Common Representations of Soft and Hard Declarative Knowledge

Wiesław Traczyk
National Institute of Telecommunications, Szachowa 1, 04-894 Warsaw, Poland
Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland
W.Traczyk@itl.waw.pl

ABSTRACT

Features *soft* and *hard* are often attached to such concepts as *computing, knowledge*, *systems, operational research* and the others, in order to stress different level of precision, certainty and completeness. In particular, Soft Systems Methodology and knowledge creation spirals are iterative processes, which steps should have different levels of hardness: from a very general and vague form in the first iteration (a beginning stage of knowledge creation) to more formal description in the last iteration (particularly if a model is to be processed by computer). The basic assumption of this approach is that, inserting certainty factors and concept qualifiers into conventional expressions of knowledge representation, we can control their level of hardness, preserving common representation for soft and hard knowledge.
Declarative knowledge is defined and its representations are divided into relation-based and mapping-based. The former are described by modified Description Language, and the latter by conventional logic with inserted factors.

Keywords: certainty factors, value qualifiers, fuzzy sets, rough sets, description logics.

1. INTRODUCTION

In the case of physical objects hardness is well defined and can be measured: by the level of compressibility (water, ball) or deformation (metals). Softness is usually understood as a low level of hardness (described as a continuum), and there is no precise border between *soft* and *hard* objects.
When features *soft* and *hard* are attached to more abstract concepts such as *computing, knowledge*, *systems, operational research* and the others, *soft* is commonly considered as opposition to *hard*, and handled by quite different methods.
Hard systems take the world as being systemic; systems exist and have a clear purpose, goals and well defined boundaries and solutions. We can define what success will look like prior to implementation of the solution. Hard systems analysis is largely devoid of human aspects and is useful in technical or relatively simple administrative or biophysical problems. *Soft systems* are characterized by structuring the problem situation rather then by problem solving (since there might be many problems which need to be solved), and "what" questions more than "how" questions. We know that things are not working the way we want them to and we want to find out why and see if there is anything we can do about it. Goals, objectives and even the interpretation of events are all problematic. Human perceptions, behaviour or activity seem to be dominating factors [1].
Hard knowledge is the part of what people know that can be articulated, formalized and stored, and is well structured. *Soft knowledge* is less quantifiable and can not be so easily captured and stored in conventional way. The social context and human aspect are very important. Examples of such knowledge might include tacit knowledge, experience, skills and cultural knowledge [2].

Soft computing includes a set of tools, used for handling problems which can not be solved by strict, algorithmic methods but permit approximate results. Applied values are vague, uncertain, fuzzy or rough. Linguistic descriptions frequently replace numeric data and some methods imitate the nature (neural nets, genetic algorithms,...).

Even from this short presentation one can observe similarities between features of "soft" concepts, and deduce that soft knowledge can describe soft systems, modeled and constructed with the help of soft computing. The need for more formal description of soft systems comes from Checkland's Soft Systems Methodology (SSM) map, containing important stages: root definitions and conceptual model. *A root definition* is expressed as a transformation process that takes some entity as input, changes or transforms that entity, and produces a new form of the entity as output. A *conceptual model* is a human activity model that strictly conforms to the root definition, using the minimum set of activities.
SSM is an iterative process so root definition and conceptual model should have different levels of vagueness: from a very general and vague form in the first iteration (a beginning stage of knowledge creation) to more formal description in the last iteration (particularly if a model is to be processed by computer). Entities should be defined, relations between them precisely described and actions clearly presented.
Very similar situation exists in *knowledge spirals* [3,4]. A *tacit knowledge* is initially very soft but externalisation makes it *explicit*, that means – more

codified and harder. After several circulations explicit knowledge may be quite hard and needs a formal description.
Knowledge related to systems and creation procedures should be therefore considered as a spectrum: at one extreme it is completely soft and at the other end it is hard. Very often initial knowledge is represented in natural language. In this stage the information elicited will be incomplete, and contain contradictions and ambiguities. Likely it is qualitative rather than quantitative. As our understanding of the system becomes clearer (during circulation though the spiral), the knowledge can be refined and represented more formally. It will be good to have such methods of knowledge representation which are able to present the whole spectrum in uniform manner.
The approach suggested here presents some modifications of existing knowledge representation languages, making them common for both soft and hard notions.

