Representation of Medical Knowledge by Arden Syntax with Fuzzy Extensions

Sven Tiffe[1] and Klaus-Peter Adlassnig[2]

[1]Siemens Medical Solutions, Med GT3, Henkestraße 127, D-91052 Erlangen, Germany
e-mail: Sven.Tiffe@med.siemens.de
[2]Department of Medical Computer Sciences, Section on Medical Expert and Knowledge-Based Systems
University of Vienna Medical School, Spitalgasse 23, A-1090 Vienna, Austria
e-mail: kpa@akh-wien.ac.at

Abstract. *The representation of knowledge is one of the fundamental research areas in the realm of artificial intelligence. One approach to this subject has been provided by the Arden Syntax for Medical Logic Systems. At present, this syntax includes typical procedural programming language elements used to define crisp decision logic. However, medical knowledge usually contains linguistic uncertainty that can be represented and manipulated by concepts of fuzzy set theory and fuzzy logic. This paper presents concepts for extending Arden Syntax elements, such as operators, data types, and statements by concepts of fuzzy set theory and fuzzy logic to represent rules with inherent uncertainty.*

1. Introduction

Medical knowledge frequently exists in descriptive form and contains linguistic uncertainty. For instance, one textual definition of high blood pressure (cited from [1] according to [2], translated from German) is:

"From a medical point of view, the clinical degree of severity III in high blood pressure can be described as follows: (a) Systolic blood pressure higher than 200 Torr or diastolic blood pressure between 120 and 130 Torr, (b) *largely stable* blood pressure level, and (c) *very often* an indication of cardiac insufficiency, changes in the fundus of the eye, *stronger* pain, *sometimes* cerebral disturbances."

The inherent uncertainty incorporates flexibility in terms of decision-making for the treating physician. Uncertainty due to lexical fuzziness—named vagueness—can be handled by concepts based on fuzzy set theory. While fuzzy sets describe the relationships between data and linguistic terms by degrees of compatibility, the relationships between linguistic terms can be modeled by fuzzy logical operations [3].

The Arden Syntax is a language for the representation of medical knowledge and can be used for clinical decision support [4]. It was first standardized by the American Society for Testing &

Materials in 1992 [5]. In 1998, the maintenance of the Arden Syntax was taken over by the Health Level 7 Committee (HL7). Version 2.0 is now available as an ANSI standard [6].

An Arden Syntax knowledge base consists of a set of single independent rules known as Medical Logic Modules (MLMs). Each MLM provides process-specific knowledge and usually contains sufficient knowledge for one specific medical decision [7].

The basic function of an MLM—decision-making—is based on query, manipulation, and evaluation of data. The Arden Syntax comprises different data types for representing numerical values, character strings, truth values, temporal values, and lists.

In order to handle logical operations, a trivalent logic is used. The classical logical operators and, or, and not, are defined to operate on true, false, and additionally on null. The use of an additional value for logical variables was first introduced by Kleene, who termed the third value 'undefined' [8]. The Arden Syntax uses this value, which can be assigned to any variable of any data type, to express uncertainty. The uncertainty may arise from missing data in the database, errors during the execution of operators (e.g., division by 0), or due to an explicit assignment of null. Thus, the uncertainty represented by null rather signifies incompleteness than vagueness. To avoid misunderstandings we use the term 'fuzziness' to express linguistic uncertainty in the present paper.

2. Methods

Since the Arden Syntax does not provide elements to represent vague knowledge, one goal of our work was to extend the syntax by concepts of fuzzy set theory and fuzzy logic. The system is expected to provide better representation of rules given in natural medical language and human reasoning.

Basically, fuzziness can be contained in those parts of an MLM which include selection criteria, for example, when selecting data from a database, or branch conditions in the decision logic. These conditions are defined by the author of the MLM and are based on knowledge that usually includes fuzziness as mentioned earlier.

