Levels of Abstraction and the Turing Test

Luciano Floridi[1, 2]

[1]Research Chair in Philosophy of Information and GPI, University of Hertfordshire; [2]Faculty of Philosophy and IEG (OUCL), University of Oxford.

Address for correspondence: School of Humanities, University of Hertfordshire, de Havilland Campus, Hatfield, Hertfordshire AL10 9AB, UK; l.floridi@herts.ac.uk

Abstract

An important lesson that philosophy can learn from the Turing Test and computer science more generally concerns the careful use of the method of Levels of Abstraction (LoA). In this paper, the method is first briefly summarised. The constituents of the method are "observables", collected together and moderated by predicates restraining their "behaviour". The resulting collection of sets of observables is called a "gradient of abstractions" and it formalises the minimum consistency conditions that the chosen abstractions must satisfy. Two useful kinds of gradient of abstraction – disjoint and nested – are identified. It is then argued that in any discrete (as distinct from analogue) domain of discourse, a complex phenomenon may be explicated in terms of simple approximations organised together in a gradient of abstractions. Thus, the method replaces, for discrete disciplines, the differential and integral calculus, which form the basis for understanding the complex analogue phenomena of science and engineering. The result formalises an approach that is rather common in computer science but has hitherto found little application in philosophy. So the philosophical value of the method is demonstrated by showing how making the LoA of discourse explicit can be fruitful for phenomenological and conceptual analysis. To this end, the method is applied to the Turing Test, the concept of agenthood, the definition of emergence, the notion of artificial life, quantum observation and decidable observation. It is hoped that this treatment will promote the use of the method in certain areas of the humanities and especially in philosophy.

1. Introduction

In a previous work ({Floridi, 2008 #103}), I argued that the epistemological use of "levels of abstraction" (LoAs) in philosophical analysis represents a fundamental method, which I labelled *the method of levels of abstraction*. In that context, I clarified the nature and applicability of the method and defended its philosophical fruitfulness by using Kant's classic discussion of the "antinomies of pure reason" as an example. I further supported the method by distinguishing it from three other forms of "levelism" based on (i) levels of organisation; (ii) levels of explanation and (iii) conceptual schemes, and showing its superiority. Still in that context, I addressed the problems of relativism and anti-realism allegedly affecting the use of the method. In this article, I intend to show how the method may be fruitfully applied to a selection of some long-standing philosophical problems. For the convenience of the reader, in section two, I summarise the method without presupposing any previous knowledge. In section three, I show how the method of abstraction may be profitably applied to several philosophical topics. In the conclusion, I briefly recall why the method can be useful in philosophical disputes. A final introductory comment before going to work: the reader interested in the topic of abstraction will probably find Colburn and Shute (2007) enlightening. They trace the connection between abstraction in mathematics and abstraction in computer science. They define mathematical abstraction in terms of *information neglecting*, while "computational" abstraction in terms of *information hiding*. While in this paper I shall speak more positively of computational abstraction in terms of *information selecting*, the idea is the same. In mathematics, abstraction is a matter of essentialism. In computer science it is not that one does know but chooses to ignore but, rather, that one chooses what may or need to be observable and hence knowable. In computer science, abstraction is a matter of constructionism. Hence, this paper might be read as sequel to Colburn and Shute (2007), which it complements by linking their discussion of mathematical and computational abstraction to the discussion of philosophical abstraction.

2. The Method of Abstraction: a Quick Guide

A reliable way to introduce the method of levels of abstraction (LoAs) is by clarifying first the basic concepts used to define LoAs and then show a few examples.

As is well known, a "typed variable" is a variable qualified to hold only a declared kind of data:

Definition. A *typed variable* is a uniquely-named conceptual entity (the *variable*) and a set, called its *type*, consisting of all the values that the entity may take. Two typed variables are regarded as *equal* if and only if their variables have the same name and their types are equal as sets. A variable that cannot be assigned well-defined values is said to constitute an *ill-typed variable* (see the example below).

When required, I shall write $x{:}X$ to mean that x is a variable of type X.

The notion of an "observable" is common in science, where it occurs whenever a (theoretical) model is constructed. Although the way in which the features of the model correspond to the system being modelled is usually left implicit in the process of modelling, it is important here to make that correspondence explicit. I shall follow the standard practice of using the word "system" to refer to the object of study. This may indeed be what would normally be described as a system in science or engineering, but it may also be a domain of discourse, of analysis, of conceptual speculation, or a purely semantic system (imagine for example, the system of Greek divinities).

Definition. An *observable* is an interpreted typed variable, that is, a typed variable together with a statement of what feature of the system under consideration it represents. Two observables are regarded as *equal* if and only if their typed variables are equal, they model the same feature and, in that context, one takes a given value if and only if the other does.

Being an abstraction, an observable is not necessarily meant to result from quantitative measurement or even empirical perception. The "feature of the system under consideration" might be a physical magnitude, but we shall see that it might also be an artefact of a conceptual model, constructed entirely for the purpose of analysis.

An observable, being a typed variable, has specifically determined possible values. In particular:

Definition. An observable is called *discrete* if and only if its type has only finitely many possible values; otherwise it is called *analogue*.[1]

[1] The distinction is really a matter of topology rather than cardinality. However, this definition serves our present purposes.

When working with the method of abstraction, one is interested in observables as a means of describing behaviour at a precisely qualified (although seldom numerical) level of abstraction; in general, several observables will be employed. Let us now consider a few simple examples.

1) The design of a database is a special case of the definition of a collection of observables. In a database, an observable is called a key, its correspondence with reality is left implicit (although it is often reflected in the name) and its type is inferred from either the values it takes or from its declaration in the database programming language. The type "finite string of characters", for instance, is frequently used, often being the most appropriate concrete method of description. Examples include names, addresses and such like. So too with an observable in general: it is sometimes preferable not to provide in advance all the possible outcomes (i.e. a type) but simply to define its type to consist of all finite sequences of characters, with each value equal to a character string. This definition reflects a decision to the effect that, although the observable is well-typed, its actual type is not of primary concern.

