_A not only includes the quintuples for the machine but also its tape (its initial string), which must also be d_A and so on..).

It is of interest to note that if such a hypothetical Turing Machine actually operates, it will simply loop forever "fetching the first intruction".

5 Identity Theorem as a cause of Ambiguity

For computer programs in general, the following statements apply regarding antinomies:

- If A is an executing program (i.e. the operator or subject) und B is a program which is being executed by A (i.e. the operand or object) then it is clear that A can contradict or exceed the actions of B .
- If $B = A$ then the relation $A \longleftrightarrow A$ applies. In the case of a Turing Machine $A(d_A, d_A)$ the resulting antinomy can be stated as:

If [A with the tape of A] halts then [A with the tape of A] does **not** halt.

In shortened form:

If [A/A] halts then [A/A] does **not** halt.

- The reason for the antinomy is the identity subject = object. In the situation $A \longleftrightarrow A$, A plays two different "roles", (one as subject and one as object). Due to the classical Identity Theorem, $A = A$, the difference between these two roles is ignored.
- If the above result is stated as follows:

If [A/A] (object) halts then [A/A] (subject) does **not** halt

it is no longer absurd. However, there is no formal system at present which seperates out the roles of subject and object if the formal symbols are identical.

5.1 Antinomies and Cognition: Inclusion of the Observer

One of the key SOC concepts mentioned at the beginning was the **inclusion of the observer** in the world which is being observed. The general representation of cognition was: A distinguishes between A and U .

Clearly, the observer also makes a distinction between the system and its environment. This can be regarded as a sort of "standard" viewpoint or a "starting point" and can be indicated as:

$$A/U$$

For the system A to make this distinction, it must have an internal "image" of A and U . This can be written as:

$$A(A/U)$$

It is clear that any internal image is **not** the same as the "reality" which the outside observer sees. However, with classical formal systems this distinction is not made. An antinomy results because $A(\text{image})$ and $A(\text{reality})$ are regarded as identical simply because the formal symbols are identical. This explains the apparant limitations of formal systems, in that it is impossible for a system to "jump out" of its own boundaries, although everyday experience and empirical observations in biology indicate otherwise.

Alternatively, the cause of the antinomy can be stated as follows: an antinomy results when a system is observed from a viewpoint where it should be described as *closed* but is instead described as *open*, i.e. the **discontinuity** of information between the system and the observer is ignored, and in this way the autonomy of the system is also ignored. In other words it is assumed that the system has no internal image; the only "reality" that exists is that of the observer.

In the present context, the term "image" corresponds to the **process** of operating on a particular object, sometimes called the "reflection" of the object. The term "reflection" is the one normally used by Günther in [4] and [6]). One can therefore regard the "image" of an object as a subject or operator. In the same way, the execution of a computer program is an ongoing active *interpretation* of its data. This is not the same as "output" or "result" of a computation. Instead the term "image" enables a "stepping back" from a particular process and comparing it with other possible interpretations. We can therefore say that the ambiguity between image and reality has the same form as the ambiguity between subject and object.

The classical identity theorem allows only one "reality". In order to formalise cognition, it is necessary to talk about more than one "interpretation" (world view), all of which have the same validity. One of these interpretations can be regarded as "reality" (or the observer starting point) for practical purposes. The following conclusions can now be drawn:

1. Several simultaneously active *ontological locations* are necessary.
2. A new understanding of identity is necessary.

6 Polycontextural Logic (PCL)

The idea of an extension of classical logic to cover many simultaneously active ontological locations was introduced by Gotthard Günther (1900-1984). His most important works are [3], [4] and [6]. The ideas of PCL (which was initially called a "non-Aristotelian logic") originate from Günther's study of Hegel. His aim was to develop a formal theory of dialectics. Further work on the conceptual and theoretical issues is being carried out by Rudolf Kaehr, Eberhard von Goldammer and others. The

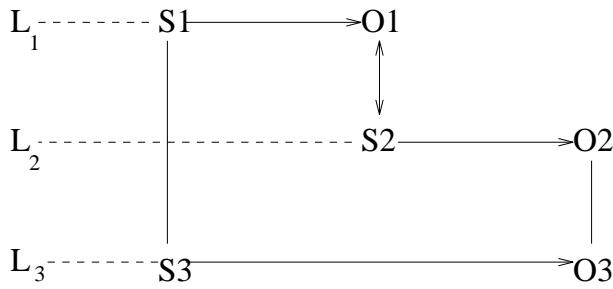


Figure 2: Polycontextural structure

latest articles include [19], [9], [10], [22] and [11]. The content of this section is largely based on these papers.

