Creating Clinical Fuzzy Automata with Fuzzy Arden Syntax

Jeroen S. de Bruin, PhD, MSc[1,2], Heinz Steltzer, PhD, MD[3], Andrea Rappelsberger, MSc[1], Klaus-Peter Adlassnig, PhD, MSc, FACMI, MIAHSI[1,2]
[1]Section for Artificial Intelligence and Decision Support, Center for Medical Statistics, Informatics, and Intelligent Systems, Medical University of Vienna, Vienna, Austria; [2]Medexter Healthcare, Vienna, Austria; [3]Trauma Hospital Vienna South, Vienna, Austria

Abstract

Formal constructs for fuzzy sets and fuzzy logic are incorporated into Arden Syntax version 2.9 (Fuzzy Arden Syntax). With fuzzy sets, the relationships between measured or observed data and linguistic terms are expressed as degrees of compatibility that model the unsharpness of the boundaries of linguistic terms. Propositional uncertainty due to incomplete knowledge of relationships between clinical linguistic concepts is modeled with fuzzy logic. Fuzzy Arden Syntax also supports the construction of fuzzy state monitors. The latter are defined as monitors that employ fuzzy automata to observe gradual transitions between different stages of disease. As a use case, we re-implemented FuzzyARDS, a previously published clinical monitoring system for patients suffering from acute respiratory distress syndrome (ARDS). Using the re-implementation as an example, we show how key concepts of fuzzy automata, i.e., fuzzy states and parallel fuzzy state transitions, can be implemented in Fuzzy Arden Syntax. The results showed that fuzzy state monitors can be implemented in a straightforward manner.

Introduction

By definition an automaton is "an abstract state-determined machine designed to follow automatically a predetermined sequence of operations or respond to encoded instructions". Automata are prime examples of general systems over discrete spaces [1]. The concepts of *state* and *transition* are core aspects of automata theory. The best known class of automata are deterministic finite-state automata, in which systems can be in exactly one of a finite number of states at any given time. These states are labeled using natural language concepts and/or classifications. Transitions between system states (i.e., linguistic concepts) are triggered by an external input, and may be presented as a function mapping the current state and the input into the next state [1].

Although the concept of a discrete state is attractive due to its simplicity in terms of implementation, it is of limited use for modeling situations and contexts where discrete states cannot be accurately defined using natural language concepts, or where knowledge about the situation is imprecise or incomplete. For instance, an individual's health cannot be accurately classified by discrete, mutually exclusive concepts at the sole end of the spectrum, such as *healthy* or *sick*, nor would it be practical to introduce a myriad of states between these spectrum poles. To address these shortcomings in classification with linguistic concepts, Zadeh introduced fuzzy sets and fuzzy logic [2]. With fuzzy sets, the relationship between linguistic terms and measured or observed data is expressed as a degree of compatibility (DoC) calculated by fuzzy membership functions rather than a discrete, dichotomous classification. In this context, a DoC formally models the unsharpness of the boundaries of linguistic terms. Propositional uncertainty due to incomplete knowledge of relationships between clinical linguistic concepts is modeled with fuzzy logic, allowing for approximate reasoning instead of exact rule inference.

The incorporation of fuzzy sets and fuzzy logic into automata gives rise to the notion of fuzzy automata, which are able to handle uncertainty and continuous space [3,4]. With fuzzy automata, a system can be in more than one state at the same time, whereby the applicability (or membership) of each state is expressed as a DoC. Because of this multi-state membership, multiple simultaneous transitions of state are also possible. The system as a whole is far more expressive because the potentially endless combinations of state values (i.e., applicability of linguistic concepts) can imply richer, more abstract linguistic concepts. Furthermore, multiple (partial) parallel transitions cause transitions between linguistic concepts to be more gradual and thus more intuitive.

Given the properties of fuzzy automata discussed above, they are well suited for the use in (automated) clinical monitors [5]. In clinical monitoring, streams of patient data are measured in (occasionally brief) time intervals and presented to the clinician. Given the increasing body of patient data being measured in various time intervals, the monitor is growing more complex, both in terms of the number of presented data elements as well as the interpretation of combinations of different data elements. In other words, the more powerful a monitor becomes (from a functional perspective), the greater is the risk of errors of omission in the interpretation of data. By using a fuzzy automaton in

clinical monitoring for the interpretation and aggregation of measured patient data over time, the dimensionality of input data can be reduced and presented in semantically meaningful, clinically relevant linguistic concepts over fixed time intervals, along with a gradual indication of their applicability to the patient.

Standardized technical communication with hospital information systems and electronic health records is an equally important element of meaningful semantic communication with users. The acceptance and dissemination of the aforementioned monitoring systems could be improved by employing known communication standards, especially when the systems are to be embedded into existing clinical monitors. A widely used standard for computerized knowledge representation and processing is Arden Syntax [6], which is a programming language for the collection, description, and processing of medical knowledge in a machine-executable format. Since Arden Syntax version 2.9 was augmented by formal constructs based on fuzzy set theory and fuzzy logic, it is now possible to implement a fuzzy automaton in a clinical system using a standard that intrinsically supports fuzzy methods.