2) Suppose Peter and Ann wish to study some physical human attributes. To do so Peter, in Oxford, introduces a variable, *h*, whose type consists of rational numbers. The typed variable *h* becomes an (analogue) observable once it is decided that the variable *h* represents the height of a person, using the Imperial system (feet and parts thereof). To explain the definition of equality of observables, suppose that Ann, in Rome, is also interested in observing human physical attributes, and defines the same typed variable but declares that it represents height in metres and parts thereof. Their *typed variables* are the same, but they differ as *observables*: for a given person, the two variables take different representing values. This example shows the importance of making clear the interpretation by which a typed variable becomes an observable.

3) Consider next an example of an *ill-typed variable*. Suppose we are interested in the roles played by people in some community; we could not introduce an observable standing for those beauticians who depilate just those people who do not depilate themselves, for it is well-known that such a variable would not be well typed (Russell 1902). Similarly, each of the standard antinomies reflects an ill-typed variable (Hughes and Brecht 1976). Of course, we are at liberty to choose whatever type befits the application, and if that involves a potential antinomy then the appropriate type might turn out to be a non-well-founded set (Barwise and Etchemendy 1987). However, in this paper we shall operate entirely within the boundaries of standard naive set theory.

4) Gassendi provides another nice example. As he wrote in his *Fifth Set of Objections to Descartes' Meditations* "If we are asking about wine, and looking for the kind of knowledge

which is superior to common knowledge, it will hardly be enough for you to say 'wine is a liquid thing, which is compressed from grapes, white or red, sweet, intoxicating' and so on. You will have to attempt to investigate and somehow explain its internal substance, showing how it can be seen to be manufactured from spirits, tartar, the distillate, and other ingredients mixed together in such and such quantities and proportions." What Gassendi seems to have in mind is that observables relating to tasting wine include the attributes that commonly appear on "tasting sheets": *nose* (representing bouquet), *legs* or *tears* (viscosity), *robe* (peripheral colour), *colour*, *clarity*, *sweetness*, *acidity*, *fruit*, *tannicity*, *length* and so on, each with a determined type. If two wine tasters choose different types for, say, *colour* (as is usually the case) then the observables are different, despite the fact that their variables have the same name and represent the same feature in reality. Indeed, as they have different types they are not even equal as typed variables.

Information about how wine quality is perceived to vary with time – how the wine "ages" (Robinson 1989) – is important for the running of a cellar. An appropriate observable is the typed variable *a*, which is a function associating to each year *y*: *Years* a perceived quality *a*(*y*):*Quality*, where the types *Years* and *Quality* may be assumed to have been previously defined. Thus, *a* is a function from *Years* to *Quality*, written a: $Time \rightarrow Quality$. This example shows that, in general, types are constructed from more basic types, and that observables may correspond to operations, taking input and yielding output. Indeed, an observable may be of an arbitrarily complex type.

5) The definition of an observable reflects a particular view or attitude towards the entity being studied. Most commonly, it corresponds to a simplification, in which case nondeterminism, not exhibited by the entity itself, may arise. The method is successful when the entity can be understood by combining the simplifications. Let us consider another example. In observing a game of chess, one would expect to record the moves of the game.[2] Other observables might include the time taken per move, the body language of the players, and so on. Suppose we are able to view the chessboard by looking just along *files* (the columns stretching from player to player). When we play "files-chess", we are unable to see the ranks (the parallel rows between the players) or the individual squares. Files cannot sensibly be attributed a colour black or white, but each may be observed to be occupied by a set of pieces (namely those that appear along that file), identified in the usual way (king, queen and so forth). In "files-chess", a move

[2] As the reader probably knows, this is done by recording the history of the game: move by move the state of each piece on the board is recorded – in English algebraic notation – by rank and file, as the piece being moved and the consequences of the move.

may be observed by the effect it has on the file of the piece being moved. For example, a knight moves one or two files either left or right from its starting file, a bishop is indistinguishable from a rook, which moves along a rank, and a rook that moves along a file appears to remain stationary. Whether or not a move results in a piece being captured, appears to be nondeterministic. "Files-chess" seems to be an almost random game. Whilst the "underlying" game is virtually impossible to reconstruct, each state of the game and each move (i.e. each operation on the state of the game) can be "tracked" with this dimensionally-impoverished family of observables. If one then takes a second view, corresponding instead to rank, we obtain "ranks-chess". Once the two views are combined, the original game of chess can be recovered, since each state is determined by its rank and file projections, and similarly for each move. The two disjoint observations together, that is, "files-chess" + "ranks-chess", reveal the underlying game.

6) The degree to which a type is appropriate depends on its context and use. For example, to describe the state of a traffic light in Rome one might decide to consider an observable *colour* of type {*red*, *amber*, *green*} that corresponds to the colour indicated by the light. This option abstracts the length of time for which the particular colour has been displayed, the brightness of the light, the height of the traffic light, and so on. This is why the choice of type corresponds to a decision about how the phenomenon is to be regarded. To specify such a traffic light for the purpose of construction, a more appropriate type would comprise a numerical measure of wavelength. Furthermore, if we are in Oxford, the type of colour would be a little more complex, since – in addition to red, amber and green – red and amber are displayed simultaneously for part of the cycle. So, an appropriate type would be {*red*, *amber*, *green*, *red-amber*}.

With the previous examples in place, we are now ready to appreciate the basic concept of *level of abstraction* (LoA). Any collection of typed variables can, in principle, be combined into a single "vector" observable, whose type is the Cartesian product of the types of the constituent variables. In the wine example, the type *Quality* might be chosen to consist of the Cartesian product of the types *Nose*, *Robe*, *Colour*, *Acidity*, *Fruit* and *Length*. The result would be a single, more complex, observable. In practice, however, such vectorisation is unwieldy, since the expression of a constraint on just some of the observables would require projection notation to single out those observables from the vector. Instead, I shall base our approach on a *collection* of observables, that is, a level of abstraction:

Definition. A *level of abstraction* (*LoA*) is a finite but non-empty set of observables. No order is assigned to the observables, which are expected to be the building blocks in a theory characterised by their very definition. A LoA is called *discrete* (respectively *analogue*) if and only if all its observables are discrete (respectively analogue); otherwise it is called *hybrid*.