Although PCL is a "many valued" logic, it does not fall into the category of fuzzy or continuous logics, which allow only values **between** true and false within the one "absolute" system. Instead it is a "many-system" logic, in which the systems (called contextures) are enabled to interfere with each other, resulting in a totality which is qualitatively different from the sum of its individual components. In this summary of PCL, the following issues will be addressed:

1. Steps towards a formal representation of self-reference which avoids antinomies,
2. The concept of polycontexturality as a foundation for a formal model of operational closure,
3. The role of multiple logical values,
4. Current unsolved problems.

6.1 Removal of Antinomies

In the previous section, the reason for the antinomy was identified as the ambiguity between subject and object in the special case where the formal symbols are identical. This ambiguity can be shown as follows:

$$S/O \longleftrightarrow S/O$$

It is clear that there are two possible interpretations here, according to the different "roles" that are played: i.e. either

$$S \longrightarrow O$$

or

$$O \longleftarrow S$$

For total self-reference to apply, both must be valid simultaneously. The polycontextural logic requires these "interpretations" to be specifically labelled. The resulting structure is shown in figure 2.

For figure 2, the following terminology applies:

- L_1, \dots, L_3 are **contextures**.
- Important relations:
 1. \rightarrow **Order relation**,
 2. \updownarrow **Exchange relation**,
 3. $|$ **Coincidence relation**.
- L_1 and L_2 are **Elementary Contextures**, i.e. they represent a *direct* hierarchical relation between subject and object.
- L_3 is a **Compound Contexture**, i.e. an indirect relation between subject and object.

A contexture can represent an observer position (or point of view) from which descriptions may be made. Effectively, the idea of ontological location is represented.

Each of the vertically connected pairs $(S_1, S_3), (O_1, S_2), (O_2, O_3)$ are **identities** with two simultaneous forms of expression. This means that for example S_1 cannot mean something **different** from S_3 . This is what the vertical line between them indicates. These identities may have various interpretations. Günther originally called them "identities of reflection" as follows: (O_2, O_3) - "Irreflexivity", i.e. the objective world as it is, (O_1, S_2) - "Reflected objective world", (S_1, S_3) - "double reflexion" - reflection on the process of reflection. In computer science, the word "reflection" may be replaced with "operator", and the object of reflection with "operand". This enables an operator in one contexture to act simultaneously as an operand in another contexture.

The above diagram is a minimum configuration for a self-referential system (i.e. 3 contextures). Most realistic systems will require considerably more complex structures than the one given here but they will not be considered in this study.

6.2 Connection between Open and Closed Systems

An elementary contexture represents **either**:

1. an open system (i.e. the contexture in which an observer is currently positioned)
or
2. a closed system (i.e. a different contexture from the one the observer is in).

It is to be noted that the definition of an open system requires only one contexture (observer(S), reality(O)). In contrast, the definition of a closed system requires more than one contexture. This is because, in effect, a contexture **defines** a system which "seperates itself" from its environment (i.e. cognition) and to **describe** a contexture requires more than one contexture. In the natural and information sciences we operate **within** the one contexture; we do not try to describe one. If we attempt to do this then a circularity results which can take either of the forms shown in figure 3. Both are different representations of the same problem. In the first case, one is attempting to define "reflection of reflection" ($S \rightarrow S$) and in the second, an attempt is made to represent an observed object with autonomous properties ($O \rightarrow O$).

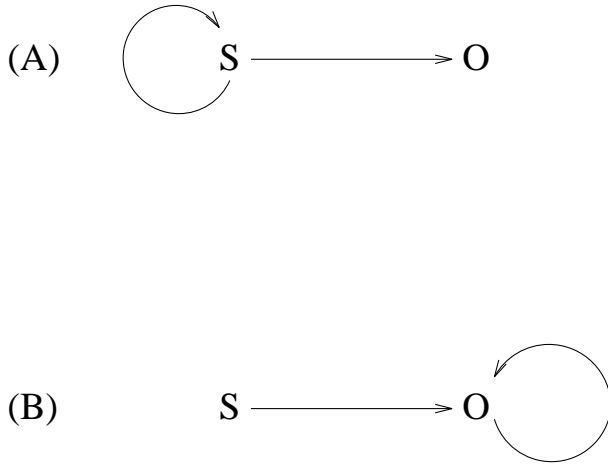


Figure 3: Two circularities

If we consider the first representation, L_2 may be interpreted as the classical universe, with O_2 as the "objective reality" of the observer. This is generally the interpretation of Günther in [6]. The following descriptions are then valid: L_2 is "open" and L_1 is "closed" relative to L_2 .

