Chapter 3

Formalised theories

3.1 Some intuitions

Autopoietic theory was most prominently proposed by Maturana and Varela (Maturana & Varela, 1980) and also many modern works often refer back to that particular formulation. One of the hallmarks of the theory is the definition of an autopoetic system (or autopoietic machine). Maturana and Varela's language has been described as idiosyncratic (Beer, 2015), hence some terms in that definition require further analysis. The definition of an autopoietic system integrates and implies a couple of other concepts, which shall be introduced here in brief and which will be defined further in the next section.

An observer is able to distinguish *unities*. A unity is either an "unanalysable wholes" or a composite, i.e. a *structure* made of other unities. Unities can have *relations* between them. When a certin set of relations are the case for a unity, that unity has a certain *organisation*. Unities can be transformed into other unities (or destroyed) over time by *processes*. Processes can be "concatenated" into a *network*.

If a network of processes produces unities (or components) that maintain the same organisation (i.e. same set of relations) over time, that network of processes is called an *autopoietic system* (or autopoietic machine). In the original definition (Maturana & Varela, 1980), autopoiesis is described as:

> *An autopoietic machine is a machine organized (defined as a unity) as a network of processes of production (transformation and destruction) of components that produces the components which:*
>
> 1. *through their interactions and transformations continuously regenerate and realize the network of processes (relations) that produced them; and*
> 2. *constitute it (the machine) as a concrete unity in the space in which they (the components) exist by specifying the topological domain of its realization as such a network.*

Similarly Piccinini defines his mechanistic account of computation based on more abstract terms (Piccinini & Scarantino, 2011), which however will be shown to blend into the formalisation of autopoietic theory.

> *We use "generic computation" to designate the processing of vehicles according to rules that are sensitive to certain vehicle properties and, specifically, to differences between different portions of the vehicles.*

The general goal here is to keep the formalisations as true to the original source as possible, i.e. to minimize the addition of further assumption, to facilitate best use of the formal theory in the future.

3.2 Autopoietic theory

1 Unities and properties

In various works Maturana argued that "everything said is said by an observer" (Maturana, 1987). In our first-person experience we observe things we do not understand as being part of ourselves (as observers), hence the very basic distinction between observer-things and non-observer-things is being made. Non-observer-things can be distinguished further, but naturally their nature depends on the observer. Hence a formalisation of autopoietic theory should start (similar to what the authors did (Varela, Maturana, & Uribe, 1974)) at and with the observer.

A given *observer* is embedded in a spatially extended environment S and persists over time T. The set of all possible 2-tuples (S', t) of a subspace S' and a time t shall be called the set of all possible spacetimes X and can be defined as follows, with $\mathcal{P}(S)$ being the power set[1] of S:

$$X := \mathcal{P}(S) \times T \tag{3.1}$$

The observer's most notable capacity is to distinguish something as different from itself. In a "conceptual operation of distinction" a *unity* is "separated from a background" (Varela et al., 1974, p.15). A unity can be composite(details in the next section) or simple. A simple unity is understood as "unanalyzable whole" and "endowed with constitutive properties" by the observer.[2]

Here the notion of a *property* is interpreted such that a unity can only be distinguished by means of endowing it with a property. That is, without a defining property, a unity cannot be separated from a background.[3]

While properties are mentioned in the literature, the notion of property values shall be introduced here. Property values were not found to be mentioned in the literature, but shall be used to demonstrate the flexibility of the presented approach without loss of generality.

Let A the set of all properties the observer can distinguish and $a \in A$ some property, then the value domain for a can be defined as $V_a = \{v_{a,1}, v_{a,2}, \ldots\}$ with $|V_a| \geqslant 2$. The operation of *distinction* by a observer can now be captured as a function given a property, that maps a spacetime (subspace at a given time) to a property value for a given property $a \in A$.

$$q_a : X \mapsto V_a \tag{3.2}$$

Independent of the specific instance of "background" or environment of the observer, a unity exists (like an observer) somewhere in space and at a point in time. Based on the definition above we can define the set of all possible unities $\bar{U}_{t,a,v}$ at time $t \in T$ for a given value $v \in V_a$ of a given property $a \in A$.[4]

[1] The power set $\mathcal{P}(A)$ of a set A is the set of all possible subsets of A. For example, $\mathcal{P}(\{1,2,3\}) = \{\{\}, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$.