Consider Gassendi's wine example. Different LoAs may be appropriate for different purposes. To evaluate a wine, the "tasting LoA", consisting of observables like those mentioned in the previous section, would be relevant. For the purpose of ordering wine, a "purchasing LoA" (containing observables like *maker*, *region*, *vintage*, *supplier*, *quantity*, *price*, and so on) would be appropriate; but here the "tasting LoA" would be irrelevant. For the purpose of storing and serving wine – the "cellaring LoA" (containing observables for *maker*, *type of wine*, *drinking window*, *serving temperature*, *decanting time*, *alcohol level*, *food matchings*, *quantity remaining in the cellar*, and so on) would be relevant.

Not all values exhibited by combinations of observables in a LoA may be realised by the system being modelled. For example, if the four traffic lights at an intersection are modelled by four observables, each representing the colour of a light, the lights cannot in fact all be green together (assuming they work properly). In other words, the combination in which each observable is green cannot be realised in the system being modelled, although the types chosen allow it. Similarly, the choice of types corresponding to a rank-and-file description of a game of chess allows any piece to be placed on any square, but in the actual game two pieces cannot occupy the same square simultaneously. Some technique is therefore required to describe those combinations of observable values that are actually acceptable. The most general method is simply to describe all the allowed combinations of values. Such a description is determined by a predicate whose allowed combinations of values is called the "system behaviours".

Definition. A *behaviour* of a system, at a given LoA, is defined to consist of a predicate whose free variables are observables at that LoA. The substitutions of values for observables that make the predicate true are called the *system behaviours*. A *moderated LoA* is defined to consist of a LoA together with a behaviour at that LoA.

Consider two previous examples. In reality, human height does not take arbitrary rational values, for it is always positive and bounded above by (say) nine feet. The variable *h*, representing height, is therefore constrained to reflect reality by defining its behaviour to

consist of the predicate $0 < h < 9$, in which case any value of h in that interval is a "system" behaviour. Likewise, wine too is not realistically described by arbitrary combinations of the aforementioned observables. For instance, it cannot be both white and highly tannic.

Since Newton and Leibniz, the behaviours of the analogue observables, studied in science, have typically been described by differential equations. A small change in one observable results in a small, quantified change in the overall system behaviour. Accordingly, it is the rates at which those smooth observables vary which is most conveniently described.[3] The desired behaviour of the system then consists of the solution of the differential equations. However, this is a special case of a predicate: the predicate holds at just those values satisfying the differential equation. If a complex system is approximated by simpler systems, then the differential calculus provides a supporting method for quantifying the approximation.

The use of predicates to demarcate system behaviour is essential in any (nontrivial) analysis of discrete systems because in the latter no such continuity holds: the change of an observable by a single value may result in a radical and arbitrary change in system behaviour. Yet, complexity demands some kind of comprehension of the system in terms of simple approximations. When this is possible, the approximating behaviours are described exactly, by a predicate, at a given LoA, and it is the LoAs that vary; becoming more comprehensive and embracing more detailed behaviours, until the final LoA accounts for the desired behaviours. Thus, the formalism provided by the method of abstraction can be seen as doing for discrete systems what differential calculus has traditionally done for analogue systems.

Likewise, the use of predicates is essential in subjects like information and computer science, where discrete observables are paramount and hence predicates are required to describe a system behaviour. In particular, state-based methods like *Z* (Hayes and Flinn 1993, Spivey 1992) provide notation for structuring complex observables and behaviours in terms of simpler ones. Their primary concern is with the syntax for expressing those predicates, an issue that will be avoided in this paper by stating predicates informally.

The time has come now to combine approximating, moderated LoAs to form the primary concept of the method of abstraction.

For a given (empirical or conceptual) system or feature, different LoAs correspond to different representations or views. A *Gradient of Abstractions* (GoA) is a formalism defined to

[3] It is interesting to note that the catastrophes of *chaos theory* are not smooth; although they do appear so when extra observables are added, taking the behaviour into a smooth curve on a higher-dimensional manifold. Typically, chaotic models are weaker than traditional models, their observables merely reflecting *average* or *long-term* behaviour. The nature of the models is clarified by making explicit the LoA.

facilitate discussion of discrete systems over a range of LoAs. Whilst a LoA formalises the scope or granularity of a single model, a GoA provides a way of varying the LoA in order to make observations at differing levels of abstraction. For example, in evaluating wine one might be interested in the GoA consisting of the "tasting" and "purchasing" LoAs, whilst in managing a cellar one might be interested in the GoA consisting of the "cellaring" LoA together with a sequence of annual results of observation using the "tasting" LoA.

In general, the observations at each LoA must be explicitly related to those at the others; to do so, one uses a family of relations between the LoAs. For this, one needs to recall some (standard) preliminary notation.

Notation. A *relation* R from a set A to a set C is a subset of the Cartesian product $A \times C$. R is thought of as relating just those pairs (a, c) that belong to the relation. The *reverse* of R is its mirror image: $\{(c, a) \mid (a, c) \in \Box R\}$. A relation R from A to C translates any predicate p on A to the predicate $P_R(p)$ on C that holds at just those $c{:}C$, which are the image through R of some $a{:}A$ satisfying p

$$P_R(p)(c) = \exists a{:}\ A\ R(a,c) \wedge p(a)$$

We have finally come to the main definition of the paper:

Definition. A *gradient of abstractions*, *GoA*, is defined to consist of a finite set[4] $\{L_i \mid 0 \le i < n\}$ of moderated LoAs L_i, a family of relations $R_{i,j} \subseteq L_i \times L_j$, for $0 \le i \ne j < n$, relating the observables of each pair L_i and L_j of distinct LoAs in such a way that:

1. the relationships are inverse: for $i \ne j$, $R_{i,j}$ is the reverse of $R_{j,i}$
2. the behaviour p_j at L_j is at least as strong as the translated behaviour

$$P_{R_{i,j}}(p_i)\ p_j \Rightarrow P_{R_{i,j}}(p_i). \qquad (1)$$

and for each interpreted type $x{:}X$ and $y{:}Y$ in L_i and Lj respectively, such that $(x{:}X, y{:}Y)$ is in R_{ij}, a relation $Rxy \subset X \times Y$.