Technically, such criteria are formulated using comparison operators. Furthermore, the resulting fuzziness must be processed and taken into account in the execution of an MLM.

2.1. Operators

In order to handle uncertain comparisons, we selected a subset of the comparison operators defined in Arden Syntax and extended them.

Arden provides basic comparison operators, such as 'is equal to', 'is less than', or 'is greater than'. Such comparisons can be described by compatibility functions, which define the relationship

between a value and a term—the comparison—and return either 0 (false) or 1 (true), as shown in the following example.

Example 1: The expression 'diastolic blood pressure is increased' can be represented by a crisp comparison $x > x_0$ and can be formulated in Arden as

```
x is greater than x0.
```

Here, `x` stands for the measured diastolic blood pressure and `x0` for the threshold, which indicates the unintuitively sudden transition to an increased diastolic blood pressure. The comparison is defined by the compatibility function μ(x) as shown in Figure 1.

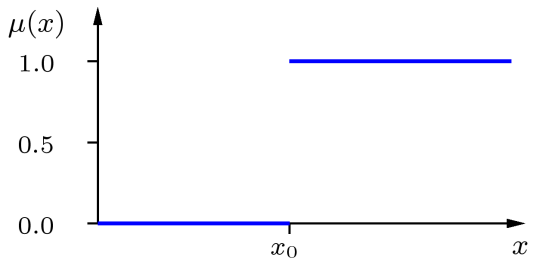

Figure 1: Compatibility function for a crisply defined comparison 'x is greater than x_0'.

The crisp definition of the comparison operator in Example 1 causes an abrupt change in the output value when the input value exceeds the threshold x_0. To avoid this abrupt change, we redefined the respective compatibility function of the operator. We defined a gradual transition between false and true using an additional parameter, namely Δx. Example 2 shows an extended 'greater than':

Example 2: In contrast to the compatibility function of Example 1, the fuzzy definition of the comparison operator is defined using a fuzzy threshold (Figure 2). Δx is included in the Arden representation of this operator as follows:

```
x is greater than x0 fuzzified by delta
```

where `delta` is replaced by the value of Δx.

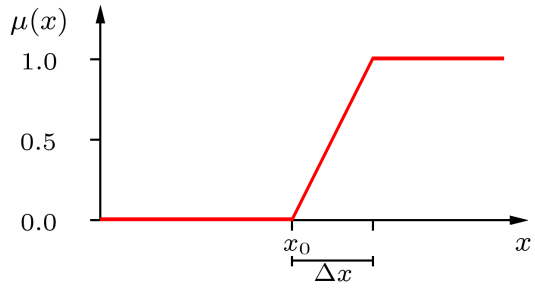

Figure 2: Compatibility function for a fuzzily defined comparison 'x is greater than x_0'.

The result of this fuzzily defined comparison is a fuzzy truth value between 0 and 1. The fuzzy comparison is closer to intuitive human reasoning than is a crisp one. Furthermore, this extension can be used to stretch the range of selected data as shown in Example 3 later.

More complex comparisons, such as temporal or range comparisons, can be expressed using a combination of simple comparisons. Thus, the extension can analogously be applied to these operators.

As the operators technically create data with inherent uncertainty, we term this set of operators 'fuzzy generating operators':

Definition 1: A *fuzzy generating operator* is an operator which creates a fuzzy truth value based on a fuzzy set compatibility function.

Not every comparison operator was assigned fuzzy behavior. For example, Arden Syntax provides comparison operators which check the type of a data value, such as 'is number'. Such comparisons are crisp by definition. In contrast to fuzzy generating operators, the resulting fuzziness does not stem from these operators themselves, but is due to inherent fuzziness in the arguments.

Definition 2: A *fuzzy processing operator* is an operator which returns data with associated fuzziness, if any. This fuzziness results from the inherent uncertainty of arguments, not from the operator itself.

The set of fuzzy processing operators comprises all remaining operators of the Arden Syntax, such as arithmetical, numerical, and logical operators.