This means that L_3 (compound contexture) can be interpreted as representing the following:

1. the autonomy of the system and
2. the form of expression of a closed system in an open system.

Because the **causes** of the actions observed in L_3 are not identifiable, the actions appear non-deterministic. In this way the *volition* of the system is explicitly represented. It was not possible to do this within a classical framework, because the simultaneous forms of expression could not be represented.

6.3 The Role of Multiple Logical Values

To explain the reason for introducing multiple values, it is first necessary to define the term **contexture** more precisely. The following properties apply:

1. A contexture is a domain in which exactly two logical values exist (i.e. a two-valued logic). From **within** a contexture, these values play the roles of "true" and "false". In contrast, when one is talking **about** a contexture, a degree of "relativisation" or abstraction is required and it is common to talk about the "positive" (designatory) value and the "negative" (non-designatory) value for the contexture. These are sometimes written as P and N.
2. A contexture is a domain in which values **between** "true" and "false" can exist. This means that fuzzy logic as well as other similar many-valued logics are monocontextural. Because a third value "outside" the system cannot exist, one can say that *tertium non datur* (TND) applies in a broader sense
3. A contexture is a domain in which a subject **sets itself apart** from its objects (i.e. a process of cognition is defined).

A logical domain (or contexture) can also be regarded as a "universe".

6.4 Subject as non-designation

One of the key concepts of PCL is the idea of subject and object as logical values. Since a contexture defines a "setting itself apart" of a subject from its objective environment (in other words it is stating that it is **not** its environment), it follows that everything that is not designated in that universe can only refer to the subject itself (because of TND).

In the terminology of computer science, one can imagine that a (designated) object for a computer program is one of its internal data structures. When one states that a particular object does **not** have a certain property, one is also stating that an object which **does** have the property is not designated. i.e. it could be **any** of the possibilities in that universe. To identify this "any" positively, one must ask what determines the whole universe for the program. The answer is the operators (code) of the program itself. For example, if the program classifies colours, then its whole universe is the set of all colours. To identify this universe positively is to say something positively about the program or procedure (i.e. the subject).

When one is acting **inside** such a universe then it has no boundary and the "any" or "everything" is infinite. For example, one can be in a universe where only particular colours exist and nothing else. The observer inside the system cannot label his universe as "colours" (this would be "designation"). For him it is "everything" ("non-designation").

Although this principle can also apply to hierarchical systems, it plays a particularly important role as a link between open and closed descriptions of a self-referential system. When the objects and internal operations of a system are clearly visible, the system itself has no boundary and we are talking about an open system in a particular logical domain, say L_2 . When we "jump out" of such a system, and give it a *name*, a boundary for it is generated and it now becomes a single object in a different universe or logical domain, say L_1 . The term "jumping out" was first used by Hofstadter in [7] to describe such a process. Effectively it characterizes an inductive step.

It is not true to say that the new bounded entity is an object in the "normal" (or classical) sense since it acts as an object and a subject simultaneously. For such entities, Günther introduced the term *objective subject* (S^o) in contrast with the subjective subject S^s . Other terms used are: "internal observer" or *Du* ("thou") in contrast with the classical identities of external observer (*Ich*) and object (*Es*). S^o is just another way of describing a system as operationally closed.

In the above inductive step the (external) observer can be said to have "changed position" from L_2 to L_1 and what was previously an open system is now closed and vice versa. Alternatively, one can say that the subjective subject (S^s) has become the objective subject (S^o) and what was S^o has now become S^s . This is like the "I" and the "thou" changing places.

This (inductive) process of designating the "everything" in one universe as a positive object in another universe is a important operation in PCL and will be considered in the next section.