2. KNOWLEDGE DEFINITIONS AND VALIDATION

Description of knowledge representation should be started from the knowledge definition. Unfortunately, there is no generally accepted such definition. Most likely the cause of it lies in very big diversity of the notion. Knowledge (in its intuitive meaning) can be *descriptive* or *procedural, individual* or *collective, general* or *specific, tacit* or *explicit*, *structured* or *unstructured* and so on. The term *knowledge* suffers from a high degree of what might be called "terminological ambiguity" and often requires a lot of adjectives to make clear in what sense it is being used. Therefore it is essential to determine the need that the knowledge must fulfill.
If a range of interesting knowledge is limited to static descriptions and "how" questions, only declarative knowledge can be considered. In this case the following definitions seem to be appropriate.
Declarative knowledge is a collection of information and relations between information descriptions. Either information or relations are complex.
For example complex information and rather simple relations has knowledge (described sometimes by *frames*) relating to a university organization, but knowledge concerning a large family may have quite complex relations (and *nets* will be more suitable) .
Information is understood here (with compliance with C.E. Shannon) as everything that decreases uncertainty. More formally:

$DKNOWL = \langle INF, REL \rangle$

$INF = \{inf_1, inf_2, \ldots\}$,

$REL = \{rel_1, rel_2, \ldots\}, \quad rel \subseteq inf_i \times inf_j$

The symbol *inf* represents a set of information in the form of simple data, statements or expressions, and *rel* shows adequate relation.
For example the signature:

$SERVICE\text{-}KNOWLEDGE \subseteq TELEPHONE\text{-}OWNERS \times SERVICE\text{-}PARAMETERS$

describes clusters of telephone owners with common service products provided. Each component of the relation can be presented as another relation, e.g.

$TELEPHONE\text{-}OWNERS = \langle category, living\text{-}place, time\text{-}of\text{-}cooperation, \ldots \rangle$.

Instead of this *intensional* form one can use *extensional* one e.g.

⟨business, small-town, 3-year,... ⟩.

Important special case of the above relation, having properties of a function, has a form of the mapping:

$map: inf_S \rightarrow inf_D$

describing dependencies between the source information and information in demand.
Example of general signature may look as:

$DIAGNOSTIC\text{-}RULE: SYMPTOMS \rightarrow DIAGNOSIS$.

Practical representation tools for relational knowledge include *frames, semantic nets* and another graphical descriptions, jointly expressed by languages using Description Logics. Representations with mapping comprise *decision tables, trees, decision rules* or *association rules,* and appropriate languages use conventional first order logic or its subset.
All these languages and tools are open to some methods, making expressions and actions more "soft". Approach presented here shows such possibilities in the ordered form.

Effective way of stepwise transformation from soft to hard knowledge should take into account:
- substitution of conventional natural language statements by more structured expressions,
- reduction of imprecision of all values,
- decreasing uncertainty of all representation elements,
- elimination of vague decisions.

Reduction of imprecision can be achieved by changing a form of values (*qualifiers* α). In the case of parameters or attributes with numerical values, a spectrum of precision can have several steps (from more soft to more hard):
1. undetermined value in linguistic form:
   tax is small
2. linguistic variable defined as fuzzy set:

tax is *small* ≡ tax ∈ ⟨3,5,7,10⟩[1]

3. interval (sometimes in linguistic form):
   tax ∈ [5,6]
4. precise value (with a needed level of precision):
   tax = 5.

Quite soft (imprecise and linguistic) value "small" may be, in this way, transformed to quite hard value "5". General description of these steps is denoted as *V is* α or $V = \alpha$.

In the case of concepts which can be evaluated only linguistically, a level of hardness depends on the granularity of valuation. For instance, if weather is evaluated, the scale {bad, good} gives softer description than the scale {bad, nice, good, fine}. Denotation αV will be used for such valuations.