Two GoAs are regarded as *equal* if and only if they have the same moderated LoAs (i.e. the same LoAs and moderating behaviours) and their families of relations are equal. A GoA is called *discrete* if and only if all its constituent LoAs are discrete.

[4] The case of infinite sets has application to analogue systems but is not considered here.

Condition (1) means that the behaviour moderating each lower LoA is *consistent* with that specified by a higher LoA. Without it, the behaviours of the various LoAs constituting a GoA would have no connection to each other. A special case, to be elaborated below in the definition of "nestedness", helps to clarify the point.

If one LoA L_i extends another L_j by adding new observables, then the relation $R_{i,j}$ is the inclusion of the observables of L_i in those of L_j and (1) reduces to this: the constraints imposed on the observables at LoA L_i remain true at LoA L_j, where "new" observables lie outside the range of $R_{i,j}$.

A GoA whose sequence contains just one element evidently reduces to a single LoA. So our definition of "LoA" is subsumed by that of "GoA".

The consistency conditions imposed by the relations $R_{i,j}$ are in general quite weak. It is possible, though of little help in practice, to define GoAs in which the relations connect the LoAs cyclically. Of much more use are the following two important kinds of GoA: "disjoint" GoAs (whose views are complementary) and "nested" GoAs (whose views provide successively more information). Before defining them some further notation needs to be introduced. We are now ready to appreciate the following definition of GoA:

Definition. A GoA is called *disjoint* if and only if the L_i are pairwise disjoint (i.e. taken two at a time, they have no observable in common) and the relations are all empty. It is called *nested* if and only if the only nonempty relations are those between L_i and L_{i+1}, for each $0 \leq i < n-1$, and moreover the reverse of each $R_{i,\,i+1}$ is a surjective function from the observables of L_{i+1} to those of L_i.

A disjoint GoA is chosen to describe a system as the combination of several non-overlapping components. This is useful when different aspects of the system behaviour are better modelled as being determined by the values of distinct observables. This case is rather simplistic, since the LoAs are more typically tied together by common observations.

A nested GoA (see Figure1) is chosen to describe a complex system exactly at each LoA and incrementally more accurately. The condition that the functions be surjective means that any abstract observation has at least one concrete counterpart. As a result, the translation functions cannot overlook any behaviour at an abstract LoA: behaviours lying outside the range of a function translate to the predicate *false*. The condition that the reversed relations be functions means that each observation at a concrete LoA comes from at most one observation

at a more abstract LoA (although the converse fails in general, allowing one abstract observable to be refined by many concrete observables). As a result the translation functions become simpler.

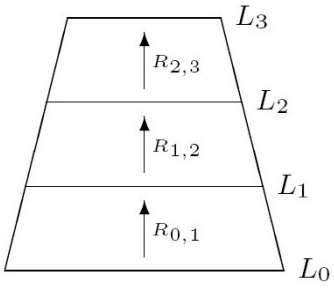

Figure 1 Nested GoA with four Levels of Abstraction

For example, the case of a traffic light which is observed to have colour *colour* of type {*red*, *amber*, *green*} is captured by a LoA, L_0, having that single observable. If one wishes to be more precise about colour, e.g. for the purpose of constructing a new traffic light, one might consider a second LoA, L_1, having the variable *wl* whose type is a positive real number corresponding to the wavelength of the colour. To determine the behaviour of L_1, Suppose that constants $\lambda_{red} < \lambda_{red'}$ delimit the wavelength of red, and similarly for amber and green. Then the behaviour of L_1 is simply this predicate with free variable *wl*

$$(\lambda_{red} \leq wl \leq \lambda_{red'}) \vee (\lambda_{amber} \leq wl \leq \lambda_{amber'}) \vee (\lambda_{green} \leq wl \leq \lambda_{green'}).$$

The sequence consisting of the LoA L_0 and the moderated LoA L_1 forms a nested GoA. Intuitively, the smaller, abstract type {*red*, *amber*, *green*} is a projection of the larger. The relevant relation associates to each value *c*:{*red*, *amber*, *green*} a band of wavelengths perceived as that colour. Formally, *R*(*colour*,*wl*) is defined to hold if and only if, for each *c*:{*red*, *amber*, *green*},

$$colour = c \quad \leftrightarrow \quad \lambda_c \leq wl \leq \lambda_{c'}.$$

In the wine example, a first LoA might be defined to consist of the variable “kind” having type consisting of *red*, *white*, *rose* under the obvious representation. A second LoA might be defined to consist of the observable “kind” having type

{*stillred*, *sparklingred*, *stillwhite*, *sparklingwhite*, *stillrose*, *sparklingrose*}.

Although the second type does not contain the first, it produces greater resolution under the obvious projection relation. Thus, the GoA consisting of those two LoAs is nested.

Those two important forms of GoA – disjoint and nested – are in fact theoretically interchangeable. For if A and B are disjoint sets then A and their union $A \cup B$ are increasing sets and the former is embedded in the latter. Thus, a disjoint GoA can be converted to a nested one. Conversely, if A and B are increasing sets with the former embedded in the latter, then A and the set difference $A \setminus B$ are disjoint sets. Thus, a nested GoA can be converted to a disjoint one.

Following the technique used to define a nested GoA, it is possible to define less restricted but still hierarchical GoAs. Important examples include tree-like structures, of which our nested GoAs are a special, linear case.

For theoretical purposes, the information captured in a GoA can be expressed equivalently as a single LoA of more complicated type, namely one whose single LoA has type equal to the sequence of the LoAs of the complex interface. However, the current definition is better suited to application.