[2] "A unity may be treated as an unanalyzable whole endowed with constitutive properties, or as a composite entity with properties as a unity that are specified by its organization and not by the properties of its components" (Varela et al., 1974, p.15)

[3] While a property may emerge ipso facto, but it always needs to be recognised.

[4] Here the symbol $\cdot$ is used to denote the presence of a value, which does not get bound by a variable, as the value is not used elsewhere.

$$\bar{U}_{t,a,v} := \{x \mid x \in X \ \wedge \ x = (\cdot, t) \ \wedge \ q_a(x) = v\} \tag{3.3}$$

Utilising the definition above, the observer's set of all known unities (for a time t and in total) can then be formulated as:

$$\bar{U}_t := \bigcup_{a \in A} \bigcup_{v \in V_a} \bar{U}_{t,a,v} \tag{3.4}$$

$$\bar{U} := \bigcup_{t \in T} \bar{U}_t \tag{3.5}$$

The set of all theoretically possible unities is a superset of the observer's actual set of recognised unities $U_{t,a,v} \subseteq \bar{U}_{t,a,v}$ for a given time, property, and property value. Similarly, $U_t \subseteq \bar{U}_t$ and $U \subseteq \bar{U}$. Note that the indices t here refer to the moment in time of the distinguished unities exist in, which is generally different from the moment in time the observer exists in. The observer-specific set $U_{t,a,v}$ and by extension U_t and U may change over time.

2 Composition, structure, and organisation

As mentioned above, a unity can be simple or composite. *Composition* here is understood in that a unity can contain other unities, which are often referred to as components. Containment is understood such that for two unities $u_1, u_2 \in U_t$ (for some t and $u_1 \neq u_2$) $u_1 = (s_1, \cdot)$ contains $u_2 = (s_2, \cdot)$, iff the subspace covered by u_1 is a superset of the subspace covered by u_2, i.e. $s_1 \subseteq s_2$. Let the function σ_t map a set of unities (at time t) to a subspace of S as follows:

$$\sigma_t : \mathcal{P}(U_t) \mapsto \mathcal{P}(S) \quad \text{with} \quad \sigma_t(U') = \bigcup_{(s,\cdot)\,\in\, U'} s \tag{3.6}$$

Let $u_0, u_1, \ldots, u_n \in U_t$ be unities for a given time t and some $n > 0$. If it is the case that $\sigma_t(\{u_1, u_2, \ldots, u_n\}) \subseteq \sigma_t(\{u_0\})$, then u_0 contains the unities $u_1, u_2, \ldots, u_n$ and in turn those unities are components of u_0.

Next to unities and properties the obeserver is able to identify a *relation* between two unities. A relation between two unities is of a certain type. Let B be the set of all possible relation types, then the set of all possible relations can be defined as[5]:

$$\bar{R}_t := B \times \bar{U}_t \times \bar{U}_t \tag{3.7}$$

A set of relations between a set of unities can form a *structure*, if there is a unity that contains all unities from the set.[6] The set of all possible structures can be noted as:

$$\bar{K}_t := \{k \mid k \subseteq \bar{R}_t\} \tag{3.8}$$

This would be the analytical view, where structure is discovered for a given, still simple unity. However, a synthetical view is also possible, where a set of unities form a structure and the space covered by the structure can be used to distinguish a new unity, if an appropriate property exists.

[5]In principle, a relation could also be formalised not as binary predicate (with domain $\{\text{true}, \text{false}\}$), but as certainty/probability (with domain $[0, 1]$), positive- or real-valued extent (with domains $\mathbb{R}^+$ or $\mathbb{R}$). This might be possibly useful generalisation for future consideration, but the added complexity was not required for the present goals.

[6] "[...] by making reference to the process of construction as well as to the components of a construct, it [structure] refers to the actual components and to the actual relations which these must satisfy in their participation in the constitution of a given unity." (Varela et al., 1974, p.15)

To capture this more formally, we define the function ρ_t for a given t that maps a set of relations to the set of all unities involved in those relations:

$$\rho_t : \mathcal{P}(R_t) \mapsto \mathcal{P}(U_t) \quad \text{with} \quad \rho_t(R') = \{u \mid (\cdot, u_1, u_2) \in R' \ \wedge \ (u = u_1 \ \vee \ u = u_2)\} \tag{3.9}$$

Now the synthetic view can be described as: Given a time t and a structure $k \in K_t$ for which $s = \sigma_t(\rho_t(k))$, a unity can be established using $q_a(x)$ for a spacetime $x = (s, t)$ and a suitable property $a \in A$.

