Practical Theory and Theory-Based Practice

Gerd Wagner

Institut für Informatik, Universität Leipzig,
Augustusplatz 10-11, 04109 Leipzig, Germany

gw@inf.fu-berlin.de

Abstract. In computer science, the relationship between theory and practice seems to be uneasy in comparison to other sciences, such as, e.g., chemistry or electrical engineering. This may be partly created by the strong influence of mathematicians (in theoretical computer science) who don't care much about real systems, but also by the obvious success of ingenious programmers and infamous hackers who don't care much about theory. It may indicate that computer science is 'very special', but it is more likely just a sign of its immaturity.

1 Do the Real Thing

Why should we try to develop a theoretical basis for agent systems rather than immediately develop agent systems themselves on our favorite platforms ? Programmers tend to be impatient. They don't want to wait for suitable theories. They prefer to do the real thing right away.[1] And it is exactly this pioneering behavior which has led to many new ideas and techniques in computer science. But in order to get a deeper understanding of a new idea, and in order to make further progress in its development, it is essential to establish formal concepts and methods whose properties can be mathematically analyzed. Only formal concepts can serve as a non-ambique and platform-independent reference framework for comparisons and further extensions which are necessary for any real progress. To see this, let's take a brief look at the history of two other fields of computer science: the success story of database technology, and the still-no-success (= failure?) story of expert systems.

2 The Success of Database Systems

In the sixties and seventies, pushed by the need to store and process large data collections, powerful database software based on the file system technology available at that time has been developed. These types of systems have been named *hierarchical* and *network* databases, refering to the respective type of file organization. Although these systems were able to process large amounts of data efficiently, their limitations in terms of flexibility and ease of use were severe. Those difficulties were caused by the rather low-level data operations of hierarchical and network databases dictated by their respective

[1] Only Dijkstra's theoretically motivated insistance that GOTO statements have to be considered harmful has eventually led to more programming discipline in the software industry. Dijkstra also reports (in [1]): *The programmers didn't like the idea [of verification] at all because it deprived them of the intellectual excitement of not quite understanding what they were doing.*

file organization. Thus, both database models have later on turned out to be cognitively inadequate. Only through the formal conceptualization of *relational databases* by Codd in the early seventies could the inadequacy of the then prevailing database technology be overcome. Only the logic-based formal concepts of the relational database model have led to more cognitive adequacy, and have thus constituted the conceptual basis for further progress (towards deductive, OO, temporal, etc. databases). Unlike much of theoretical computer science, Codd's theory of relational databases is an example of a 'practical theory'.

3 The Difficulties of Expert Systems

In the field of expert systems, on the other hand, there has never been a conceptual breakthrough like that of the relational database model. It seems that the field is still at the 'pre-conceptual' level of hierarchical and network databases, and there seems to be almost no measurable progress since MYCIN. There is neither a formal definition of an 'expert system', nor is there any clear semantics of expert system rules. The field has rather developed a variety of notions (such as 'production rules,' or 'certainty factors') which have often not been clearly defined, or have not been based on a logical semantics. Lacking a clear (and formally defined) conceptual framework, it is difficult for the expert system community to compare different systems and to identify shortcomings and measure progress. One may argue that these problems are due to the inherent difficulties of knowledge representation and processing, but it rather seems that the expert system community has failed to develop cognitively adequate formal concepts and has even failed to learn from its neighbor fields of logic programming and deductive databases which have successfully devolped a clear concept of rules based on a logical semantics.

So, the question for the field of multiagent systems really is: does it learn its lesson from the database and expert system stories, or is it going do repeat the mistakes of the expert system field ?

4 From Agent Theory to Agent Construction

There is much theoretical work in AI and MAS lacking any serious reference to practical problems, and there is much application-oriented work lacking any serious reference to theoretical concepts. To me, these papers not attempting to combine theory and practice are often not very interesting. It is therefore a pleasure to comment on a paper by Michael Luck, Nathan Griffiths and Mark d'Inverno which proposes a way *From Agent Theory to Agent Construction*. However, I am not sure whether the paper keeps the promise of its title, because

1. The theory used is **very limited**:
   - goals are just flags associated with a behavior,
   - the state of the environment is represented by simple attribute/value sentences, excluding any form of incompleteness and uncertainty (or 'noise'),
   - actions are not associated with preconditions and effects,

   - agents react without memory.
2. Many ingredients of the proposed *Agent Simulation Environment* are **not explicitly defined**, but rather left implicit:
   - the perception mechanism,
   - the behavior control (by means of production rules),
   - the entire execution model.
3. The issue of **information and knowledge processing is neglected**. Which types of information may occur in which agent domains ? How is incomplete, uncertain, temporal, confidential, unreliable, etc., information processed ?

My impression is that the authors' choice of the Z specification language has rather led to premature and inadequate stipulations, such as *'an agent is an object with an associated set of goals'*, and not to generic conceptualizations. Unfortunately, the formal character of Z is of no help in the modeling of agents because Z does not support any of the high-level functionality needed in agent systems (such as knowledge-based inference, knowledge assimilation, connecting perceptions to beliefs and reactions, planning, plan execution, communication, etc.)

5 Question: What is an Agent ? Answer: What is a Number ?

We should not attempt to define what is an agent in general. This is not necessary, and probably even impossible, as there is no definition of *what is a number* in mathematics, but only definitions of specific kinds of numbers capturing important cases, such as natural or rational numbers. The same applies to databases: there is no formal definition of what is a database in general, but only of specific kinds of databases, such as relational or deductive databases.

While we can certainly not find a generic definition of *the agent*, we should look what are the important cases of agent types to be captured by precise mathematical definitions. Such a conceptualization can only be successfull if it is based on a sufficiently rich collection of practical experience.

References

1. D. Shasha and C. Lazere: *Out of Their Minds*, Copernicus, New York, 1995.