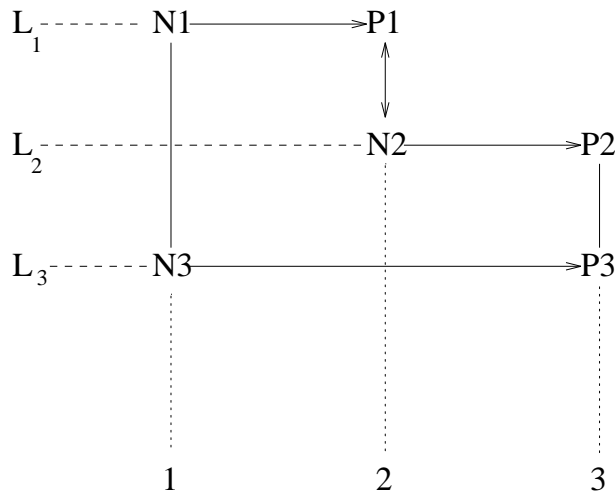


Figure 4: Polycontextural structure with inserted global values

6.5 Second Negation Operator

The different "identites" which were referred to in section 6.1 are indicated in PCL with "global" logical values. There are many ways of denoting these values but the simplest is to use natural numbers 1, 2, 3, .. The highest number indicates the highest "degree of reflection" and the lowest denotes objectivity. The addition of "global" values to the polycontextural representation is shown in figure 4. The details of how a new contexture is generated when self-reference occurs can be described as follows (Let L_2 be the initial contexture):

1. The non-designatory value (2) must become the positive value (because the subject must become the object). Since it is a positive value, it must have a negation. However if the classical negation operator is used then it simply becomes 1 which is the positive value in L_2 . No new contexture is generated and we are left with the ambiguity between subject and object which caused the antinomy.
2. It follows that an additional negation operator is required, which produces the value 3 when 2 is negated. In this way the requirement of transforming 2 from the negative value in one contexture to the positive value in another contexture has been met.

6.6 Global Logical Values and Heterarchy

It has been shown by Günther in [5] that the global logical values 1, 2, 3, .. can represent a *heterarchy*. The term heterarchy was introduced originally by McCulloch in [1]. Basically it means that the law of transitivity does not apply. The logical operators conjunction and disjunction can be generalised in a multi-valued system to indicate **preference**. This means that conjunction prefers the "most negative" (the highest) value if two different values are "offered". With disjunction the lowest value is preferred. If $D =$ disjunction and $K =$ conjunction and $pXYZq$ means: apply X in L_1 , Y in L_2 and Z in L_3 as a single parallel logical operation then in a multi-valued logical table it can be shown that the combinations $pDDKq$ and $pKKDq$ produce a

heterarchical order of preferences. (The details of multi-valued tables are to be found in the above article and will not be reproduced here). In the first combination $pDDKq$, the following order of preference is obtained: $1 \gg 2$, $2 \gg 3$, $3 \gg 1$, where \gg indicates "preferred over". Similarly, $pKKDq$ gives the order: $2 \gg 1$, $3 \gg 2$, $1 \gg 3$. This ordering of preferences reflects the heterarchy of the different identities.

In practice, "preference" can be regarded as *taking precedence over* or determining something. This can be shown more clearly if instead of using numbers 1, 2, 3 we use the letters originally used by Günther: I, R, D where $I = 1 =$ irreflexive object, $R = 2 =$ reflection of object and $D = 3 =$ double reflection (reflection of reflection). It is then certainly true that: I determines R (in the sense that the actual content determines what is reflected), R determines D (for the same reason) but D determines I (in the sense that the volitive operation of the system actively changes the outside world). This all happens simultaneously. It follows that the existence of the above parallel operation is a possible indicator for a formal model of the distributed circularity of a system and its environment.

6.7 Summary of Important Concepts in PCL

The PCL concepts introduced here can be summarized as follows:

1. Removal of ambiguity: Explicit representation of subject/object relations when formal symbols are identical.
2. New concept of identity and *parallelism*: one identity has several different simultaneous forms of expression, thereby enabling *volition* to be formally represented.
3. Induction: "stepping out" of a system.
4. Identities: Subject, object and objective subject as *global logical values*.
5. *Heterarchy* or distributed circularity of global values.

6.8 Problems that still remain

It is envisaged that PCL will someday be used in the technical realisation of intelligent control systems. The following problems remain unsolved at present:

1. It is not yet clear how the dynamics of the system will operate. **Simultaneous networks** and perturbations should be implemented with transclassical logical operators (transjunctions).
2. A transclassical machine theory (Polylogical machines) is required.
3. The required concept of "simultaneity" presents conceptual difficulties. Each process (as a form of expression) must have its own time. There is no global synchronization.