In this paper, we report our preliminary experience concerning the implementation of a clinical fuzzy automaton using Arden Syntax 2.9 (Fuzzy Arden Syntax). As a use case we re-implemented a previously published monitoring system named FuzzyARDS [7], which is a clinical monitoring system for patients suffering from acute respiratory distress syndrome (ARDS) and is based on the DiaMon-1 framework [8]. For easier comprehension, we provide an overview of Arden Syntax and the fuzzy methods incorporated in Fuzzy Arden Syntax, as well as a short description of the DiaMon-1 framework and the FuzzyARDS monitor. Then, using the re-implementation of parts of the FuzzyARDS monitor (referred to henceforth as *FuzzyArdenARDS*) as an example, we show how key concepts of fuzzy automata, such as fuzzy states and parallel fuzzy state transitions can be implemented in Fuzzy Arden Syntax.

Methods

Arden Syntax

Arden Syntax is a standard for computerized medical knowledge representation and processing. Arden Syntax knowledge bases are fragmented into medical logic modules (MLMs), which are collections of medical rules and knowledge for the purpose of making at least one medical decision [9]. Several properties make Arden Syntax well suited for the computerized representation of medical knowledge [10]. In fact, the program code in Arden Syntax resembles natural language. Thus MLMs are understood more easily by healthcare professionals. Moreover, pure medical knowledge is separated from more technical processing, which improves code transparency. Finally Arden Syntax supports various data types specific to medical documentation, such as time and duration types.

A complete overview of Arden Syntax is beyond the scope of this paper. However, for comprehension of the examples presented in this paper we will provide a short overview of Fuzzy Arden Syntax. For a complete description we refer to the Arden Syntax version 2.9 specification [6].

Each Arden Syntax MLM is hierarchically structured. At the top level an MLM is divided into the categories of maintenance, library, knowledge, and resources. Each of these categories contains category-specific slots. Slots in the maintenance category enable the MLM author to provide metadata on the MLM, such as the MLM title, author, or version. Slots in the library category provide contextual information about the MLM, such as the purpose of the MLM, an explanation of its functionality, and evidence-based resource citations. The medical knowledge of the MLM is implemented in the knowledge category. In the data slot, MLM parameters can be assigned to variables, and data from external sources can be obtained through curly braces expressions. The MLM's program logic is implemented in the logic slot. Apart from implementing logic, processing can also be deferred through the invocation of other MLMs. Execution ends with a concluding statement which, if considered true, results in the execution of the action slot. Finally, localized messages can be constructed through the optional definition of the resources category.

Fuzzy Arden Syntax

Since version 2.9, the Arden Syntax supports formal constructs based on fuzzy set theory and fuzzy logic. We will present a selection of fuzzy extensions implemented in Fuzzy Arden Syntax in this section. Note that this is not a complete overview of all fuzzy concepts implemented in Fuzzy Arden Syntax; for a detailed description of fuzzy constructs implemented in Fuzzy Arden Syntax we refer to previously published work [11].

The truth value model of Arden Syntax was expanded in order to extend traditional methods of calculation and logic. Prior to version 2.9, the truth value model was dichotomous, supporting only the values "true" and "false". However, truth values in Fuzzy Arden Syntax may now constitute any value in the range [0, 1]. True and false still exist in this model as the extremes of the range: "false" has a truth value of 0, whereas "true" has a truth value of 1. Based on this

truth value model, Fuzzy Arden Syntax incorporates fuzzy set data types, built-in propositional fuzzy logic operators, and degrees of applicability of conditional branches.

The fuzzy set data type can be used to model and quantify the unsharpness of boundaries in definitions of linguistic concepts. A fuzzy set is declared with two or more value tuples which define the fuzzy region(s). From these tuples, a linear membership function is constructed for the fuzzy set, which is used to calculate the DoC of measured or observed data with respect to the clinical linguistic concept under consideration.

With fuzzy logic operators, propositional uncertainty in relationships between linguistic clinical concepts can be modeled implicitly. Three basic propositional fuzzy logic operations are implemented in Fuzzy Arden Syntax – negation, conjunction, and disjunction. These are equipped to handle all truth values in the extended truth value model. By default, negation of a concept is implemented as 1 minus the truth value of that concept. The standard minimum function is used as the fuzzy conjunction operator, and the standard maximum function is used as the fuzzy disjunction operator.