2.2. Data Types

Actually, a variable holds either a data value or null. By including fuzziness, a variable may hold a value that has a certain degree of compatibility with the associated expression, e.g., its selection criterion.

The following extensions have been made for individual data types:

Boolean: The Boolean value was expanded to support uncertain truth values. First, the former range of the Boolean value

$$\{\text{false, true}\}$$

was interpreted as

$$\{0, 1\}$$

where 0 corresponds to false and 1 to true. Then, the two truth values were extended to a range of truth values

$$\{[0, 1] \in R\}$$

to specify a degree of truth. The former values false and true now correspond to the boundary values 0 and 1. This new data type, which represents gradual truth values, is termed 'fuzzy'.

Number, Time, Duration, and String: These values may be the result of a sequence of operators which might include uncertainty. Thus, the result includes a degree of compatibility with the overall operation, that is defined in the value range $[0, 1] \in R$. These data types are internally represented by a tuple composed of the data item and the degree of compatibility.

Lists: Lists may have a certain degree of compatibility with an operation. This uncertainty affects the degree of compatibility of all data items in the list. The resulting degree of compatibility of a single data item is calculated by applying a fuzzy 'and' operation, like the algebraic product of the degree of compatibility of the data item and the list

$$A \wedge B = \{(x,\mu) \mid \mu = \mu_A(x)\, \mu_B(x)\}. \quad (1)$$

In the present study, we use the following terms to distinguish between conventional data types and data types extended with fuzziness:

Definition 3: A *crisp data value* or *crisp data* is a classical data value of the Arden Syntax, whose variables hold either null or a concrete value. In the following examples, a crisp data value is written in the notation

<value>

e.g., <42> or <null>.

Definition 4: A *fuzzy data value* or *fuzzy data* is a data value with an associated degree of compatibility with an operation. In the following examples, a fuzzy data value is written in the notation

<value, degree of compatibility>

e.g., <42, 0.9> or <42, 0.0>.

Fuzzy truth values are indicated as floating point numbers with a leading F, e.g., F0.8.

The following example illustrates the use of fuzzy generating and processing operators.

Example 3: The average of test results of the past three days is to be computed. In order to additionally consider the test results, collected shortly before this period, the range of selected data was stretched by 12 hours using a fuzzily defined selection. This selection is sketched in Figure 3. The variable t_0 defines the actual time, t_1 to t_5 are time stamps of test results collected within the preceding 84 hours.

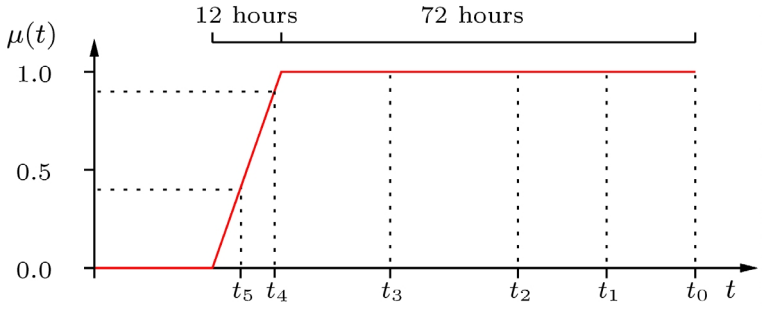

Figure 3: Selection of test results from a previous period.

Two of these five selected test results, collected at t_4 and t_5, are within the stretched range and would not have been considered in a crisp selection. However, by making a fuzzy selection, their respective values influence the result of this operation as well.

In Arden, this task can be formulated in the following way:

```
read average {} where it occurred within past 3 days
                fuzzified by 12 hours
```

Since the Arden operators are not outlined in this paper, the processing will be explained step by step:

First, an institution-specific instruction within the curled brackets* of the read statement returns a list of crisp data elements from the database:

`read {}` → (<42>, <40>, <38>, <41>, <42>)

The next operation contains a fuzzy operator and returns a list of fuzzy truth values:

```
it occurred within past 3 days
    fuzzified by 12 hours
```
→ (F1.0, F1.0, F1.0, F0.9, F0.4)

These values represent the degree of compatibility of the collection time of the single test results to the expression ‘within past 3 days’, which is defined as a fuzzy set with Δx set to twelve hours.

