MEDINFO 2010
C. Safran et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-588-4-831

Processing Gradual Information with Fuzzy Arden Syntax

Thomas Vetterlein[a], Harald Mandl[b], Klaus Peter Adlassnig[a,b]

[a] *Section for Medical Expert and Knowledge-Based Systems, Medical University of Vienna, Austria*
[b] *Medexter Healthcare GmbH, Vienna, Austria*

Abstract

The programming language Arden Syntax is especially adapted to the needs of computer-based clinical decision support. Recently, an extension of Arden Syntax, named Fuzzy Arden Syntax, was proposed by the authors. Fuzzy Arden Syntax is a conservative extension of Arden Syntax and offers special functionality to process gradual information. The background is the observation that in medicine we frequently deal with statements which are neither clearly false nor clearly true but hold to some intermediate degree. In this paper, we demonstrate under which circumstances a Medical Logic Module (a program unit written in Arden Syntax) may show unintended behavior and how the situation can easily be improved by means of the possibilities offered by Fuzzy Arden Syntax. To this end, an example from the domain of nosocomial infection control is discussed in detail.

Keywords:

Clinical Decision Support Systems, Arden syntax, Fuzzy logic, Nosocomial infections.

Introduction

An important aim in medical information sciences is to define standards to represent clinical data in a computer-interpretable form. This task concerns various levels of abstraction and is generally difficult. We may mention, for example, current efforts to establish standards for the structure of an electronic health record, to establish medical terminologies on a symbolic level, or to define computer-interpretable medical procedures.

In healthcare, an issue which becomes the more relevant the more data is available electronically is the automated and intelligent interpretation of facts concerning a patient's course of disease, like symptoms, signs, diagnoses, and treatment. On the lowest technical level, this purpose is supported by a programming language in which general clinical knowledge can be formulated as directly as possible, minimizing the need for the user to get familiar with technical peculiarities.

Arden Syntax

In this latter field, the developers of Arden Syntax have contributed in a specific but valuable way [1]. The programming language Arden Syntax is endowed with an (institution-dependent) interface to a patient database, can be applied to encode methods to interpret the data, and can be used to trigger appropriate reactions in real time. Its syntax has been chosen to make the program flow easily traceable. Arden Syntax was originally developed by the American Society for Testing and Materials in 1992 and is at present maintained by Health Level Seven, Inc. (HL7). The document containing the official specification [1] can be acquired from HL7; (see www.hl7.org.)

Arden Syntax has been developed as a standard of representing general clinical knowledge. This knowledge needs to be present in modular form; an Arden Syntax system is composed of an unstructured set of so-called Medical Logic Modules, or MLMs for short. Each MLM can be designed to react in real time to some specified event, like for instance a certain change in the patient database, in which case a message to the host system is sent or other reactions are induced.

Fuzzy Arden Syntax

The content of an Arden Syntax MLM can be the result of a formalization of information provided in natural language. When mapping information contained in a text to a computer program, even if this text is meant as a precise specification of facts or procedures, we typically encounter the problem that the level of precision is not sufficient to allow the formulation of the program in a unique way. Often then, the original specification needs to be extended in an ad-hoc way. To give a simple example of what we mean, for a clinician it might not be a problem to interpret statements like "the level of aspartate transaminase (AST) is significantly increased" if the normal range of AST is known; in a computer program however sharp limits must be provided.

Conversely, if precise specifications are provided it might sometimes not be desirable to follow them in a purely mechanical way. For example, consider the condition "body temperature $\geq$ 38.0 °C". If the actual body temperature of a patient is, say, 37.9 °C, a clinician might hesitate to simply consider the condition as not fulfilled or might at least argue differently than in the situation when the temperature is 36.9 °C.

If sharp limit points are not specified or even unknown, some extra effort will be needed to deal with this situation in automated decision support. Furthermore, if sharp limit points are

chosen, borderline cases will in general not be apparent in the output: a condition "temperature ≥ 38.0 °C" will be evaluated negative if the communicated temperature is lower than 38 °C, and positive if it is 38 °C or higher, and information about changes of the result under small changes of the input would again require extra effort. It is the idea of Fuzzy Arden Syntax to include into Arden Syntax additional functionality to make the definition of "soft" limit points possible and furthermore to avoid discontinuities in the results.

Fuzzy Arden Syntax was originally proposed by S. Tiffe [2]. It has recently been further elaborated by the authors; we refer to [3] for a complete specification. The new features have been summarized in [4]. The present note shows the effects of these features on the basis of a specific example.

