From Modeling with Words to Computing with Numbers

Jose Mira*, Ana E. Delgado*
*Dpto. de Inteligencia Artificial, ETSI Informática, UNED. Madrid, Spain
Email: {jmira,adelgado}@dia.uned.es

Abstract— **Many of the criticisms received by soft-computing and computational intelligence (fuzzy systems, artificial neural networks, and neurofuzzy networks) came from the lack of distinction between entities of the initial conceptual model and entities of the corresponding formal representation of this model.**

To contribute to the dissolution of some of these criticisms we use in this paper the methodological frame of three levels and two domains in each level for the description of a calculus. This frame enables us to establish a crystal-clear distinction between (1) the use of natural language words in the construction of the conceptual model of a calculus and (2) the formalization of these structures (nouns, verbs, and conditionals) using abstract symbols and logical and/or mathematical formal rules. This distinction is made for the current paradigms in Artificial Intelligence (symbolic, situated, connectionist and hybrid).

Index Terms— **Computing with words, knowledge modeling, formal description languages**

I. Introduction and problem statement

A certain part of the Artificial Intelligence (*AI*) community attempt to interpret and use some natural language words as effective computational elements assuming that words have the same meaning and functionalities in humans as in computer programs and hardware. This attitude does not take under consideration the deep change of meaning when moving from natural language to formal languages, as consequence of the primary and constitutive differences between human beings and digital electronics.

Many of the criticisms received by soft-computing and computational intelligence (fuzzy systems, artificial neural networks, and neurofuzzy networks) came from the lack of distinction between entities of the conceptual models and entities of the corresponding formal models and programs, as well as from the subsequent optimistic interpretations of the supposed functionalities of these programs based on the initial meaning of the words used in the informal specification. The reason for this misunderstanding is that in the initial descriptions using concepts, nouns and verbs of natural language, the proper semantics of entities of this language are mixed with that of the abstract entities and formal rules constitutive of the formal model. A clear distinction appears if we use explicit tables of correspondence when we rewrite the conceptual models in terms of logic and mathematics. In the formal models previous to programs there are no purposes, no goals, no perceptions, no meaning, no concepts, no natural language words. There are only abstract symbols and formal rules expressing how these symbols have to be combined in order to create new symbols.

We propose in this paper to replace "*computing with words*" [10] by "*modeling with words*" and "*computing with numbers*". This, more likely, represents reality, and do not confuse the effective functionalities of words with that of abstract symbols. The use of words in conceptual modeling of human knowledge concerning Problem Solving Methods (*PSMs*) has the sufficient computational usefulness not to require the addition of other functionalities that they don't possess.

To help in the understanding of the effects of this change of meaning when moving from natural language to programming language we bring to mind the clear distinction between the three basic steps in all computation:

1) Conceptual modeling of the theory of the calculus (analysis phase)
2) Formal rewriting of the conceptual model (formalization phase)
3) Programming of the formal model (implementation phase)

In the first step we usually use a semantically limited version of natural language (nouns for concepts, verbs for inferences, conditional statements for control rules) to describe the procedure used by humans in the solution of a specific problem (the *PSM*).

In the second step we have to rewrite all the entities and relations used in the construction of the conceptual model (all the words), in terms of formal entities and operators with clear logical and/or mathematical meaning. In the third step we rewrite the formal model in terms of the set of entities constitutive of a programming language. The initial natural language meaning of these words always remains in the domain of the external observer and at the knowledge level, outside the formal model and also outside the computer program. The original meanings of the words play any causal role neither in the formal model nor in the program. In fact, both, model and program, can serve to accommodate many other different natural language descriptions (many other conceptual models).

Keeping this perspective in mind we dedicate section two to the description of the building where the knowledge involved in a calculus resides. In section three we summarize the different proposals on the use of words in knowledge modeling. We conclude stating the necessity of a clear distinction between the modeling and formalization stages in the development of a calculus.