Next, the crisp operator ‘where’ selects elements from its left argument based on the truth value of its right argument:

`(<42>, <40>, <38>, <41>, <42>) where (F1.0, F1.0, F1.0, F0.9, F0.4)`
→ ([42, 1.0], [40, 1.0], [38, 1.0], [41, 0.9], [42, 0.4])

In this step, crisp data and fuzzy truth values are combined to fuzzy data.

* The space within the curled brackets was intentionally left blank in the example.

The last operator 'average' computes the average of the values and must consider the fuzziness of individual data elements. In this example, the chosen average operator considers fuzzy data only in proportion to their degree of compatibility using

$$avg(x_1,\ldots,x_n)=\frac{\sum_{i=1}^{n} x_i\mu(x_i)}{\sum_{i=1}^{n}\mu(x_i)} \tag{2}$$

Finally, the operation results in [40.4, 1.0], whereas the crisp operation would have computed only the average of the first three values returning 40.

The average operator is a rather complex example of a fuzzy processing operator. In general, the fuzziness of the result is determined by selecting the fuzziest argument and adopting its degree of compatibility.

2.3. Defuzzification

During the execution of an MLM, individual decisions have to be made. The use of fuzzy truth values causes decision elements, such as the if-then statement, the while loop, or the conclude statement, to handle fuzzy conditions. Different concepts for fuzzy algorithms have been presented in the literature; formal definitions can be found in [9].

If-then statement: An Arden Syntax if-then statement controls whether a block of statements would have been executed or not. It is only executed if the condition is met; otherwise an optional else block is executed.

On the other hand, using Fuzzy Arden, both blocks are executed in parallel, if the condition is neither false nor true. Each complete block has a degree of compatibility with the condition, which can be interpreted as the significance of its actual execution.

The significance of the first code block is equal to the truth value *d* of the condition, that of the second block is equal to the complementary truth value *1-d*. Figure 4 shows a scheme for a fuzzy if-then statement; the gray boxes indicate the significance of the code blocks.

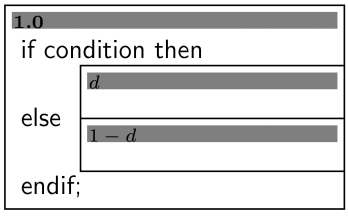

Figure 4: Scheme for a fuzzy if-then statement.

The significance of the code block, which encloses the entire statement, is 1.0, because its execution does not depend on any conditions.

The execution of every statement and operator within each of the code blocks is influenced by the respective significance. Thus, an included if-then statement would be influenced as shown in Figure 5.

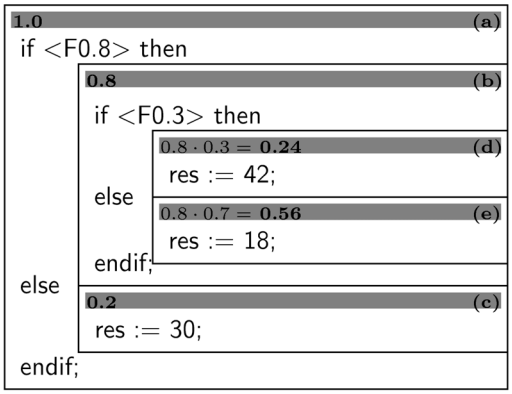

Figure 5: Data assignment within nested fuzzy if-then statements.

The following example illustrates this concept:

Example 4: If the condition of the outer if-then statement were 0.8, the significance of the first code block (b) would be 0.8, and the significance of the second one (c), 0.2.