Depending on the properties A the observer is able to distinguish it is possible to identify different classes of structures this way. However, this is only useful for unities that effectively have a very rigid, immutable structure such as e.g. a crystal. Hence it would not be useful for systems (see next section) that change over time. For those systems it is not the structure that remains constant, but its *organisation*, which is the "set of relations between its components that define it as a system of a particular class".[7] Formally the set of possible organisations is defined as:

$$\bar{O}_t := \{o \mid o \subseteq \bar{K}_t\} \tag{3.10}$$

Analogously to the above, the synthetic view for organisations can be described as: Given a time t and an organisation $o \in O_t$ for which $s = \sigma_t(\rho_t(o))$, a unity can be established using $q_a(x)$ for a spacetime $x = (s, t)$ and a suitable property $a \in A$.

Similar to the set of known unities, the set of all possible structures is a superset of an observer's set of known structures $K_t \subseteq \bar{K}_t$ for a given t.

3 Processes, networks, and preservation

Until here all considerations so far were applicable to one point in time for a snapshot of the observer's environment. Let's now turn to how unities change over time.

The persistence and transformation of unities over time is captured by processes. A *process* transforms unities over time into other unities or destructs them. An observer can distinguish a process based on the unities that go into the process and the unities that come out of the process. Naturally a process can only start with at least one unity that goes into the process and at least one unity that comes out of the process.

$$\begin{aligned} \bar{P} := \{(U', V') \mid U', V' \subseteq \bar{U} \ \wedge \\ (\exists(\cdot, t_1) \in U' : \forall(\cdot, t_2) \in V' : t_1 < t_2) \ \wedge \\ (\exists(\cdot, t_2) \in V' : \forall(\cdot, t_1) \in U' : t_1 < t_2)\} \end{aligned} \tag{3.11}$$

The definition of a process above is very flexible, which makes further description more complicated than necessary to achieve the goals of this thesis. Hence further considerations should be restricted mostly to processes where the sets of incoming and outgoing unities all share the same moment in time, respectively. This restriction will still allow to sufficiently formalise autopoietic theory, but can be loosened at a later point in time. The set of all processes that have incoming unities at t_1 and outgoing unities at t_2 is:

$$\bar{P}_{t_1,t_2} := \{p \mid p \in \bar{P} \ \wedge \ p = (U', V') \ \wedge \ U' \subseteq \bar{U}_{t_1} \ \wedge \ V' \subseteq \bar{U}_{t_2}\} \tag{3.12}$$

[7] "the organization of a system as the set of relations between its components that define it as a system of a particular class, is a subset of the relations included in its structure" (Maturana & Varela, 1980, p.xx)

Under certain conditions processes can be "concatenated" with each other. Let there be three points in time $t_1, t_2, t_3 \in T$ with $t_1 < t_2 < t_3$ and two processes $p_1 \in \bar{P}_{t_1,t_2}$ and $p_2 \in \bar{P}_{t_2,t_3}$. With $p_1 = (U_1', V_1')$ and $p_2 = (U_2', V_2')$ let $V_1' \cap U_2' \neq \emptyset$, i.e. there is at least one unity that p_1 ends with and p_2 starts with. Then p_1 and p_2 can be concatenated into a *network* of processes. This step of course can be repated for already concatenated processes to form a a much bigger network of processes.

As another way of abstraction and simplification, we allow a network of processes to be combined into new processes. This will come in useful a bit further below. Given a set of processes, the combination of those processes is again a process with each unity removed that is simultaneously an element of the incoming unities for a process as well as an element of the outgoing unities for another process.

$$\begin{aligned}
&\pi : \mathcal{P}(\bar{P}) \mapsto \bar{P} \\
&\pi(P') = (U_1' \setminus U_2',\ U_2' \setminus U_1') \\
\text{with}\quad &U_1' = \{U' \mid (U', \cdot) \in P'\},\ U_2' = \{U' \mid (\cdot, U') \in P'\}
\end{aligned} \tag{3.13}$$

This operation of process combination allows talking about certain qualities of single processes as well as qualities of networks of processes (that follow when processes of a network are combined into one).