II. The building of and for knowledge

To help us in the establishment of the distinctions between the different components of the human knowledge that take part in the description of a calculus we introduce the "building" of knowledge [5]. This building has three levels and two domains in each level; that is to say we distinguish six types of knowledge involved in a calculus.

The three levels ("storey") of this building are the physical level (*PL*) where lives the machine hardware, the symbol level (*SL*) where the programs live and the third level, introduced by Newell [6] and Marr [4], located over the symbol level and named by Newell "the knowledge level" (*KL*) and by Marr the level of "the theory of calculus" (figure 1 in next page).

To know the physical machine and the software components of a calculus is not enough to obtain a complete description of all the knowledge involved in this calculus. We need also to know the purpose of the calculus and the *conceptual and formal models* to describe the procedure (*PSM*) used by the human expert (i.e. medical doctor) to solve this calculus (i.e. to diagnose). The conceptual model is constructed at the knowledge level, using words with the meaning usual for each domain of expertise. The formal model is also constructed at the knowledge level, but using only abstract entities and rules.

In order to understand the differences between the two types of knowledge involved in each level, we make a new distinction between two domains of description: the levels *own domain* (*OD*) and the *domain of the external observer* (*EOD*) which first build-up the different tables of correspondence between words and abstract symbols and later interprets the results of computation using the same tables.

The knowledge of the *OD* is concerned with the constitutive elements and the intrinsic causality law of the level ("apartments" to the right in figure 1). Here things happen as "they have to happen" (inverters do invert and counters do count at the *PL*, branching instructions branch at the *SL*, and formal models behave as logic and mathematics dictate at the *KL*).

On the other part ("apartments" to the left in figure 1) we have the *EOD*, where the models-maker and the programmer resides and uses words with the "right" meaning in natural language and in the metalanguage of each domain of expertise, without the semantic and syntactic restrictions imposed by the formal and programming languages. Natural language words mean more that we can tell to a computer. Entities of the *OD* (numbers) do not share the meanings of the corresponding entities in the *EOD* (words) because the constituent elements of the *EOD* do not belong to the repertory of elements constitutive of the *OD*, for the same level.

A simple example can help us to understand the need of these three storey building. What is fever for a medical doctor it is temperature for a physician, a formal variable for a computer programmer and a set of electrical potentials for an electronic engineer. It is clear that fever and temperature reside in third floor, the programming variables in the second and the electrical potentials in the first floor. For this simple diagnostic rule we have:

- Natural language rule (at the *EOD* of *KL*):

```
When the body temperature is high,
then the diagnostic must be fever.
```

- Formal language rule (at the *OD* of *KL*):

```
IF Body_temperature = 'High'
THEN Diagnostic = 'Fever'
```

Observe that now `Body_temperature`, `'High'`, `Diagnostic` and `'Fever'` are only frozen labels for the use of the external observer but without any causal connection within the formal model that can be rewrited using only the abstract symbols (`x`, `'y'`, `z`, `'u'`) respectively

```
IF x = 'y' THEN z = 'u'
```

We now move from the formal model (*OD* of *KL*) to the symbol level (*OD* of *SL*) using a table of correspondence with the statements of a high level programming language as shown in Algorithm 1 (in next page).

Finally, by compiling that program, we automatically obtain the assembler version for the same rule, moving from *OD* at the *SL* to *OD* at the *PL*.

III. Modeling with words at the *KL* and in the *EOD*

Computation involves conceptual modeling (3^{rd} left apartment) followed by formalization (3^{rd} right apartment) and then programming (2^{nd} right apartment) (figure 1). So, to build a computable model we start at the *KL* and in the *EOD* and this task has been always made using natural language from Wittgenstein [8,9] for whom language is a model of reality (a "map of reality"). Language functions in its uses and consists of a net of integrated activities governed by rules to which he calls "games of language". The meaning of each word resides in its use in the language, in the role or the function that this word subserves in the model of a calculus (words-concept, words-relation, words-inference, words-method or words-task). These rules of use of the words as components of a model of human knowledge don't happen overnight, rather appear historically, by consensus in each domain. Meanings are not private of some individual, but accepted for all the members of a scientific or technical community.