A case study: nosocomial infection control

We will exhibit some features concerning behavior and performance of Arden Syntax with or without the extension provided by Fuzzy Arden Syntax. To this end, we shall consider a typical example of computer-based clinical decision support.

Computer-supported detection of nosocomial infections

For the surveillance of nosocomial infections in ICUs, standards have been defined [5–7]. If the relevant data is available electronically, the possibility exists to have suspicious cases detected automatically. For instance, at the Vienna General Hospital, a system with exactly this purpose has been developed [8]. It is named Moni/Surveillance-ICU and based on HELICS [5] as well as KISS [6, 7] criteria for nosocomial infections.

The following conditions must be fulfilled:

- It is not the case that the patient has fever (≥ 38.0 °C), or urinary urgency, or frequency, or dysuria, or suprapubic tenderness.
- The patient has had
  1. either a positive urine culture ($\geq 10^5$ microorganisms/cm^3 of ≤ 2 species) and an indwelling urinary catheter within 7 days before the culture
  2. or two positive urine cultures (the ≤ 2 species being the same) and no indwelling urinary catheter within 7 days before the culture

Figure 1 - Definition of asymptomatic bacteriuria (UTI-C)

We shall consider the following example which is a specification of a nosocomial infection according to [5] (coded UTI-C there). This criterion is part of the Moni system as well. We note that we have slightly changed the formulation of [5].

As we will see, the implementation of this rule in Arden Syntax is straightforward. Arden Syntax programs are to a certain extent self-explanatory; for this reason also the reader not familiar with Arden Syntax should be able to follow our argumentation. To be sure, we will however include some basic explanations.

An MLM has a predefined structure. The first two parts are the *maintenance category* and the *library category*, which are not important here. Only the third part, the *knowledge category*, contains the executable program. The latter consists in turn of several so-called *slots*. In particular, the *data slot* contains the commands to read the relevant data from the host and possibly additional commands to preprocess this data; in the *logic slot* the decision is made if the *action slot* is executed. A jump to the action slot is caused by the command `conclude true`, whereas `conclude false` terminates execution.

The relevant slots of the knowledge category of an MLM modeling the criterion for asymptomatic bacteriuria could look as follows.

```
data:
   (Patient, Date) := argument;
     /* Input data: patient ID and date of stay */
   Temperature :=
     read {body_temperature,
           Patient, Date};
     /* Body temperature */
   Fever := Temperature >= 38;
   Urgency :=
     read {urinary_urgency, Patient, Date};
     /* Urge to urinate? */
   Micturition :=
     read {micturition, Patient, Date};
     /* Increased frequency of urination? */
   Dysuria :=
     read {dysuria, Patient, Date};
     /* Painful urination? */
   Suprapubic_tenderness :=
     read {suprap_tenderness, Patient, Date};
     /* Suprapubic tenderness? */
   Organisms_in_1_urine_culture :=
     read {organ_1_urcult, Patient, Date};
     /* Number of microorganisms of ≤ 2 species per
     mm^3 */
   One_urine_culture :=
     Organ_in_1_urine_culture >= 1e5;
   if One_urine_culture
     then Urine_culture_time :=
       time of One_urine_culture;
     endif;
   Organisms_in_2_urine_cultures :=
     read {organ_2_urcult, Patient, Date};
     /* Number of microorganisms of ≤ 2 species per
     mm^3 in the last two cultures */
   Two_urine_cultures :=
     Organisms_in_2_urine_cultures >= 1e5;
   if Two_urine_cultures
     then Urine_culture_time :=
       time of One_urine_culture;
     endif;
   Catheter :=
     read {catheter, Patient, Date};
     /* Latest indwelling urinary catheter */
   ;;

logic:
```

```
   if (Fever OR Urgency OR Micturition OR Dy-
       suria OR Suprapubic_tenderness)
     then conclude false;
     endif;
   if (Catheter is present
       AND time of Catheter is at least 7 days
       before Urine_culture_time)
     then if One_urine_culture
       then conclude true; endif;
     else if Two_urine_cultures
       then conclude true; endif;
     endif;
   ;;
action:
   write "The conditions of an
       asymptomatic bacteriuria are met.";
   ;;
```

Behavior in borderline cases

We may say that this MLM reflects the specification UTI-C of asymptomatic bacteriuria one-to-one. However, we may wonder if this is what we want. As a matter of fact, it is easy to construct a case where a clinician familiar with the facts expressed by UTI-C is likely to conclude differently from the MLM. We have in mind the borderline cases; consider the following situations:

- A patient had three positive urine cultures with the same single species and an indwelling urinary catheter until one day before the first positive culture; he has no urgency, frequency, dysuria, or suprapubic tenderness and a temperature of 38.0 °C.
- A patient had a positive urine culture and a urinary catheter until seven and a half days before the culture; he has no fever, urgency, frequency, dysuria, or suprapubic tenderness.
- In the urine culture of a patient $9 \cdot 10^4$ microorganisms/cm^3 were found, the patient had two days earlier an indwelling urinary catheter, and otherwise no fever, urgency, frequency, dysuria, or suprapubic tenderness.