If the condition of the inner if-then statement were 0.3, the significance of the inner code blocks (d) and (e) would depend not only on this condition, but additionally on the significance of the enclosing code block (b), and thus on the condition of the outer if-then statement. The significance of the inner code blocks is calculated using the product of both conditions for code block (d), and the product of the outer condition and the complementary of the inner condition for code block (e).

Since the execution of every statement and operator is influenced by the significance of the current code block, every data assignment returns fuzzy data values. If—as shown in Figure 5—a variable holds different data in different code blocks, the final value of the variable will be computed as an average value, where every individual value is weighted by its fuzziness (see equation 2).

In the above example, the value of the variable res would be computed as follows:

- In the inner code blocks, res would hold [42, 0.24] (d) and [18, 0.56] (e).
- In code block (b), res would hold the weighted average of these values [25.2, 0.8]. In code block (c), res would hold [30, 0.2].
- The value of res after execution of the entire statement would be [26.16, 1.0].

If it is not possible to compute an average value, e.g., when using string variables, the value with the lowest fuzziness (the highest degree of compatibility) is chosen.

While loop statement: A while loop can be handled in a similar way.

Conclude statement: The result of an MLM usually is a message, which is generated in the action slot. This slot gets executed only when a conclude statement is executed, whose crisp condition returns true.

Using fuzzy truth values, the decision logic of an MLM might conclude neither false nor true. Thus, true or false conclusions are now borderline cases of an extended range of conclusions. Furthermore, it is possible to execute more than one conclude statement with different fuzzy truth values, e.g., within if-then-else blocks.

Now, the action slot is always executed. The fuzzy truth values of the conclude statements are averaged in the same way as variables are defuzzified. The overall concluding truth value can be accessed in the action slot. It is either possible to include this truth value in the message of the MLM or to select a message from a set of different messages based on the truth value, for example.

3. Conclusion

These concepts show that uncertainty can be supported in Arden by means of small extensions of the syntax. The extensions offer an alternative way to represent linguistic medical rules and their inherent uncertainty.

Technically, the generation of fuzzy data based on the inherent uncertainty of the medical rule can be realized rather simply by choosing a subset of operators provided by Arden and extending them by one or two additional arguments, which describe the compatibility function of the fuzzy set.

The consideration of uncertainty in Medical Logic Modules will change the design process of medical decision logic. Instead of deciding whether or not to send a message, for example, it would be possible to send a message extended by the final concluded fuzzy truth value.

References

[1] Bocklisch, S.F. (1987) *Prozeßanalyse mit unscharfen Verfahren*. VEB Verlag Technik, Berlin.

[2] Bothe, H.H. (1993) *Fuzzy Logic – Einführung in Theorie und Anwendungen*. Springer Verlag, Berlin.

[3] Klir, G.J. and Folger, T.A. (1988) *Fuzzy Sets, Uncertainty, and Information*. Prentice Hall, London.

[4] Ahlfeld, H., Johansson, B., Linnarsson, R., and Wigertz, O. (1994) Experiences from the Use of Data-Driven Decision Support in Different Environments. *Computers in Biology and Medicine* 26, 397–404.

[5] American Society for Testing & Materials Committee E-31. (1992) *Standard Specification for Defining and Sharing Modular Health Knowledge Bases (Arden Syntax for Medical Logic Modules)*, Philadelphia.

[6] Health Level Seven. (1999) *Arden Syntax for Medical Logic Systems*. Health Level Seven, Inc., 3300 Washtenaw Ave, Suite 227, Ann Arbor, MI 48104.

[7] Hripcsak, G. (1994) Writing Arden Syntax Medical Logic Modules. *Computers in Biology and Medicine* 24, 331–363.

[8] Kleene, S.C. (1938) On Notation for Ordinal Numbers. *Journal of Symbolic Logic* 3, 150–155.

[9] Zadeh, L.A. (1968) Fuzzy Algorithms. *Information and Control* 12, 94–102.