The use of a limited version of natural language in the construction of the theory of a calculus is accepted by all the current paradigms in *AI*. The differences appear in the set of concepts and relations used in the initial conceptualization and in the selection of the corresponding formal tools used to mirror the structure of the conceptual model. Both types of differences also depend on the style of modeling and on the balance between knowledge and data available for the formalization stage.

A. The symbolic paradigm

The current methodological approaches, like CommonKADS [7], distribute these components of knowledge into two layers:

1) Layer of task, methods and inferences
2) Layer of the domain knowledge

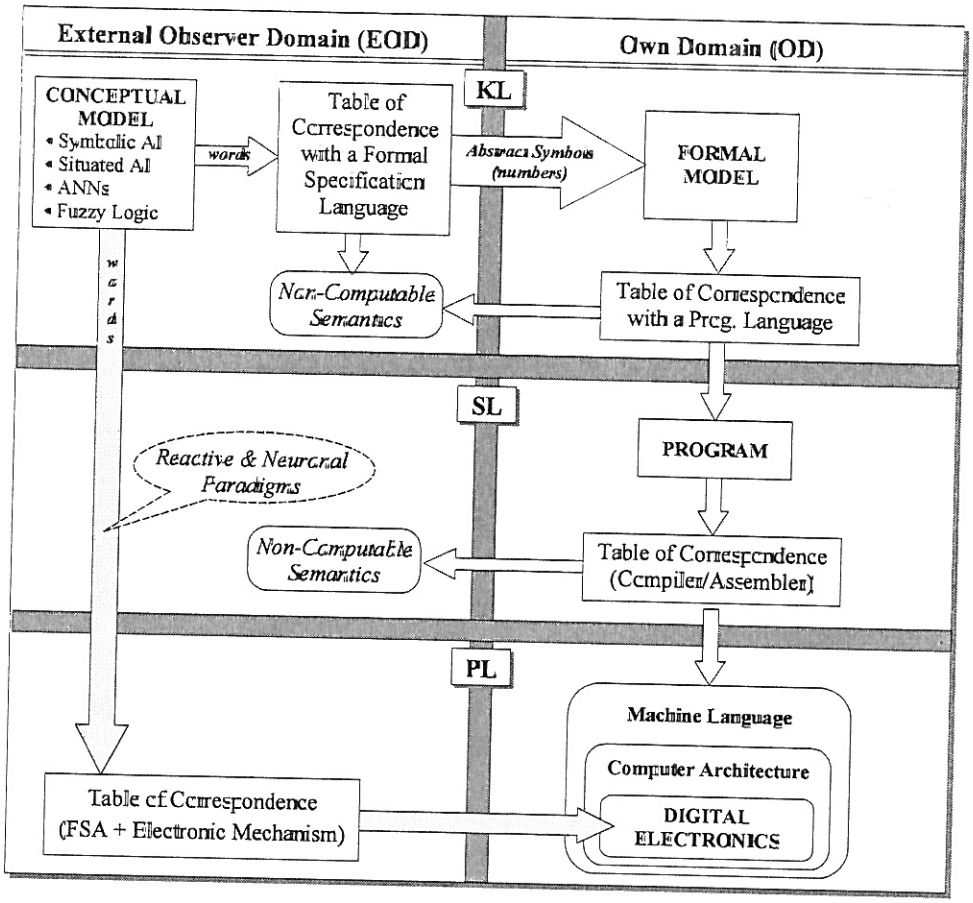

Fig. 1. The three levels of description of a calculus (knowledge, program and hardware), and the subsequent distinction between two different domains of description for each level. Arrows show the usual pathway in knowledge engineering for all the paradigms of AI. We start with a conceptual model (*KL-EOD*), then build-up a table of correspondence to produce the underlying formal model (*KL-OD*) and, finally we rewrite again the entities and relations of the formal model in terms of the statements of a programming language or of an electronic circuit.