Based on the definition of a process above we can now define several classes of processes that perserve certain qualities over time. A simple example is the set of processes that maintains that the same space is covered by the respective total of incoming and outgoing unities for a given start and end time for the process (incoming and outgoing unities respectively).

$$\bar{P}^{\text{space}}_{t_1,t_2} := \{p \mid p \in \bar{P}_{t_1,t_2} \ \wedge\ p = (U_1, U_2) \ \wedge\ \sigma_{t_1}(U_1) \subseteq \sigma_{t_2}(U_2)\} \tag{3.14}$$

Another class of processes is those that preserve covered space and also maintain unity properties. The next definitions preserve one or a set of properties, respectively.

$$\bar{P}^{\text{prop}}_{t_1,t_2,a} := \{p \mid p \in \bar{P}^{\text{space}}_{t_1,t_2} \ \wedge\ p = (U_1, \cdot) \ \wedge\ \forall (s, \cdot) \in U_1 : q_a((s, t_1)) = q_a((s, t_2))\} \tag{3.15}$$

$$\bar{P}_{t_1,t_2,A'} := \bigcap_{a \in A'} \bar{P}_{t_1,t_2,a} \tag{3.16}$$

If a structure was distinguished for the set of incoming unities and the contributing relations are time-independent (i.e. only depend on space and unity properties), this class of processes is also structure-preserving.

While structure-preserving processes are important to describe e.g. crystals over time, perservation of organisation is crucial to autopoietic theory. As a modelling tool the expression $F[Z, Z']$ is used to denote the set of all injective[8] functions from set Z to set Z'.

$$F[Z, Z'] := \{f \mid (f : Z \mapsto Z') \ \wedge\ \forall z_1, z_2 \in Z : f(z_1) = f(z_2) \Rightarrow z_1 = z_2\} \tag{3.17}$$

Now the class of processes that preserve relations can be defined for which a mapping exists between the unities from the first set of relations to the second set of relations. Both sets of relations have include unities that exist one at a single point in time, respectively.

[8]An injective function is a function where each element in the function's range is mapped to by at most one element of the function's domain.

$$\bar{P}^{\text{rel}}_{R'_1,R'_2} := \{p \mid t_1, t_2 \in T \ \wedge \ t_1 < t_2 \ \wedge \ R'_1 \subseteq \bar{R}_{t_1} \ \wedge \ R'_2 \subseteq \bar{R}_{t_2} \ \wedge \\ p \in \bar{P}_{t_1,t_2} \ \wedge \ p = (U_1, U_2) \ \wedge \ f \in F[U_1, U_2] \ \wedge \\ \forall (b, u_1, u_2) \in R'_1 : (b, f(u_1), f(u_2)) \in R'_2\} \tag{3.18}$$

Based on the the relation-preserving processes the set of organisations can be defined that are preserved between two points in time.[9]

$$\bar{O}_{t_1,t_2} := \{(o_1, o_2) \mid o_1 \in \bar{O}_{t_1} \ \wedge \ o_2 \in \bar{O}_{t_2} \ \wedge \ \exists p \in \bar{P}^{\text{rel}}_{o_1,o_2}\} \tag{3.19}$$

In the definition above as well as below, the explicit inclusion of a start and end time for the existence of an organisation over time is important. No sufficiently complex physical entity (no composed unity in spacetime) can be expected to have existed or will exist indefinitely. All known autopoietic systems too have had a time where they did not be or when they will be disintegrated. To make a meaningful claim about whether a system is autopoietic or not (preserves organisation or not) it is important to also state the temporal boundaries in which the system displays autopoiesis.

Above a network of processes was introduced as a set of concatenated processes. An autopoietic network of processes is a 3-tuple that includes a network of processes and the organisation respectively at two different points in time. To avoid triviality, only those networks of processes are allowed that constist of a minimal number of processes and still preserves organisation.

$$\bar{N}_{t_1,t_2} := \{(o_1, o_2, P') \mid o_1 \in \bar{O}_{t_1} \ \wedge \ o_2 \in \bar{O}_{t_2} \ \wedge \ P' \subseteq \bar{P} \ \wedge \\ \pi(P') \in \bar{P}^{\text{rel}}_{o_1,o_2} \ \wedge \ \forall p \in P' : \pi(P' \setminus \{p\}) \notin \bar{P}^{\text{rel}}_{o_1,o_2}\} \tag{3.20}$$

3.3 Mechanistic computation

1 Preliminaries

The common understanding of computation is largely based on Turing's description of Turing machines, which Piccinini called *digital computation*. This was in contrast to the more general notion of *generic computation*, which is described formally in this section. Piccinini explicated generic computation in various works in very similar ways, but one detailed version (Piccinini, 2015) outlined as below.[10]