Language and conceptual problems:

1. How does one specify a PCL system? Positions of description (ontological locations) exist which are not contextures. Somehow the simultaneous networking and interaction between contextures must be specified. This cannot be done in statements of a conventional language because then one must operate **within** a particular contexture (where one interpretation is valid). Günther has introduced the idea of abstract patterns of distinction (*kenogrammatics*) in an attempt to describe *structure* or form without requiring semantics. Practical application is not yet possible.
2. How does one overcome the conceptual problems? In order to make more theoretical progress, practical familiarisation with the new concepts is essential. A possible solution is the development of interactive software tools.

References

- [1] W. S. McCulloch. *A Heterarchy of values Determined by the Topology of Nervous Nets*, Bull.math.biophys.7, 1945.
- [2] Heinz von Foerster. *Sicht und Einsicht*, Vieweg, 1984.
- [3] Gotthard Günther. *Grundzüge einer neuen Theorie des Denkens in Hegels Logik*, Felix-Meiner Verlag, Hamburg, 1933, zweite erweiterte Auflage 1978.
- [4] Gotthard Günther. *Idee und Grundriß einer nicht-Aristotelischen Logik*, Felix-Meiner Verlag, Hamburg, 1959, zweite erweiterte Auflage 1978, dritte Auflage, 1991.
- [5] Gotthard Günther. "Cognition and Volition" in *Beiträge zur Grundlegung einer operationsfähigen Dialektik*, Band II, Felix-Meiner Verlag, Hamburg, 1979.
- [6] Gotthard Günther. *Beiträge zur Grundlegung einer operationsfähigen Dialektik*, Bands I-III, Felix-Meiner Verlag, Hamburg, 1976, 1979, 1980.
- [7] Douglas Hofstadter. *Gödel, Escher, Bach: an Eternal Golden Braid*, Basic Books, 1979.
- [8] R. Kaehr, E. von Goldammer. *Poly-contextural modelling of heterarchies in brain functions* in: *Models of Brain Function* (R.M.J.Cotterill, ed.), Cambridge University Press, 1988.
- [9] R. Kaehr. "Zur Logik der 'Second Order Cybernetics'" in *Kybernetik und Systemtheorie - Wissenschaftsgebiete der Zukunft?* (ICS, ed.), IKS-Berichte, Heft 1, Dresden, 1992.
- [10] R. Kaehr. "KOMPASS - Expositionen und Programmatische Hinweise zur weiteren Lektüre der Schriften Gotthard Günthers" in *Gotthard Günther - Technik, Logik, Technologie* (Ernst Kotzmann, ed.), Profil Verlag München, 1994.

- [11] Klagenfurt, Kurt (Arbeitsgruppe) *Techinsche Zivilisation und transklassische Logik*, Suhrkamp, 1995.
- [12] T. Kohonen. *Self-Organization and Associative Memory* Springer Verlag, 1984.
- [13] Maes, Pattie and Nardi, Daniele (eds.) *Meta-Level Architectures and Reflection* North-Holland. 1988.
- [14] H. R. Maturana, F.J. Varela. *Autopoiesis and Cognition*, D.Reidel Publishing Company, 1980.
- [15] A. Narayanan. *On being a Machine*, Vol. I, Formal Issues in Artificial Intelligence. Ellis-Horwood, 1988.
- [16] B.A. Trachtenbrot. *Algorithmen und Rechenautomaten*. VEB Deutsche Verlag der Wissenschaften Berlin 1977.
- [17] F. J. Varela. *Principles of Biological Autonomy*, North-Holland, 1979.
- [18] F. J. Varela and P. Bourgine (eds.) *Towards a Practice of Autonomous Systems*, Proceedings of the First European Conference on Artificial Life. MIT Press, 1991.
- [19] E. von Goldammer, R. Kaehr. "Problems of Autonomy and Discontextuality in the Theory of Living Systems", in *Analyse dynamischer Systeme in Medizin, Biologie und Ökologie* (D.P.F.Möller and P.Richter, eds), Springer Verlag, 1990.
- [20] E. von Goldammer, H. Spranger. "Kybernetik und Systemtheorie aus der Sicht der Medizin" in *Kybernetik und Systemtheorie - Wissenschaftsgebiete der Zukunft?* (ICS, ed.), IKS-Berichte, Heft 1, Dresden, 1992.
- [21] E. von Goldammer. *Self-Organizing Systems in Dynamically Changing Environments*, Contribution to the "Real World Computing" by Axis Computing Ltd.
- [22] E. von Goldammer. "Polycontextuality" in *Gotthard Günther - Technik, Logik, Technologie* (Ernst Kotzmann, ed.), Profil Verlag München, 1994.