Models are the outcome of the analysis of a system, developed at some LoA(s) for some purpose. An important contribution of these ideas is to make precise the commitment to a LoA/GoA before further elaborating a theory. This is called the *method of abstraction*. Three main advantages of the method can be highlighted here.

First, specifying the LoA means clarifying, from the outset, the range of questions that (a) can be meaningfully asked and (b) are answerable in principle. One might think of the input of a LoA as consisting of the system under analysis, comprising a set of *data*; its output is a *model* of the system, comprising *information*. The quantity of information in a model varies with the LoA: a lower LoA, of greater resolution or finer granularity, produces a model that contains more information than a model produced at a higher, or more abstract, LoA. Thus, a given LoA provides a quantified commitment to the kind and amount of information that can be "extracted" from a system. The choice of a LoA pre-determines the type and quantity of data that can be considered and hence the information that can be contained in the model. So, knowing at which LoA the system is being analysed is indispensable, for it means knowing the scope and limits of the model being developed.

Second, being explicit about the LoA adopted provides a healthy antidote to ambiguities, equivocations and other fallacies or errors due to level-shifting, such as Aristotle's "metabasis eis allo genos" (shifting from one genus to another), Ryle's "category-mistakes", and Kant's "antinomies of pure reason".

Third, by stating its LoA, a theory is forced to make explicit and clarify its ontological commitment. The ontological commitment of a theory is best understood by distinguishing between a *committing* and a *committed* component. A theory commits itself ontologically by opting for a specific LoA. Compare this to the case in which one has chosen a specific kind of car but has not bought one yet. A theory becomes ontologically committed in full through its model, which is therefore the bearer of the specific commitment. The analogy here is with the specific car one has actually bought. So LoAs commit a theory to types, while their ensuing models commit it to the corresponding tokens.

3. Some Philosophical Applications

The time has come to provide some philosophical applications of the method of Levels of Abstraction.

3.1 Agents

An *agent* can be thought of (Floridi and Sanders 2004) as a *transition system* (i.e. a system of states and transitions between them) that is *interactive* (i.e. responds to stimulus by change of state), *autonomous* (i.e. is able to change state without stimulus) and *adaptable* (i.e. is able to change the transition rules by which it changes state). However each of those properties, and hence the definition of agenthood, makes sense only at a prescribed LoA. For example, whether or not a rock is deemed to be interactive depends on the length of time and level of detail of observation. Over a long period, it erodes and hence changes state. By day it absorbs solar radiation which it emits at night. But with observables resulting from scrutiny over a period of ten seconds by the naked eye from ten metres, it can be deemed not to be interactive. If the LoA at which one observes it abstracts gravity and resistance, a swinging pendulum appears to be autonomous but neither interactive nor adaptive. By extending the LoA to incorporate air resistance, it becomes adaptive. By observing also the whistling sound it makes with the air, it becomes interactive. If a piece of software that exhibits machine learning (Mitchell 1997) is studied at a LoA which registers its interactions with its environment, then the software will appear interactive, autonomous and adaptive, i.e. to be an agent. But if the program code is revealed then the software is shown to be simply following rules and hence not to be adaptive.

These two LoAs are at variance. One reflects the "open source" view of software: the user has access to the code. The other reflects the commercial view that, although the user has bought the software and can use it at will, he has no access to the code. At stake is whether or not the software forms an (artificial) agent.

3.2 The Turing test

Turing (1950) took the crucial step of arguing that the ability to think (since called "intelligence") can be satisfactorily characterised by means of a test, rather than by explicit definition. In retrospect, that step may seem a small one. After all, we are quite familiar with areas in which no explicit definition is possible or sensible. Society makes no attempt to characterise what it means to be an acceptable driver in terms of vision, response times, coordination, experience and other physical attributes. Instead, it relies on a driving test. Likewise, society does not attempt to define what it means for a school student to have reached an acceptable academic standard by the end of school; it relies on final school examinations. But incisive that step certainly must have been in view of the vast number of attempts (even to this day) to characterise intelligence explicitly. Opponents of Turing's approach usually object that his test functions at the wrong LoA: perhaps it ought to include a component of creativity, of spontaneity, and so on. However, without concepts like those introduced above, it is very difficult to make one's objections precise or defend Turing's approach. It is therefore of considerable interest to see, first, how the Turing test can be expressed using phenomenological LoAs and, second, how it can be analysed using conceptual LoAs.

3.2.1 Turing's imitation game

Let us start, as did Turing, by considering an imitation game, in which a man A and a woman B are placed in a room separate from an interrogator C, who communicates with each by teleprinter (these days replaced by computer). C puts questions to A and B, known only as X and Y. C's task is to identify $X = A$ and $Y = B$ or, conversely, $X = B$ and $Y = A$, by considering their responses. We might describe that scenario by taking a first, extremely abstract, LoA to reflect just the correctness of C's identification. The LoA L_0 consists of a single variable *ans* of type {*right*, *wrong*} which becomes an observable under the correspondence: *ans* takes the value *right* if C is correct and the value *wrong* if C is incorrect. In choosing this LoA, we are intentionally abstracting the actual answer (whether X was A or B), the questions and answers, response times, and so on in order to capture just the outcome of the imitation game.

We might reveal *C*'s actual identification by defining a second, disjoint, LoA L_1 whose single variable, *Z*, is of type {(*A*,*B*), (*B*,*A*)} which is made into an observable under the correspondence that the first component of *Z* is the putative identity of *X* and the second component that of *Y*. Combining the two LoAs L_0 and L_1 gives a disjoint GoA.

Of course, there are alternative approaches, which is why it is important to be precise about the one taken. We might have defined a GoA by replacing the LoA L_1 with a LoA containing two observables, the first corresponding to the identity of *X* and the second corresponding to the identity of *Y*. That would be more involved, since each would have type {*A*, *B*} and we would have to moderate it with the behaviour that the values of *X* and *Y* differ. Our choice of L_1 avoids the complication by building that behaviour into the type of *Z*. But with several observables, in general such moderating behaviours cannot be avoided.