Algorithm 1 Program in C language to implement the simple fever rule (comments are between /* and */ marks).

```
#include <stdio.h>          /* Use standard input/output libraries */
#define MAXTEMP 38          /* Constant for value of limit temperature (Celsius) */
main( ) {
  float Body_temperature;          /* Declare variable as floating point number */
  printf("\nPlease, introduce the body temperature in Celsius degrees: ");
  scanf("%f", &Body_temperature);                    /* Read value from keyboard */
  if (Body_temperature > MAXTEMP)                    /* RULE for temperature */
     printf("\nThe diagnostic is: Fever\n");        /* Positive case */
  else
     printf("\nThe diagnostic is: No Fever\n");     /* Negative case */
}
```

Task knowledge provides a description of which are the goals of the *system* (diagnosis, system configuration, learning, ...). *PSMs* knowledge specifies ways of decomposing this task up to the level of primitive inferences and *inference knowledge* describe the most basic reasoning steps (*cover, select, establish, abstract, match, refine*, ...) in which the selected *PSM* decompose the task. The description of the inferences is completed with the specification of the input and output roles played by the domain knowledge entities.

The *domain knowledge* layer models the set of entities and relations specific of a domain of application (medicine, economy, ...) which are independent of its use in inference. These structures are usually instantiated from an *ontology*.

We now have to construct the correspondence table for each of the entities of the domain layer and for each of the inferences of the inference layer. For example van Harmelen and Aben uses the $(ML)^2$ formal specification language, as shown in figure 2 for the "*abstract*" inference [3].

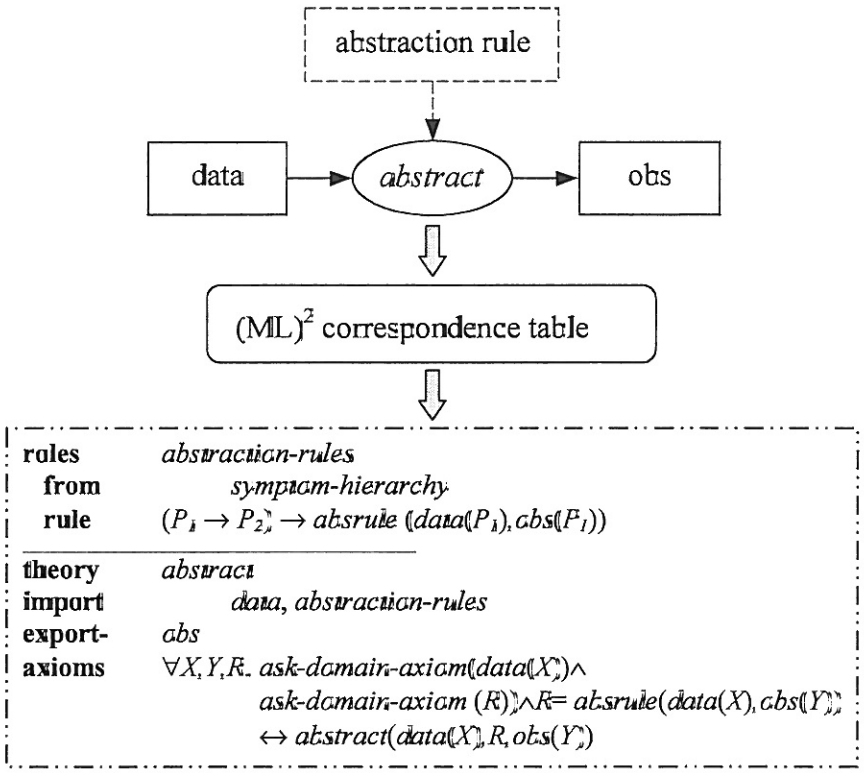

Fig. 2. Formalization of the inference "*abstract*" using the formal specification language $(ML)^2$ (adapted from [3])

