The cognitive value of abstractions

Kalin Penev

02.05.2001

1. Introduction.

In traditional logic, the process of deriving a universal from particulars is an abstraction [Boruch, 1967]. This essay discusses the notion of abstraction, definitions, classifications, and levels. It points to the cognitive value of abstraction and to conditions for a good abstraction. The essay considers also a temporal abstraction and its cognitive value.

2. Abstraction.

2.1. Definitions

There exist many definitions of abstraction. For a long time, many authors have tried to explain this notion, and its role in cognition. They present a different points of view. Earlier definitions oppose abstractive cognition to intuitive cognition. Now in line of the development of new software models, abstraction and it definition are a topical problem. The definitions below are an example.

"Intuitive cognition is defined as an act of apprehension in virtue of which the intellect can evidently judge that the apprehended object exists or does not exist, or that it has or does not have some particular quality or other condition; in short, an intuitive cognition is an act of immediate awareness in virtue of which an evident judgement of contingent fact can be made.

Abstractive cognition is defined as any act of cognition in virtue of which it cannot be evidently known whether the apprehended object exists or does not exist, and in virtue of which an evident contingent judgement cannot be made." William of Ockham (1285-1349) [Ernest, 1967]

"Knowledge of the external world can be obtained either by intuition or by abstraction. ... By abstraction, one obtains only an indistinct and confused image of a thing. Tommaso Campanella (1568-1639) [Bernardine, 1976]

"The essence of abstraction is to extract essential properties while omitting inessential details." [Ross et al, 1975]

"A view of a problem that extracts the essential information relevant to a particular purpose and ignores the remainder of the information." [IEEE, 1983]

"Abstraction is a process whereby we identify the important aspects of a phenomenon and ignore its details." [Ghezzi et al, 1991]

"Abstraction is generally defined as 'the process of formulating generalised concepts by extracting common qualities from specific examples.'" [Blair et al, 1991]

"Abstraction is the selective examination of certain aspects of a problem. The goal of abstraction is to isolate those aspects that are important for some purpose and suppress those aspects that are unimportant." [Rumbaugh et al, 1991]

"The meaning [of abstraction] given by the Oxford English Dictionary (OED) closest to the meaning intended here is 'The act of separating in thought'. A better definition might be 'Representing the essential features of something without including background or inessential detail.'" [Graham, 1991]

"[A] simplified description, or specification, of a system that emphasises some of the system's details or properties while suppressing others. A good abstraction is one that emphasises details that are significant to the reader or user and suppress details that are, at least for the moment, immaterial or diversionary." [Shaw, 1984]

"An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of object and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer." [Booch, 1991]

"Abstraction is one of the fundamental ways that we as humans cope with complexity." [Miksch *at al.,* 1996]

From these definitions it is clear that abstraction is a way to mastering complexity. In the context of the modern Computer Science abstraction is a technique to cope with large complex data and operations using (building) adequate models.

2.2. Classification of abstraction.

Definitions for abstraction, consider this term from two points of view: one is abstraction as a process, and the other is abstraction as an entity. Abstraction, as a process, points at the extracting of the essential details, while ignoring the inessential details. [Miksch *at al.,* 1996], [Ross et al, 1975], [Ghezzi et al, 1991], [Rumbaugh et al, 1991], and [Graham, 1991] all appear to view abstraction as a process. Abstraction, as an entity, points at a view, a model, or other representations of the essential details. [Booch, 1991] and [Shaw, 1984] describe abstraction as an entity. Both views are equally valid, and necessary [Berard E, 1995].

Abstraction can be classified also by its subject. Some authors recognise different types of abstraction such as functional abstraction, data abstraction, process abstraction, event abstraction, and object abstraction [Park 1991]. Each of the above definitions, because they are general definitions of abstraction, avoids describing which specific categories of information are emphasised or de-emphasised.