Before we get to the definition of generic computation below a short comment should be made on instrumental definitions. Piccinini describes a "functional mechanism" (Piccinini, 2015, p.119), which will not be modelled, but their contribution to the definition of generic computation should be outlined. He defines it as follows:

> *A system X is a* functional mechanism *just in case it consists of a set of spatiotemporal components, the properties (causal powers) that contribute to the system's teleological functions, and their organization, such that X possesses its capacities because of how X's components and their properties are organized.*

[9]Note that $\bar{O}_{t_1,t_2}$ must not be confused with $\bar{O}_t$ defined earlier.
[10]Similarly, see (Piccinini & Scarantino, 2011, p.10) or (?, ?, p.458) for alternative, similar descriptions.

There are a few notions to unpack. Piccinini describes "spatiotemporal components" as realisation of entities[11]than can be arranged such that they form bigger components (complex entities)[12] Such an arrangement is called "organization", i.e. components can be organized. (Complex) components can have "properties", which bring about "causal powers"[13] that are the forces that over time causally influence other components[14]. The account of causal powers share similarities with the account of dispositions or natural laws[15]. A "capacity" can be read as one of the functions, behaviours, or activities of a system[16]. When a system is maintaining a certain capacity over time then this is termed a stable contribution. Mechanisms then are complex entities that make such a stable contribution[17]. A "teleological function" of a mechanism is a stable contribution in some wider (spatiotemporal) scope the mechanism is part of[18]. For a more detailed explanation on functional mechanisms or a discussion on teleological notions please refer to the original literature.

We can now turn to the definition of generic computation, which is described in full (Piccinini, 2015, p.121) as:

> ***Generic Computation****: the processing of vehicles by a functional mechanism according to rules that are sensitive solely to differences between different portions (i.e., spatiotemporal parts) of the vehicles.*
> ***Rule****: a mapping from inputs* ***I*** *(and possibly internal states* ***S****) to outputs* ***O****.*

In other words, when the teleological function of a functional mechanism is to compute, then the activity performed by the functional mechanism is computation and can be summarised more abstractly as generic computation like described above. The quoted lines above are preceded by another definition essential to Piccinini's theory and may help to drive the point home:

> *A* ***Physical Computing System*** *is a physical system with the following characteristics:*
>
> - *It is a functional mechanism—that is, a mechanism that has teleological functions.*
> - *One of its teleological functions is to perform computations.*
>
> *[. . .]*
> *In other words, a physical computing system is a mechanism whose teleological func-*

[11] "I assume an ontology of particulars (entities) [. . .]" (Piccinini, 2015, p.105)

[12] "When many entities are organized together in various ways, they form (constitute) more complex entities." (Piccinini, 2015, p.105)

[13] "I assume an ontology of particulars (entities) and their properties understood as causal powers." (Piccinini, 2015, p.105)

[14] "But physical things have causal powers that they exert through spacetime—they do things (push and pull, attract and repel, etc.). By contrast, abstract entities are supposed to have no spatiotemporal location and no causal powers." (Piccinini, 2015, p.59)

[15] "According to a popular account of dispositions, dispositions are causal powers (and vice versa). [. . .] In other words, a plausible view is that a physical system has causal powers C if and only if it has dispositions whose manifestations are C, and this is the case if and only if the system follows natural laws corresponding to C." (Piccinini, 2015, p.21)

[16] "Mechanistic explanation is the explanation of the capacities (functions, behaviors, activities) of a system as a whole in terms of some of its components, their properties and capacities (including their functions, behaviors, or activities), and the way they are organized together." (Piccinini, 2015, p.84)

[17]use quote "The subsystems that constitute complex entities and make stable causal contributions to their behavior are what I call *mechanisms*." (Piccinini, 2015, p.105)

[18] "A teleological function (generalized) is a stable contribution to a goal (either objective or subjective) of organisms by either a trait or an artifact of the organisms." (Piccinini, 2015, p.116)
Please note that for the sake of brevity the exact notions of "goal" and "organism" are not explained further here.

tion is computing mathematical function f from inputs $\boldsymbol{I}$ (and possibly internal states $\boldsymbol{S}$) to outputs $\boldsymbol{O}$.

2 Formalisation

Now the parts that make up generic computation (and in which instances of generic computation can differ) can be investigated further. The approach taken here bears similarities to that in section *3.2 Autopoietic theory*, hence some reasoning and details will be kept brief.