To model *C*'s questions, the addressees and their responses, we define a third LoA, L_2. Let *Q* and *R* denote the sets of possible (well-posed) questions and responses respectively (an example where the type of text strings may be considered appropriate). Then each "question, addressee and response" triple is a variable whose type is the Cartesian product of *Q*, {*X*, *Y*} and *R*. It becomes an observable under the correspondence just established. The observable we seek now consists of a sequence (of arbitrary but finite length) of such triples, corresponding to the sequence of interactions in temporal order; and L_2 contains that single observable. (An alternative would be to have an observable for the number of questions, an observable for each question and an observable for each response.) L_2 can be added to either GoA *T* or *T*' to obtain a GoA which is still disjoint but has higher resolution.

More detailed LoAs are possible and easy to define but, following Turing, we stop here having appreciated that any discussion of the imitation game may be accurately "calibrated", according to its level of abstraction, with a GoA.

3.2.2 Turing's test

In the Turing test, *A* is replaced by a "machine" (nowadays "computer"). Turing proposed that the question "Can machines think?" be replaced by the question: "Will the interrogator decide wrongly as often when the game is played like this as he does when the game is played between a man and a woman?". These days,[5] the test is normally stripped of its sex-specific nature and

[5]For a summary of the Turing test today, and its incarnation in competitive form (the Loebner prize), see Moor (2001).

the interrogator is simply asked to determine the human from the machine. Appropriate GoAs are defined as above, but with A representing a computer and B a human.

Although Turing did not make it explicit, the phrase "as often" in his description implies repetition of the test, and a conclusion reached by statistical analysis. Suppose that C initiates a pair of question/answer sessions of the type used in the imitation game. A list of questions is put to two situations, one containing a man and a woman, the other containing a machine and a woman. We suppose that the answers in the first situation are of type A_1 and those in the second of type A_2, thus avoiding here the question of whether or not $A_1 = A_2$. As before, C makes an identification. The appropriate LoA has type, call it J, equal to the Cartesian product of the type $\{(A,B), (B,A)\}$ and the type of all sequences of elements of the type $Q \times \{X, Y\} \times A_1 \times A_2$.

The observable corresponding to repetition of that situation j times, though not necessarily with the same questions, has type consisting of the Cartesian product of j-many copies of type J, namely J^j. The LoA incorporating that observation plus the answer to the ultimate question is then the Cartesian product of the type J^j and the type $\{right, wrong\}$. Likewise, a more complex type can be constructed to reveal the nature of the statistical test; in this case too, let us follow Turing and overlook the details.

3.2.3 Turing discussed

The previous two sections have shown how to formalise the Turing test using phenomenologically motivated GoAs. But the method of abstraction can be used to discuss and compare variations on the test. Indeed it is difficult to imagine how such an analysis could be formulated without a concept equivalent to LoA. Now details of the LoAs need not be given as long as it is clear that they could be.

Turing couched his test in terms of a single human interrogator. In the Loebner test, interrogation is provided by a panel of humans interacting via computer interface in real time. Alternatively the interrogator could be a machine; or instead of real-time interaction, a list of pre-arranged questions might be left, and the list of answers returned to the interrogator; or instead of serial questions and answers the interrogator might hold a "general conversation". Each alternative modifies the power of the test. All can be formalised by defining different GoAs. The Turing test might be adapted to target abilities other than "thinking", like interpretation of text, game playing, puzzle solving, spatial reasoning, creativity, and so on.

Contemplating the possible GoAs provides a way to formalise the variant test clearly and elegantly, and promotes simple comparison with contending treatments.

3.3 Emergence

The method of abstraction is ideally suited to the study of systems so complex that they are best understood stepwise, by their gradual disclosure at increasingly fine levels of abstraction. The study of such systems strikes at the heart of the following controversy: how does ultimate complexity arise in the stepwise approximation by simple systems? Gell-Mann (1994) has suggested calling the study of such phenomena *plectics*, and introduces it using an idea he calls *granularity* which is conveniently formalised by LoA.

A key concept in such an approach to complex systems is that of "emergent behaviour", that is, behaviour that arises in the move from one LoA to a finer level (see Hendriks-Jansen 1989 for a discussion). In this section, I apply the method of abstraction to clarify that concept of emergence. In the next, I will shift one particular area in which emergence has received much attention, that of artificial life (ALife).

Emergence, not surprisingly for such an important concept, can take various forms. It derives from the idea that, according to Hendriks-Jansen (1989, p. 283) "properties at higher levels are not necessarily predictable from properties at lower levels". There "lower levels" are more abstract and "higher levels" more detailed or concrete. In this section we are concerned only with the idea of emergence, which is neatly captured using a GoA containing two nested LoAs. Let us consider an example first.

The process of tossing a coin may be modelled abstractly with an observable *outcome* of type {*head*, *tail*} corresponding to the side of the coin that faces upwards after the toss. This LoA abstracts any other result (like the coin's landing on its edge or becoming lost) and other features like manner of tossing, number of spins, time taken and so on. In particular, it models just one toss of the coin and so cannot account for the "fairness" of the coin, which would reveal itself statistically after a large number of tosses. Now, suppose that one wishes to model the repetition of that process with the explicit aim of discussing a coin's fairness. We introduce a more concrete LoA, whose observables are: a natural number n, corresponding to the number of tosses of the coin, and a list of n values from {*head*, *tail*}, corresponding to successive tosses as modelled above. At this second LoA, we are able to make judgements (using standard statistics, for example) about the fairness of the coin, based on the frequency of the outcomes.

This example demonstrates emergence as follows. In many repeated tosses of the coin, the more abstract model applies toss by toss, but does not allow frequency of outcome to be

observed, as it is in the finer model. We say that the notion of the coin's fairness is emergent at the finer LoA. That situation can be formalised as follows. Suppose some system is under consideration using a nested GoA consisting of two LoAs. Suppose the more abstract LoA, A, is moderated by behaviour p_A describing the abstract view of system and the more concrete LoA, C, is moderated by behaviour p_C describing the concrete view. The abstract and concrete observables are related by the total and one-to-many relation $R_{A,C}$. Recall that a behaviour of the system at LoA C is a triple of values for the observables of C that satisfies p_C.