3. The cognitive value of abstraction.

From the definitions given in &2 is clear that abstraction has an important role in understanding and learning of a complex entities or in the operations with a large multitudes. This is valid also for any entity, because any entity has an infinite number of characteristics and relations. In fact, the way of thinking is based on abstractions. From computer modelling point of view any model can be consider as an abstraction. Understanding and modelling of the entities from reality define direct cognitive value of abstraction. Cognition (and consequently abstractions) has major influence on thinking, and on behaviour. Let us consider abstraction as "The act of separating in thought" [Graham, 1991]. Obviously abstractions closely relate to the thinking. Ones an abstraction is made, about one entity, the abstraction become base for thinking about this entity. This defined our way of thinking, behaviour, actions, and respectively changes, which we make over this or other entities. Let us consider abstraction in term of computer modelling. In this case the abstraction, which we consider as a model about an entity, is a base for software operations, computer systems behaviour, and action. This is also indirect but important value of abstraction. Change of abstractions leads to change of thinking. Change of computer models changes the way of operations and results. It depends on the abstraction (model), which we have about some entity, we can understand and change it in different ways. For example if we have object based model about Traffic System we can produce one type of results. But if we change the model to field based model we can have different, possibly better, results. And if we have two models, instead of one, we will have better results. Also we can compare, which is better. Therefor we can consider the cognitive value of abstraction first directly in understanding of a complex entity and then indirectly, as utilise this cognition, in way of thinking, which we have, using this abstractions. (From computer modelling point of view we can consider the cognitive value of abstraction first directly in modelling of complex entity and then indirectly, as utilise this models, in way of operation, which computer systems have, using this model.) The indirect cognitive value of abstractions is also in the changes which we make on the reality it depends on the cognition which we have.

3.1. Characteristics of an abstraction.

There can exist many abstractions of the one entity or of one multitude, each abstraction should have certain characteristics that distinguish it as applicable, understandable, and useful [Kafura 1998]. A good abstraction has a **well specified purpose**. The purpose reflects the nature of the abstraction. Whether the purpose is well defined depends on the community of people who will use the abstraction. In some cases the purpose might be very specific in an application domain, and understandable for the group of people in that application area but may mean little to other groups. In other cases, abstractions for widely known purposes may be useful for the general population. A good abstraction is **coherent**. The abstraction should contain a related set of details that makes sense from the viewpoint of the modeller. The details must be what is needed and expected in a given setting. A meaningful abstraction is **accurate**. The abstraction should only contain details that are displayed by the entity being modelled. The abstraction should not be endowed with powers and abilities far beyond those of the actual entity. But there are special situations under which this principle may be relaxed. A valuable abstraction is **minimal**. The abstraction should not contain details extraneous to the purpose for which it is defined. A useful abstraction is **complete** [Kafura 1998]. The abstraction should contain all of the details necessary to manipulate the abstraction for its intended purpose. In conclusion we should underline that these characteristic are very subjective in nature, implying that the ability to form good abstractions requires good judgement, practice and experience.

3.2. Level of abstraction.

An abstraction has a good cognitive value when it contains minimal and sufficient essential details. To obtain such optimal level depends: on the size of the initial multitude; on the criteria of what is essential, and what is unessential; and on the community who will use the abstraction. The applicable level between essential and unessential, must be acceptable for all users of the abstraction. We can have varying levels of abstraction. [Berard E, 1995] As we move to higher levels of abstraction, we focus on the larger and more important pieces of information (using our chosen selection of criteria). Another common observation is that as we move to higher levels of abstraction, we tend to concern ourselves with progressively smaller volumes of information, and fewer overall items. As we move to lower levels of abstraction, we reveal more detail, typically encounter more individual items, and increase the volume of information with, which we must deal.

3.3. Conditions for good cognitive value of abstractions.

Assessment of the cognitive value of abstraction has a subjective and a qualitative nature. To decrease subjectivism in modelling of abstractions is necessary to create preliminary, appropriate conditions. An abstraction has good cognitive value when:

-it has a clearly defined purpose. If the purpose is not a clearly defined the cognitive value of abstraction is low.
-the multitude of entities, subject of abstraction, is well defined. If the multitude of entities is not clearly defined abstraction is not understandable for users which operate with different multitude of entities.
- what is essential and what is unessential is clearly defined. If this is not well defined the results can contain some unessential details instead essential ones, which means low cognitive value.
-the rules for extraction, of essential and important details, are clearly defined. If the rules are not clear, abstraction can disqualify important details. It means that a default rules decrease the cognitive value of abstraction. For example in algebra (A - B) - C = (A - C) - B , but for abstraction this formula is not valid.
-the presentation of the abstraction has well defined format acceptable and understandable for users of abstraction. If presentation is not understandable the cognitive value of abstraction is low.

An abstraction has good cognitive value when all these condition are preliminary defined. In terms of computer modelling any model is a kind of abstraction. A model has good cognitive value when it allows and supports mentioned conditions. In this case the model, as an abstraction, has cognitive value.

4. Temporal abstraction.

Temporal entities are a specific area for abstractions. Temporal entities change over time. Changes can affect many or few details. Changes can be continuous or periodic, essential or unessential. In this case an essential entity can by subject of temporal abstraction. If this essential entity has not essential changes during the defined period of time then this unessential period can be ignored or hidden. For a long period of time some changes of essential details can become unessential because are old, in this case they are subject of abstraction. It is clear that temporal abstraction need specific approach of abstraction different of abstraction of static entities, which has characteristic with constant value. The value of temporal abstraction is in memorising of the temporal entities. In this case temporal abstractions has a higher cognitive value than non temporal abstraction.