In the definition of generic computation and functional mechanisms the notion of space and time is mentioned, which shall be the starting point for the formalisation. More formally, a space S of a given number of dimensions and a (possibly continuous) set of moments in time T is assumed. The functional mechanism consists of (possibly complex) entities that are spatially extended, each i.e. cover a subspace of $S' \subseteq S$ and exist at least at one time $t \in T$. Then the set of all possible entities becomes:

$$E \subseteq \mathcal{P}(S) \times \mathcal{P}(T) \tag{3.21}$$

An entity may have one or more properties. Let C be the set of all possible properties, then the set of entities that have a property $c \in C$ can be captured by:

$$E_c \subseteq E \tag{3.22}$$

The functional mechanism and the vehicles have spatiotemporal parts, making an instance of generic computation covering a (changing) space over time. However, the subspaces covered by the functional mechanism and the vehicles respectively do not overlap, if they exist at the same point in time. So if for some $t \in T$ there is a functional mechanism whose entities $E' \subseteq E$ together cover subspace S'_0 and the subspaces $S'_1, S'_2, \ldots, S'_n$ of n vehicles for some $n > 0$, then $S' = \bigcup_{i \in \{0..n\}} S'_i$ is the space covered by the instance of generic computation for time t.

For a given $t \in T$ and $S' \subseteq S$ it may still be possible to conceive two different vehicles for the same t and S'. Hence vehicles are more than just their spatiotemporal aspects, they are defined by their properties too. Likewise the properties for entities in the mechanism are important, as in both cases the properties influence the causal power of the system and hence also influence the processing done by the functional mechanism. While for the functional mechanism the total of all entities can be seen as one entity holding a certain property, vehicles need to be considered separately.

The teleologic function, i.e. what the mechanism does is specified by one or more rules. A rule is given as a mapping from (spatiotemporal) vehicles that make up the input (and possibly internal state) to (spatiotemporal) vehicles that make up the output.

Let (e, x, y, M) be an instance of generic computation. $e \in E$ is an entity that includes the (spatiotemporal) functional mechanism. x and y are tuples that describe the vehicles that form the inputs (and possibly internal states) and outputs of the computation, respectively. Each element of the tuples is a pair of an entity and a property type. M is a set of mappings that captures the rule(s) of the instance of generic computation. A mapping is a pair (2-tuple) of tuples for inputs (and possibly internal states) and outputs of the computation, respectively. Each tuple is holding a truth value corresponding to the corresponding vehicle definition earlier (of spatiotemporal component and property type), denoting whether the vehicle's component has the property coded by the property type. In formal terms:

$$\begin{aligned}
& e \in E \quad m, n > 0 \\
& x = (x_1, x_2, \ldots, x_m) && \text{with} && \forall_{i\in\{1..m\}}\colon x_i \in E \times C \\
& y = (y_1, y_2, \ldots, y_n) && \text{with} && \forall_{i\in\{1..n\}}\colon y_i \in E \times C \\
M \subseteq \{\, (v, w) \mid\ & v = (v_1, v_2, \ldots, v_m) && \wedge && \forall_{i\in\{1..m\}}\colon (e, c) = x_i \ \wedge\ v_i = (e \in E_c) \\
& w = (w_1, w_2, \ldots, w_n) && \wedge && \forall_{i\in\{1..n\}}\colon (e, c) = y_i \ \wedge\ w_i = (e \in E_c) \,\}
\end{aligned} \tag{3.23}$$

As a concrete example, consider an AND-gate as protypical example for computation. Assume that $e \in E$ covers the spatiotemporal extension of the functional mechanism, then (e, x, y, M) describes an instance of generic computation following the rule of (or having the teleological function of) an AND-gate based on the following assignments:

$$\begin{aligned}
E &= \{e, e_1, e_2, e_3\} \\
C &= \{c\} \\
x &= ((e_1, c), (e_2, c)) \\
y &= ((e_3, c)) \\
M &= \{((F, F), (F)), ((F, T), (F)), ((T, F), (F)), ((T, T), (T))\}
\end{aligned} \tag{3.24}$$

Specifically, the property type c may for example translate to “5 volts” and the entities e_1, e_2, e_3 may effectively be spatiotemporal coordinates pointing to the inputs and outputs of an electronic AND-gate.

3.4 “Computational autopoietic theory”

3.5 Role of the observer