B. The situated paradigm

The situated, reactive, embodied or behavior-based approach to *AI* proposed by Brooks [1] and Clancey [2], try to model the bottom-up. We start the conceptual modeling phase focusing on the observed behaviors of a system interacting with an environment (like in ethology) and instead of using cognitive entities that demand symbolic data-structures and inferences as primitives of our model, we use distinguishable *behaviors* such as "*obstacle avoidance*", or "*standing-up*" as components of our model. A behavior is defined as a direct mapping between a class of input patterns and a predefined pattern of motor actions. The behavior-based models have a layered architecture where all the layers (behaviors) run in parallel and there are some conflict resolution mechanisms of the type "winner takes all" where the winner is always the higher layer. Brooks call this style of modeling "subsumption architecture".

Differences with the symbolic approach to knowledge modeling appear at the formal specification phase. For most cases in the behavior-based approach we use *FSA* as formal language associating behaviors to states, input classes (behavioral releasers) to the *FSA* inputs alphabet, and output patterns of action to the *FSA* output alphabet. Observe that, again, knowledge modeling is made using natural language words ("*behaviors*", "*input patterns*", "*releasers*", "*output templates*", ...) and that these words can be put in clear correspondence with that used in the symbolic paradigm, as shown in table I.

TABLE I
SIMILARITIES BETWEEN THE SYMBOLIC AND THE SITUATED PARADIGMS

Symbolic	*Situated*
Inferences	Behaviors
Inferential Scheme	Comp. of Behavior (Subsumption)
I/O roles	Releasers/Templates
Formal Esp. Lang.	*FSA*

C. The connectionist paradigm

Until we arrive to the operationalization phase, the connectionist approach to knowledge modeling follows the same descriptive pathway that the symbolic one. We start with a natural language description of the *PSM* and the corresponding task decomposition in terms of sub-tasks and inferences. If we have enough declarative knowledge to operationalize all the inferences we don't need artificial neural nets (*ANNs*) to solve the problem. If we have more data than knowledge available and we need learning in real time, then *ANNs* is the proper option. In these cases the inference (or subtask) is modeled according to the scheme of figure 3, where the following characteristics are distinctive of connectionism:

1) The inferential scheme always links *numeric labelled lines* for the input role (*observables*) with *numeric labelled lines* for the output role (*classes*).
2) Between these two sets of *numeric lines* the formal specification of the *ANN* corresponds to a numeric parametric association.
3) The local function of each neuron is an analytical rule.

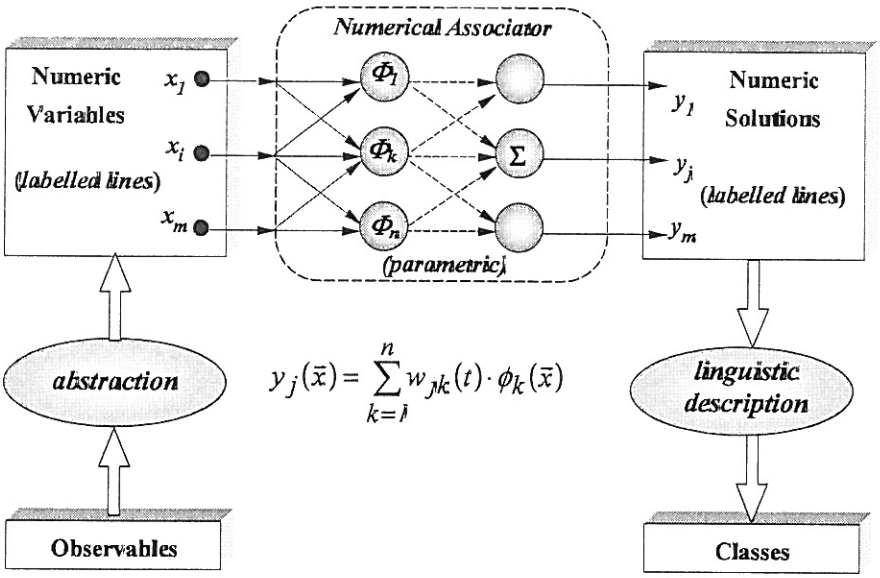

Fig. 3. The ANN modeling architecture

4) The first "hidden" layer expands the input space ($\vec{x}$)into a new functional basis, $\{\phi_j(\vec{x})\}$, and the output layer generates the labeled lines of the output space in terms of this new basis.