Measures (*factors*) of certainty (γ) can be presented and changed in the similar manner. Usually they are attached to statements or expressions (E), giving the form γE (E is certain with a grade γ). Numerical values of γ are usually taken from the interval [0, 1] (as for probability), where 0 means "impossible" and 1 means "sure".

If E="Republicans will win", then γ may have following values (from soft to hard):

1. undetermined, in linguistic form, e.g. *may-be*E,
2. fuzzy set, with different granularity, determined on the interval [0, 1]: *possible*E,
3. subinterval of [0, 1]: [0.6, 0.8]E,
4. precise value from the interval (0,1): 0.7E,
5. value 0 or 1, e.g. E.

This example shows that sometimes it is very hard to obtain harder factors.

The basic assumption of this approach is that, inserting certainty factors γ and qualifiers α into conventional expressions of knowledge representation, we can control their level of hardness, preserving common representation for soft and hard knowledge.

3. KNOWLEDGE REPRESENTED BY RELATIONS

Nowadays a whole family of relational knowledge representation systems has been built using one of Description Languages (DL). Languages from this group differ with respect of their expressiveness and their complexity, and they have been used for building a variety of applications ([5,6]). Here will be presented only some basic principles of DL, needed for explanation of suggested modifications.

In DL *concepts* are used to represent classes as sets of *individuals*, and *roles* are binary relations used to specify their properties or attributes.

Descriptions start from three alphabets of symbols:
concept names or *primitive concepts* (denoted by A)
role names (denoted by R) and
individuals (denoted by a *and* b).

A *concept* or *concept expression* (denoted C or D) is built out from concept names with the use of *constructors*, and represents information, mentioned in the knowledge definition.

An *assertion* is an expression of type $a{:}C$ ("a is C", "a is an instance of C") or an expression of type $(a,b){:}R$ ("(a,b) is an instance of R").

Semantics of DL is based on the notion of interpretation, usually taken from sets or predicates.

Thus, assertion $a{:}C$ can be interpreted as $a \in S_C$ or $P_C(a)$, where S and P are appropriate sets and binary predicates. For example:

JOHN: Republican means that
John ∈ REPUBLICANS or Republican(John), and
(JOHN, USA):inhabitant replaces predicate
Inhabitant(John, USA).

Information described by these assertions may be uncertain, changing its form to $\gamma\,[a{:}C]$ or $\gamma\,[(a,b){:}R]$ (another approach is presented in [7]). For example

almost-sure [JOHN: Republican] and
possibly [(JOHN, USA):inhabitant].

When concept or role can be evaluated qualitatively, assertions are of the form $a{:}\alpha C$ (or $\gamma\,[a{:}\alpha C]$) as in JOHN:*very*sick or *almost-sure* [JOHN:*very*sick] and $(a,b){:}\alpha R$ or $\gamma\,[(a,b){:}\alpha R]$ as in

(JOHN, MARY)*dearly*loves.

If C and D are concepts, then so are $C \oplus D$ (concept disjunction), $C \otimes D$ (conjunction)[2], $\neg C$ (negation), $\forall R.C$ (universal quantification) and $\exists R.C$ (existential quantification).

Expression $C \oplus D$ is interpreted as $S_C \cup S_D$ or $P_C \vee P_C$ and can be extended to $\gamma_C\, C \oplus \gamma_D\, D$ (conjunction and negation – similarly). For example

[1] For simplicity it is assumed that a membership function is trapezoidal and components of $\langle x,y,v,w\rangle$ show the characteristic points of a trapezium.
Since $\langle x,x,v,v\rangle \equiv [x,v]$ and $[x,x,x,x] \equiv x$, trapezoidal description is the all-purpose tool.

[2] Symbols used in Description Logic are quite different, but they are not provided by Microsoft Word, obligatory in this publication.

0.3 Fighter ⊗ 0.5 Terrorist

More complex are the interpretations of $\forall R.C$:

$\{a: \forall b \ (a,b) \notin R \text{ or } b \in S_C\}$,

$\{a: \forall b \ \neg P_R(a,b) \text{ or } P_C(b)\}$.