It would be easy to continue this list of cases, which have all in common that an automated detection of an infection fails. However, a clinician will most likely judge in all three cases that the criterion does apply, at least to a certain extent, even if criterion UTI-C is the only source of information.

Fuzzification of formalized clinical criteria

In computer-assisted decision support, one might certainly opt to have information provided only in the clear cases. After all, nobody would expect that no information about a possible disease means that the disease is not present. On the other hand, there are easy means to make the inference of a program like the above MLM more flexible so as to get results also in those cases which are not entirely conclusive on the basis of some given formal rules. Of course then it is essential to exhibit in the output explicitly that the result has restricted value.

These considerations have led to the development of Fuzzy Arden Syntax. Note first that under Fuzzy Arden Syntax, the above MLM would run without change of effect. That is, the same input would lead to the same output. This is why we call the extension "conservative".

However, Fuzzy Arden Syntax offers convenient possibilities to process not fully determinate truth values and to extend the inference coded in a given MLM to borderline cases. In medicine, the situation frequently occurs that facts about a patient do not fit perfectly to the available notions. Indeed, expressions like "having pain" or "being elevated" involve vagueness in the sense that they possess borderline cases in which it is hard or impossible, in any case unreasonable, to tell if they apply or do not.

As the starting point, we borrow from fuzzy logic [9] the simple idea to extend the two-element set of classical truth values to a continuous set of truth values. In classical propositional logic, we use 0 to denote falsity, 1 to denote truth; in fuzzy logic we use in addition any real value between 0 and 1 to express tendencies.

Examples of vague notions are immediate from the criterion UTI-C. Urinary urgency, frequency, dysuria, suprapubic tenderness may in most cases clearly hold or not hold, but there can be unclear situations as well.

The values allowed for yes-no variables in Fuzzy Arden Syntax certainly include the clear "true" or "false" just like in Arden Syntax. In addition, however, also values like 0.8 meaning, say, "rather true" are possible. We note that if continuous values are used for yes-no statements in the input, no changes in the MLM are necessary.

If intermediate truth values are used, it must however be determined in which way the connectives "and", "or", and "not" are to be interpreted. By default, if the variable `Var_1` contains the value $v_1 \in [0, 1]$ and the variable `Var_2` contains the value $v_2 \in [0, 1]$, then

```
Var_1 AND Var_2, Var_1 OR Var_2,
NOT Var_1
```

will be evaluated as $\min\{ v_1 , v_2 \}$, $\max\{ v_1 , v_2 \}$, and $1 - v_1$, respectively. Other interpretations, in particular other t-norms [10] interpreting `AND` are possible.

Let us next describe the method to avoid sharp limit points. Namely, we may replace sharp values by "soft" ones in a straightforward way. Our above-given example can be modified as follows:

```
data:
   [...]
   Fever :=
     Temperature >= 38 fuzzified by 0.5;
   [...]
   One_urine_culture :=
     Organisms_in_1_urine_culture >=
       1e5 fuzzified by 5e4;
   [...]
   Two_urine_cultures :=
     Organisms_in_2_urine_cultures >=
       1e5 fuzzified by 5e4;
```

```
   [...]
   ;;

logic:
   if (Fever OR Urgency OR Micturition OR Dy-
       suria OR Suprapubic_tenderness)
     then conclude false;
     endif;
   if (Catheter is present AND time of Cathe-
       ter is at least 7 days fuzzified by 2
       days before Urine_culture_time)
     then if One_urine_culture
       then conclude true; endif;
     else if Two_urine_cultures
       then conclude true; endif;
     endif;
   ;;
```

As to be expected, the expression `fuzzified by` is used to create some tolerance around a sharp real value. Formally, the expression

```
38 fuzzified by 0.5
```

is a triangular fuzzy set, namely the following one:

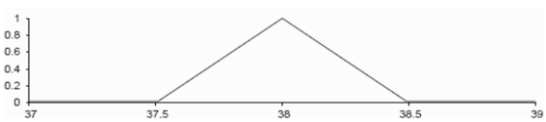

Figure 2- Example of a triangular fuzzy set

We recall that a fuzzy set is a mapping u from some universe of discourse M to the real unit interval [0, 1]. u typically models a natural-language concept, and for any $T \in [36, 42]$, the value $u(T)$ is then the degree to which T is in accordance with the concept modeled by u. The universe is typically R, the reals, and in Fuzzy Arden Syntax, we may use any fuzzy set over R provided it is piecewise linear, at each point left- or right-continuous, and constant outside a finite interval.