D. The fuzzy rules

Let us remember the two basic steps in the development of a calculus, both carried out at the *KL*: (1) Conceptual modeling in the *EOD* and (2) formal modeling into the *OD*. For the first phase all the natural language terms usually used in the context of softcomputing and computational intelligence is appropriated, in the sense that we are into the "3^{rd} left apartment", the house of modeling, the house of natural language in the building of knowledge. Here humans employ words, with the proper meaning in reasoning, when the available knowledge is *imprecise*. But computers don't have natural language compilers. Consequently, in the case of fuzzy rules, as in all the previously mentioned *AI* paradigms, it is *compulsory* to rewrite these rules in numerical terms, when moving into the formal model (*KL* into the *OD*) and before programming. In this transition (from *EOD* to *OD*) we leave out all the semantics that could justify the truthfulness of the statement "*computing with words*" (figure 1).

This means that the apparent computable character of fuzzy rules such as "`IF x is small THEN y is large`" disappears when we have to codify the words "`small`" and "`large`" in terms of its numeric membership functions (triangular, trapezoidal, Gaussian, ...). For example, in the trapezoidal case we have:

$$\mu(x,a,b,c,d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right)$$

This means that instead of codifying a number, we have to codify *four* numbers (*a,b,c,d*). And the same happens with the link between the antecedent and consequent parts of the rule, that now is a multiple "`IF`" (a "case-of" statement) with a formal specification in terms of a look-up-table (*LUT*) corresponding to the fuzzy procedure of inference (Zadeh, Mamdani, Sugeno-Tagaki, Tsukamoto, ...), still numeric.

if <condition A_1 in μ(x,a,b,c,d)> then <assign B_1 in μ(y,a,b,c,d)>
if <condition A_2 in μ(x,a,b,c,d)> then <assign B_2 in μ(y,a,b,c,d)>
... ...
if <condition A_n in μ(x,a,b,c,d)> then <assign B_n in μ(y,a,b,c,d)>

IV. Conclusions

As conclusion, we propose to replace "*computing with words*" by "*modeling with words*", and "*computing with numbers*". In this way we don't introduce unnecessary confusion and still retain all the computational usefulness of natural language in the initial specification of a calculus. For the next step we need to develop more expressive and powerful formal description languages.

References

[1] Brooks RA (1991) Intelligence without reason. MIT A.I. Memo Num. 1293
[2] Clancey WJ (1997) Situated cognition. On human knowledge and computer representation. Univ. Press, Cambridge
[3] Harmelen F van, Aben M (1966) Structure preserving specification languages for knowledge-based systems. Int. J. Human-Computer Studies 44:187212
[4] Marr D (1982) Vision. Freeman, New York
[5] Mira J, Delgado AE (2003) Where is knowledge in robotics? Some methodological Issues on symbolic and connectionist perspectives of AI. In: Zhou Ch, Maravall D, Da Rua (eds) Autonomous robotic systems. Physical-Verlag. Springer-Verlag. Berlin, pp 334
[6] Newell A (1981) The knowledge level. AI Magazine 120
[7] Schreiber et al. (1999) Engineering and Managing Knowledge: The CommonKADS Methodology. The MIT Press, Mass
[8] Wittgenstein L (2003) Investigaciones filosóficas. Ed. Crítica, Barcelona
[9] Wittgenstein L (2003) Tractatus logico-philosophicus. Ed. Tecnos, Madrid
[10] Zadeh LA (1999) From computing with numbers to computing with words–from manipulation of measurements to manipulation of perceptions. IEEE trans. Circuits and Systems 45:105119