For $\exists R.C$ the interpretations look as follows:

$\{a: \exists b \ (a,b) \in R \text{ and } b \in S_C\}$,

$\{a: \exists b \ P_R(a,b) \text{ and } P_C(b)\}$.

Some examples will make these constructors more clear.

Expression $\forall R.C$ designates *all* individuals which are in relation R with individuals from the concept C. All people having a boss or business will be described by

∀supervised.Boss ⊕ Businessman.

Expression $\exists R.C$ designates *some* individuals which are in relation R with individuals from the concept C. For example: somebody who teaches two courses belongs to the concept

∃teaches.(Mathematics ⊗ Physics).

As in the previous cases, concepts and roles can be equipped, if appropriate, with certainty factors γ and qualifiers α, but separate factors can also make quantifiers more soft:

almost∀supervised.Boss ⊕ Businessman.

Direct relations between concepts are described by *a terminology*: a finite set of *concept definitions* (expressions of the form $A:=C$) and *concept inclusions* (of the form $C \sphericalangle D$).

A concept definition allows stating a new name for the complex concept, e.g.

Son := Male ⊗ Child.

An inclusion allows stating the existence of a specialization ("more specific than") between concepts. For instance,

Smoking ∠ ∃causes.Cancer

In the case of concept definition one should decide how to propagate certainty factors from the right side to the left side, that means what is the value of γ_A in the definition $\gamma_A A := \gamma_C C \otimes \gamma_D D$. Since the tasks of constructors ⊕, ⊗ and ¬ are similar to the tasks of logical operators ∨, ∧ and ¬, it is justifiable using norms and co-norms, as in reasoning with logical formulae.

Finite sets of terminological axioms $\mathcal{T}$ and assertions $\mathcal{A}$ create a *knowledge base* $\mathcal{KB}=\langle \mathcal{T}, \mathcal{A} \rangle$.

In order for designers to be able to use Description Logic to model their application domains, it is important for the DL constructs to be easily understandable. Concepts, roles and primitives are defined in natural language but the abstract notation commonly used in DL is not fully satisfactory. To improve this situation, in some practical implementations symbols are substituted for simple words: ***or*** (⊕), ***and*** (⊗), ***not*** (¬), ***all*** (∀), ***some*** (∃) etc. These make DL expression more understandable.

4. KNOWLEDGE REPRESENTED BY MAPPINGS

Components of a mapping *map*: $inf_S \rightarrow inf_D$ are usually presented in the form of logical expressions, known as *conditions* (Φ) and *decisions* (Ψ). Functional dependencies between them are described by implication, giving the well known structure of a *decision (production) rule*: $\Phi \Rightarrow \Psi$. Similarly, if *transactions* are used as components, a mapping is known as *association rule.*

Three types of rules, being in practical use, are determined by three types of assertions φ and ψ - building blocks of expressions Φ and Ψ, with truth values *TRUTH and FALSE*. These are:

1 –*logical propositions*– sentences in natural language,

2 - *attributive statements* of the form
attribute-name = attribute-value ($a = v$) or
attribute is value (a *is* v),

3 - *predicates* - structured forms of statements:
$P(x,y,..)$.

All values present in these assertions may be qualified as more or less soft, with a full scale of qualifiers, e.g.:

1 – "Tax is small" or " Tax is equal 5%",

2 – Tax is small or Tax = 5,

3 – Equal(tax, small) or Equal(tax, 5).

Since a rule may be certain or not, typical (and simplified) set of rules $\mathcal{R}$ contains expressions of the form

$$\mathcal{R} = \mathbf{Y}_i \ \gamma_i [\varphi_{1i} \wedge \varphi_{2i} \wedge \ldots \Rightarrow \psi_{1i} \wedge \psi_{2i} \wedge \ldots].$$

Utilization of these rules is possible when there are known *facts* φ', equal or similar to some assertions φ (making conditions satisfied). A set of facts $\mathcal{F}$ comprises elements $\gamma_j \varphi'_j$, taking into account assertions certainty: $\mathcal{F} = \mathbf{Y}_j \gamma_j \varphi'_j$. There are known procedures, calculating certainty factors of decisions ψ on the basis of γ_i and γ_j, with chosen kind of norms.