Definition. A behaviour of the system at LoA C is said to be *emergent* (with respect to that nested GoA) if and only if its translation under the relation $R_{A,C}$ fails to satisfy p_A. *Emergence* is said to hold in the GoA if and only if there is some emergent behaviour.

There is frequently confusion about emergence. This is scarcely surprising, since without the notion of LoA, the various levels at which a system is discussed cannot be formalised. Emergence arises typically because the concrete LoA embodies a "mechanism", or rule, for determining an observable, which has been overlooked at the abstract LoA, usually quite deliberately, in order to gain simplicity at the cost of detail. Frequently, the breakthrough in understanding some complex phenomenon has come by accounting for emergent behaviour; and that has resulted from considering the process by which it occurs, rather than taking a more static view of the ingredients involved. In the coin example, we have avoided incorporating any mechanism; but any of the multitude of pseudo-random number generators could be used to generate lists of *head* and *tail* and hence to account for the emergent phenomenon (Knuth 1997).

An interesting example of emergence is provided by quantum mechanics, according to which each action (other than the process of observation) is (unitary and hence) reversible. Yet when observations "in the large" are made of huge physical systems, in spite of the components of those systems obeying the laws of quantum mechanics, the laws of thermodynamics *emerge*: in spite of local reversibility, entropy increases.

The majority of observables considered have been "static". However operations constitute vital observables. Indeed, the importance of "process" may be indicated by the example of a sponge cake. With only the ingredients as observables (i.e. the amount of each ingredient) the sponge-like nature of the cake is, as many a novice cook has found, emergent. But if the manner of aeration (a variable indicating the aerating effect of bicarbonate of soda under the right

conditions) is also an observable, then sponginess is explicable. In other words, the behaviour of sponginess emerges at the finer level of abstraction only.

3.4 Artificial life

Attempts to define artificial life (ALife) have been no more successful than the attempts, both before and after Turing, to define "intelligence" explicitly. Having chosen a LoA *L*, one may propose that ALife with respect to *L* be defined to consist of all those entities observably indistinguishable under *L* from a live entity. A sensible "sanity check" on *L* is that the result does not depend on the particular species of life observed, provided that it is chosen with regard to the concept of life being explored.

To give a rather coarse example, perhaps the most abstract observation possible of a population is its size: the number of members as a function of time. That is an observable of type "whole number" and it is well-typed provided that the population is finite and well defined. At this LoA, a population provides no more information than its size; there is no distinction between live populations, artificial populations and, for that matter, any set which varies in size. However, such a LoA is sufficient to support the well known Fibonacci model of population size per generation (Thompson 1992). It is already of some interest if one adds observables including the rates of birth, mortality, migration and harvesting. It then becomes sufficient, for example, to discuss age distribution across the population and for the management of pest populations, for the harvesting of natural populations, for the modelling of insect outbreaks, of interacting populations and so on (Murray 2003). Of course, at this LoA an observer from space might decide that cars are the dominant life form on parts of earth. A slightly finer LoA is that at which the location of individual members of a population is plotted against time (say in Oxford, every 24 hours). At this "spatial distribution" LoA humans are indistinguishable from ants, bicycles and books: all move around, are created and destroyed. (Were it not for copyright regulations, books might also be "cloned".) The method of abstraction seems to apply well to summarise, clarify and facilitate the comparison of existing contributions. One important approach has been that of Langton (1996), which extends the biological notions of genotype and phenotype to gtype and ptype respectively so that those terms apply equally to ALife. Gtype refers to a low-level implementation mechanism, behaviour or LoA whilst a ptype refers to a higher-level behavioural structure or LoA that emerges as a result of those mechanisms interacting. Langton discusses the subtleties of inferring the latter from the former. His model provides an important instance of the use of a GoA containing two interfaces, one for gtype observables and the other for ptype observables.

3.5 Quantum observation

It is remarkable that the disparate disciplines of quantum mechanics and social anthropology share a fundamental feature: in each, observation automatically involves interference. Observing a quantum or anthropological system is possible only at the expense of a change to the system. By contrast, the definition of observable introduced above makes no assumptions about how the entity is (capable of being) measured (effectively) in practice. In this section I address this issue.

In quantum mechanics, "observable" means something much more restricted than the sense in which I have used it here. There, it is to be distinguished from state that is posited to exist in order to explain the frequency with which observables take their values. Such "beables" (Bell 1987) are, for the method of abstraction, also observables as is, for that matter, the frequency with which an observable takes on its values. The latter might be regarded as unachievable in practice since any finite number of readings can achieve only an approximation to it. But that need be of no concern to us. Our only requirement is that an observable be well-typed. When desired, the stricter "observation as measurement" from quantum mechanics can be modelled as a certain kind of observation in the sense introduced in this paper: the change in behaviour associated with an "observation as measurement" event is simply specified to conform to the uncertainty principle. The same holds for the constraint of quantum mechanics that only certain (i.e. commuting) observables may be measured simultaneously: whilst two events, say A and B, may be observed independently, their simultaneous observation constitutes a third event, AB say, with the different behavioural consequences dictated by quantum mechanics.

3.6 Decidable observation

In the theory of computation an observable is called *decidable*, or *effective*, if and only if its behaviour is given by a computable function. For example it is known to be undecidable whether or not a program terminates i.e. there is no algorithm for its determination (Boolos et al. 2002). No assumption about the decidability of an observable should be made, for the following reason. The field of Formal Methods within Computer Science (Hoare and He 1998) concerns itself with the mathematical specification and development of information systems. Typically a specification embodies a twofold constraint: the required program must conform to such-and-such a functional specification *and terminate*. Without the last conjunct, undesirable programs, which execute forever, never yielding a result, might be allowed (in

some models of computation). But such a specification is no more than a behaviour phrased in terms of observables for input, output (appearing in the functional specification) and termination (appearing in the second conjunct). And we have just seen that termination cannot be assumed to be decidable. The consequence of allowing an observable to be undecidable is that some ingenuity is required to prove that an implementation meets a specification phrased in terms of its observables: no program can possibly achieve that task in general.