A fuzzy set like the displayed one can be used for a comparison: typically, a number like `Temperature`, which in our context is also called a *crisp number*, is compared with a fuzzy set, also called a *fuzzy number*. The expression

```
Fever:= Temperature >= 38 fuzzified by 0.5
```

no longer returns necessarily one of the sharp truth values `true` or `false`. The displayed case rather works as follows; let T be the value in `Temperature`. If $T \geq 38$, then the result is 1, or `true`, in accordance with the original MLM. If T is just slightly smaller than 38, the resulting value will still be close to 1 and in particular not reflect falsity; in the sequel the same consequences will be drawn, but with reduced weight. Falsity is the calculated outcome only if $T \leq 37.5$. This follows from our choice to fuzzify by 0.5. We conclude that `Fever` is no longer a two-valued, but a many-valued concept, with the effect that jumps from one extreme to the other are avoided.

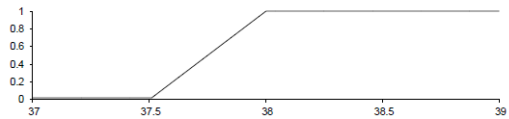

Figure 3 – Fuzzy logic and temperature

We see in particular that the specification of the conditions under which `Fever` applies requires more than determining one delimiting value. It is rather required to determine when this concept clearly applies and when this concept clearly does not apply. Accordingly the width of the transitional region must be set.

The same can be said mutatis mutandis for the variables `One_urine_culture` and `Two_urine_cultures` in our example.

The next question is what happens when an if-then-else construct depends on a many-valued condition. In Fuzzy Arden Syntax, the program splits. If the condition is evaluated to t such that $0 < t < 1$, the "if" block and the "elseif" block are executed in parallel. To any point in the program, there is associated a *program weight*; prior to the execution of the two branches this weight is multiplied by t or $1-t$, respectively.

Test case

We shall next illustrate the effect of this concept. To this end, we shall check the behavior of the above Fuzzy Arden MLM with some specific values, and we will see if the result is in accordance with the conclusions which we would draw intuitively.

Consider the following scenario. The patient does not suffer from urgency, micturition, dysuria, or suprapubic tenderness, and has a temperature of 37.6 °C; a catheter is reported to have been removed eight days before the first urine culture; he has two clearly positive urine cultures where however the number of microorganisms of coinciding species is only $6 \cdot 10^4$.

Let us guess how a clinician would judge this situation in view of the criterion UTI-C. The condition of UTI-C about the absence of symptoms is practically fully fulfilled; the height of the body temperature is not significant. Furthermore, the catheter was removed eight days, so roughly one week, before the first positive culture. Finally, there is some evidence of a second positive culture. All in all, the picture is presumably not entirely clear but rather well compatible with the conditions of UTI-C. In any case, the decision seems natural that the situation should be further examined.

Let us now compare these informal considerations with the result provided by the Fuzzy Arden MLM. We shall trace, step by step, the execution of the logic slot.

The expression

```
Fever OR Urgency OR Micturition OR
Dysuria OR Suprapubic_tenderness
```

returns, if `OR` is interpreted according to default, the largest of the values of the conjuncts. We have that `Fever` is 0.2 and `Urgency`, `Micturition`, `Dysuria`, and `Suprapubic_tenderness` are all 0; thus the result is 0.2.

The program splits; however, the line `conclude false`, to be executed with weight 0.2, terminates the MLM. So the remaining part of the program is continued with weight 1− 0.2= 0.8.

Next, we have that the expression

```
Catheter is present AND time of Catheter is at
least 7 days fuzzified by 2 days
```

is evaluated 0.5. Consequently, the program splits again; the `if` block and the `else` block are executed in parallel, both with weight $0.8 \cdot 0.5 = 0.4$. In the first branch, the program executes `conclude true` under the condition `One_urine_culture`. As this variable contains the value true, i.e., 1, the action slot is executed with weight $0.4 \cdot 1 = 0.4$. In the second branch, the program executes `conclude true` under the condition `Two_urine_cultures`. This variable contains the value 0.2 and the action slot is executed as well, but in this case with weight $0.4 \cdot 0.2 = 0.08$.