A *knowledge base* is described by $\mathcal{KB}=\langle \mathcal{R}, \mathcal{F} \rangle$. Thus, having a knowledge base with factors γ and qualifiers α, we can influence a level of hardness.

During first steps of system design its goals are often presented in "soft" manner, therefore some decisions are imprecise and ambiguous. Inconsistency valuation can help in system improving and deserves consideration.

Each row *i* in a *decision table* or path in a *decision tree* designates *decision rule* of the form

$$(c_1 = v_{1i}) \wedge (c_2 = v_{2i}) \wedge \ldots \Rightarrow$$
$$(d_1 = w_{1i}) \wedge (d_2 = w_{2i}) \wedge \ldots$$

where *c,d* are condition and decision names and *v,w* their values. In the shorter form a set of such rules is described by

$$\mathbf{Y}_{i=1}^{k}\ \Phi_i \ \Rightarrow\ \Psi_i$$

Usually if $\Phi_a \ \Rightarrow\ \Psi_a$ and $\Phi_b \ \Rightarrow\ \Psi_b$ then for $\Phi_a = \Phi_b$ there is $\Psi_a = \Psi_b$, but in the tables or rule sets with ambiguous decisions a measure of inconsistency can be obtained with the help of *rough sets* [8,9].

Equal conditions *X* and decisions Y define two distinct partitions of the set of rules, with:

$$X_1 \cup X_2 \cup \ldots \cup X_x = \mathbf{Y}_{i=1}^{k}\ \Phi_i,$$
$$Y_1 \cup \mathrm{Y}_2 \cup \ldots \cup \mathrm{Y}_y = \mathbf{Y}_{i=1}^{k} \Psi_i .$$

If conditions from a class X_i, in each appropriate rule, generate decisions from one only class Y_i (described by $X_i \rightarrow Y_i$) then decisions are undertaken univocally. If conditions from a class X_i generate decisions from two or more classes: $X_i \rightarrow Y_i \cup Y_j$, and there exists a mapping $X_j \rightarrow Y_j$, then decisions are ambiguous. A class Y_i designates the *lower approximation* of decisions when conditions are X_i, and a set $Y_i \cup Y_j$ fixes the *upper approximation* of decisions. The formula

$$\eta_i = \frac{Y_i}{Y_i \cup Y_j}$$

can be used as a measure of decision hardness: if $\eta_i = 1$ decision is hard, if $\eta_i < 1$ - decision is more or less soft.

5. CONCLUSION

Knowledge concerning soft and hard system is also soft or hard. The level of softness changes during system design, so it will be useful to have common representation for different kind of knowledge.

Examples presented above show that typical knowledge representation methods, modified by special factors, can control a level of softness of expressions, describing real systems.

.

REFERENCES

[1] P. Checkland and J. Scholes. *Soft Systems Methodology in Action.* John Wiley, 1999.

[2] P.M. Hildreth and Ch. Kimble. The Duality of Knowledge. *Information Research*, vol.8, no.1, 2002.

[3] I. Nonaka and H. Takeuchi. *The Knowledge – Creating Company.* Oxford University Press, New York, 1995.

[4] A.P. Wierzbicki and Y. Nakamori. *Creative Space and Creative Environments* (in the press).

[5] F. Baader *et al. The Description Logic Handbook.* Cambridge University Press, 2003.

[6] B.N. Grosof *et al.,* Description Logic Programs: Combining Logic Programs with Description Logic. *Proc. WWW2003,* Budapest, Hungary.

[7] U. Straccia. Reasoning within Fuzzy Descrip-tion Logic. *Journal of Artificial Intelligence Research, 14, 2001.*

[8] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data.* Kluwer Acad. Publ.,1991.

[9] T.Y. Lin, Y.Y Yao. Mining soft rules Using Rough Sets and Neighborhoods. *CESA'96,* France.