5. Conclusion

In this paper, I have summarised the method of abstraction and applied it to the study, modelling and analysis of phenomenological and conceptual systems. I have tried to show its principal features and main advantages. The method clarifies implicit assumptions, facilitates comparisons, enhances rigour and hence promotes the resolution of possible conceptual confusions. I have argued that, for discrete systems, whose observables take on only finitely-many values, the method is indispensable. Its limitations, on the other hand, are those of any typed theory. Use of LoAs is effective in precisely those situations where a typed theory would be effective (where it is possible or desirable). Introduction of LoAs is often an important step prior to mathematical modelling of the phenomenon under consideration. But even when that further step is not taken, introduction of LoA remains a crucial tool in conceptual analysis. Of course, care must be exercised in type-free systems, where the use of LoA may be problematic. Such systems are susceptible to the usual paradoxes and hence to inconsistency. However, I have also shown that, if carefully applied, the method confers remarkable advantages in terms of consistency and clarity. Too often philosophical debates seem to be caused by a misconception of the LoA at which the questions should be addressed. This is not to say that a simplistic policy of "on the one hand... and on other hand" sort of arguments would represent a panacea. Disagreement is often not based on confusion. But it seems that chances of resolving or overcoming it may be enhanced if one is first of all careful about specifying what sort of observables are at stake.

Acknowledgements

Many of the ideas presented here were developed and formulated in close collaboration with Jeff Sanders, who should really be considered a co-author of this paper, although not co-responsible for its potential shortcomings. I owe to Jeff most of my understanding of the more technical aspects of the LoA method. *Levellism* has been common currency in science for a long time, but only recently has the concept of simulation been used in computer science to

relate levels of abstraction to satisfy the requirement that systems constructed in levels (in order to tame their complexity) function correctly (see for example De Roever and Engelhardt 1998, Hoare and He 1998). The definition of *Gradient of Abstraction* in this paper has been inspired by this approach. Indeed, I learnt from Jeff to take as a definition the property established by simulations, namely the conformity of behaviour between levels of abstraction. Tim Colburn kindly allowed me to rely on his research (Colburn and Shute 2007) before it was published. Finally, I am grateful to Mark Bishop for the kind invitation to participate in the AISB Symposium, parallel to the Loebner Prize contest in 2008, and for the opportunity to receive valuable feedback during and after the talk.

References

Barwise, J., and Etchemendy, J. 1987, *The Liar: An Essay on Truth and Circularity* (Oxford: Oxford University Press).

Bell, J. S. 1987, "The Theory of Local Beables" in *Speakable and Unspeakable in Quantum Mechanics – Collected Papers on Quantum Mechanics* (Cambridge: Cambridge University Press), 52-62.

Boden, M. A. (ed.) 1996, *The Philosophy of Artificial Life* (Oxford: Oxford University Press).

Boolos, G., Burgess, J. P., and Jeffrey, R. C. 2002, *Computability and Logic*, 4th ed. (Cambridge: Cambridge University Press).

Colburn, T., and Shute, G. 2007, "Abstraction in Computer Science", *Minds and Machines*, 17(2), 169-184.

Floridi, L. submitted, "Levellism and the Method of Abstraction".

Floridi, L., and Sanders, J. W. 2004, "On the Morality of Artificial Agents", *Minds and Machines*, 14(3), 349-379.

Gell-Mann, M. 1994, *The Quark and the Jaguar: Adventures in the Simple and the Complex* (London: Little Brown).

Hayes, I., and Flinn, B. 1993, *Specification Case Studies*, 2nd ed. (New York – London: Prentice Hall).

Hendriks-Jansen, H. 1989, "In Praise of Interactive Emergence: Or Why Explanations Don't Have to Wait for Implementations" in Langton [1989].

Hoare, C. A. R., and He, J. 1998, *Unifying Theories of Programming* (New York – London: Prentice Hall).

Hughes, P., and Brecht, G. 1976, *Vicious Circles and Infinity: A Panoply of Paradoxes* (London: Cape).

Knuth, D. E. 1997, *The Art of Computer Programming*, 3rd ed. (Reading, Mass.: Addison-Wesley).

Langton, C. G. (ed.) 1989, *Artificial Life: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*. Santa Fe Institute Studies in the Sciences of Complexity, Proceedings 6 (Redwood City, CA: Addison-Wesley).

Langton, C. G., Taylor C., Farmer J. D. and Rasmussen S., (eds.) 1992, *Artificial Life II. Santa Fe Institute Studies in the Sciences of Complexity*, Proceedings 10 (Redwood City, CA: Addison-Wesley).

Langton, C. G. 1996, *Artificial Life,* in Boden [1996], 39-94, updated version of the original which appeared in Langton *et al.* [1992].

Mitchell, T. M. 1997, *Machine Learning* International edition (New York – London: McGraw-Hill).

Moor, J. H. (ed.) 2001, *The Turing Test: Past, Present and Future*, special issue of *Minds and Machines*, 11.1.

Murray, J. D. 2003, *Mathematical Biology*, 3rd ed. (New York – London: Springer).

Robinson, J. 1989, *Vintage Timecharts: The Pedigree and Performance of Fine Wines to the Year 2000* (London: Mitchell Beazley).

Russell, B. 1902, "Letter to Frege" in J. van Heijenoort (ed.), *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931* (Cambridge, MA: Harvard University Press, 1967), 124–125.

Spivey, J. M. 1992, *The Z Notation: A Reference Manual*, 2nd ed. (New York – London: Prentice-Hall).

Thompson, D. A. W. 1992, *On Growth and Form*, rev. ed. (New York: Dover Publications Inc).

Turing, A. M. 1950, "Computing Machinery and Intelligence", *Minds and Machines*, 59, 433-460.