The action slot sends to the host the message that asymptomatic bacteriuria is present. This message will be endowed with a value expressing the fact that the program weight is decreased. In our case, the host will get independently twice the same message, once with weight 0.4 and once with weight 0.08; it could add up the two values 0.4 + 0.08 = 0.48. So as the final result, we are provided a message together with a truth value of only 0.48.

The value 0.48 implies that the message is to be understood with great caution. We conclude that the message is weaker than the informally drawn conclusion. But in accordance with our informal considerations it is clearly suggested that the situation needs to be examined.

We may finally observe that the corresponding Arden Syntax MLM would not report anything. We furthermore note that the modified MLM encoding UTI-C is not essentially longer or more complicated than the original one.

Conclusion

Clinical decision support systems depend on clear specifications: criteria must be provided which do not leave room for interpretation. This is the nature of a computer program as opposed to the way a human reasons who always allows some tolerance in considerations. An extended version of Arden Syntax, named Fuzzy Arden Syntax, aims at reducing the undesirable effects of sharp delimitation of situations.

Fuzzy Arden Syntax is based on the principles of fuzzy logic and uses a continuous set of truth values, namely the real unit interval. We have demonstrated on the basis of an example, a criterion of a nosocomial infection which is in practical use, the benefits of this approach. Answers to yes-no questions, like the presence of a specific symptom, need not be decided if they are actually not well decidable. Furthermore, values separating the normal from the abnormal range of a parameter can be specified in a rough manner. Finally, the changes of an Arden Syntax program to benefit from these possibilities are minimal.

If the concepts of fuzzy logic are fully applied, there are no more any discontinuities in the output. However, since the inference becomes more sophisticated, also the output is possibly more differentiated. Namely, the user might be presented a list of alternatives together with weights. But this is not to be considered as a price to pay; it is just natural in the present context. When an input parameter is in a borderline area, more than one alternative in the output should be expected.

The work on the Fuzzy Arden compiler is in progress; the compiler will be available in the first quarter of 2010. The application of Fuzzy Arden Syntax in the above-mentioned system Moni/Surveillance-ICU is scheduled and extended practical experiences can be expected in the course of 2010.

References

[1] Health Level Seven. The Arden Syntax for Medical Logic Systems, Version 2.7, Ann Arbor, MI: Health Level Seven, Inc., 2008.

[2] Tiffe S. Fuzzy Arden Syntax: Representation and Interpretation of Vague Medical Knowledge by Fuzzified Arden Syntax, Ph.D. Thesis, Vienna: Technical University Vienna, 2003.

[3] Vetterlein T, Mandl H, and Adlassnig KP. Vorschläge zur Spezifikation der Programmiersprache Fuzzy Arden Syntax (Proposal of a specification of the programming language Fuzzy Arden Syntax – in German), Technical Report, Vienna: Medical University of Vienna, 2008. http://www.meduniwien.ac.at/user/thomas.vetterlein/articles/FuzzyArdenSpezif.pdf.

[4] Vetterlein T, Mandl H, and Adlassnig KP. Fuzzy Arden Syntax: a fuzzy programming language for medicine. Artif Intell Med (2010), doi:10.1016/j.artmed.2010.01.003.

[5] Hospital in Europe Link for Infection Control through Surveillance (HELICS) ed. Surveillance of Nosocomial Infections in Intensive Care Units, Protocol version 6.1; 2004. http://helics.univ-lyon1.fr/protocols/icu_protocol.pdf.

[6] Garner JS, Jarvis WR, Emori TG, Horan TC, and Hughes JM. CDC definitions for nosocomial infections. In: Olmsted RN, ed. APIC Infection Control and Applied Epidemiology: Principles and Practice. St. Louis: Mosby, 1996; pp. A-1–A-20.

[7] Robert-Koch-Institut Berlin, ed., Definitionen nosokomialer Infektionen (CDC-Kriterien). 6th ed. Berlin: MB-Medienhaus Berlin, 2008.

[8] Adlassnig KP, Blacky A, and Koller W. Fuzzy-based mosocomial infection control. In: Nikravesh M, Kacprzyk J, and Zadeh LA, eds. Forging New Frontiers: Fuzzy Pioneers II. Berlin: Springer, 2008; pp. 343-50.

[9] Zadeh LA. Fuzzy sets. Inform Contr 1965: 8: 338-53.

[10] Klement EP, Mesiar R, and Pap E, Triangular Norms. Dordrecht: Kluwer, 2000.

Address for correspondence

Thomas Vetterlein, Section for Medical Expert and Knowledge-Based Systems, Medical University of Vienna, Spitalgasse 23, 1090 Vienna, Austria; E-mail: thomas.vetterlein@meduniwien.ac.at.