5. Conclusion

Abstraction is a technique that helps to identify which specific information should be visible and which information should be hidden. This help in understanding and studding of the complex problems, and operations with large complex data, difficult for direct analysis. This defines the cognitive value of abstraction.

References:

Bernardine M. Bonansea, 1967 Campanella, Tommaso, Encyclopedia of Philosophy, Macmillan, New York, 1967, Vol. 2, p. 12.

Berard E, 1995, Abstraction, Encapsulation, and Information Hiding, The Object Agency. 1995.

Blair G., 1991, Blair, J. Gallagher, D. Hutchison, and D. Sheperd, Object-Oriented Languages, Systems and Applications, Halsted Press, New York, New York, 1991.

Boruch A. Brody, 1967, Logical Terms, Glossary of, Encyclopedia of Philosophy, Macmillan, New York, 1967, Vol. 5, p. 57.

Booch G., 1991, Object-Oriented Design With Applications, Benjamin/Cummings, Menlo Park, California, 1991.

Budd T., 1991, An Introduction to Object-Oriented Programming, Addison-Wesley, Reading, Massachusetts, 1991.

Dijkstra E.W. Dijkstra, 1968, "Structure of the 'THE'-Multiprogramming System," Communications of the ACM, Vol. 11, No. 5, May 1968, pp. 341-346.

Ellis M.A., and B. Stroustrup, 1990, The Annotated C++ Reference Manual, Addison-Wesley, Reading, Massachusetts, 1990.

Ernest A. Moody, 1967, William of Ockham, Encyclopedia of Philosophy, Macmillan, New York, 1967, Vol. 8, p. 308.

Ghezzi C., 1991, M. Jazayeri, and D. Mandrioli, Fundamentals of Software Engineering, Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

Goldberg and Robson, 1983]. A. Goldberg and D. Robson, Smalltalk-80: The Language and Its Implementation, Addison-Wesley, Reading, Massachusetts, 1983.

Graham I., 1991, Object-Oriented Methods, Addison-Wesley, Reading, Massachusetts, 1991.

Guttag J., 1977, "Abstract Data Types and the Development of Data Structures,"

IEEE, 1983, IEEE Standard Glossary of Software Engineering Terminology, The Institute of Electrical and Electronic Engineers, New York,New York, 1983.

Kafura D., 1998, OBJECT-ORIENTED SOFTWARE DESIGN and CONSTRUCTION with C++, Prentice-Hall, Inc., A Simon & Schuster Company, Upper Saddle River, New Jersey 07458, ©1998.

Kuhn T.S., 1962, The Structure of Scientific Revolutions, University of Chicago Press, Chicago, Illinois, 1962.

Liskov B., and J. Guttag, 1986, Abstraction and Specification in Program Development, The MIT Press, Cambridge, Massachusetts, 1986.

Miksch S., Horn W., Popow C., Paky F., 1996, Context-Sensitive and Expectation-Guided Temporal Abstraction of High-Frequency Data, Proc. Of the Tenth International Workshop on Qualitative Reasoning, Stanford Sierra Camp, Fallen Leaf Lake, CA, May 21-24, 1996.

Miksch, S.; Horn, W.; Popow, C.; Paky, F.: Utilizing Temporal Data Abstraction for Data Validation and Therapy Planning for Artificially Ventilated Newborn Infants, Artificial Intelligence in Medicine, 8(6), pp. 543-576, 1996.

Mish F.C., 1988, Editor in Chief, Webster's Ninth New Collegiate Dictionary, Merriam-Webster Inc., Springfield, Massachusetts, 1988.

Myers G.J. Myers, 1978, Composite/Structured Design, Van Nostrand Reinhold, New York, New York, 1978.

Park. H.-S. Park, 1991, "Abstract Object Types = Abstract Knowledge Types + Abstract Data Types + Abstract Connector Types," Journal of Object-Oriented Programming, Vol. 4, No. 3, June 1991, pp. 37 - 39, 42 - 44, 46 - 48, 51 - 52.

Ross D.T. Ross, J.B. Goodenough, and C.A. Irvine, 1975, "Software Engineering: Process, Principles, and Goals," IEEE Computer, Vol. 8, No. 5, May 1975, pp. 17 - 27.

Rumbaugh J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, 1991, Object-Oriented Modeling and Design, Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

Shaw M. Shaw, 1984, "Abstraction Techniques in Modern Programming Languages," IEEE Software, Vol. 1, No. 4, October 1984, pp. 10 - 26.