The new truth value model called for an extension of the evaluation mechanism for conditional branches. Whereas only one conditional branch was executed at a time in the traditional model, in the extended model each branch whose condition amounts to non-zero is executed in parallel. To this end, all simple data types have a degree of applicability, which is set to 1 by default. However, when the program execution splits into multiple parallel branches, each branch is provided with its own set of duplicated variables, and each variable is assigned a degree of applicability which is equal to the relative truth value of the respective branch's conditional expression. After execution of all conditional branches, variable values can either be aggregated or not. When the branches are aggregated (through the aggregate keyword at the end of the conditional block), duplicated variables are joined using a weighted average (applicability multiplied by the variable value). If not aggregated, the MLM will conclude with multiple return values, each with its own applicability.

As an example, consider the Fuzzy Arden Syntax MLM below in which we use fuzzy sets to classify the severity of ARDS. For the sake of brevity, we have confined the example to the knowledge category.

```
maintenance:   […]
library:       […]
knowledge:
  type:        data_driven;;
  data:        (pao2, fio2) :=  argument;; // blood gas and inspiration
  priority:    ;;
  evoke:       ;;
  logic:
    // Fuzzy set definitions
    ARDS_severe := fuzzy set (100,1), (110,0);
    ARDS_moderate := fuzzy set (100,0), (110,1), (190,1), (200,0);
    ARDS_mild := fuzzy set (190,0), (200,1), (300,1), (310,0);

    // Parameter analysis
    if ((pao2 / fio2) is in ARDS_severe) then
      msg := "Patient suffers from severe ARDS.";
    elseif ((pao2 / fio2) is in ARDS_moderate) then
      msg := "Patient suffers from moderate ARDS.";
    elseif ((pao2 / fio2) is in ARDS_mild) then
      msg := "Patient suffers from mild ARDS.";
    endif;

    // Program conclusion
    conclude true;;
  action: return msg;;
  urgency: ;;
end:
```

The logic in this MLM is based on the most recent 'Berlin Definition' of ARDS published in 2012 [12]. According to this definition, the severity of ARDS is determined by the ratio of partial arterial oxygen pressure (PaO2) and the fraction of inspired oxygen (FiO2). Based on the resulting outcome, ARDS is characterized as either mild (200 < PaO2/FiO2 ≤ 300), moderate (100 < PaO2/FiO2 ≤ 200), or severe (PaO2/FiO2 ≤ 100). However, patients with measured values close to these thresholds are also of interest. As such, we created fuzzy sets for all characterizations that extend beyond the defined thresholds (lines 10-12). Next, the PaO2/FiO2 ratio is compared with each fuzzy set, and the resulting truth values serve as conditions in conditional branches (lines 15-21). In case multiple truth values are non-zero, each of these conditional branches is executed. Furthermore, as the branches are not aggregated, this would cause the MLM to return multiple copies of *msg*, each with its own degree of applicability. For example, if PaO2/FiO2 were 106, this would cause the "severe ARDS" message to be returned with an applicability of 0.4 and the "moderate ARDS" message to be returned with an applicability of 0.6. This could be interpreted as a patient's ARDS being between the "moderate" and "severe" state.

FuzzyArdenARDS and DiaMon-1

The FuzzyArdenARDS application was re-implemented based on the original FuzzyARDS automaton, which was constructed using the DiaMon-1 framework as discussed in [7,8]. This formal framework was developed in order to design monitors capable of abstracting continuously supplied, objectively observed raw data into aggregated, qualitative, linguistic concepts, such as stages of disease. Furthermore, monitors developed with this framework provide early indication of improvement in, or deterioration of, a patient's health status because the monitors include smooth transitions between stages.

Applications in the DiaMon-1 framework are referred to as *state monitors*, which employ fuzzy automata to observe gradual transitions between different stages of disease. In this context, a state represents a (linguistic) representation of a patient's health status or a specific stage of disease. Transitions provide possible pathways between states or stages, which are triggered by inputs such as time or measured data.

A fuzzy state monitor *SM* is formally defined as a 6-tuple $\overline{SM} = (Q, \tilde{q}_0, X, \delta, P, f)$. In this definition, the first four parameters jointly constitute the underlying fuzzy automaton: Q denotes a finite set of states, $\tilde{q}_0$is a (potentially) fuzzy subset of Q that marks the initial state, X is a finite set of input symbols, while $\delta: Q \times X \rightarrow Q$ is a transition function that maps states and inputs onto states. Furthermore, P is the parameter value space over all observed parameters: $p_1 \times \ldots \times p_n$, and f is a mapping function that maps parameter tuples from P to a fuzzy subset of X.

Calculation of the monitor state at time point t ($\tilde{q}_t$) proceeds through an inductive function based on $\tilde{q}_{t-1}$, δ, and f. Suppose that at time point t, f yields fuzzy subset $\widetilde{fs}_t$, which is a collection of truth values for all $x \in X$. Then, using the extension principle [13], the state for each $q \in Q$ at time point t, $\tilde{q}_t(q)$, can be calculated as follows:

$$\tilde{q}_t(q) = \begin{cases} \bigvee \{\tilde{q}_{t-1}(q') \wedge \widetilde{fs}_t(x) \mid \delta(q', x) = q, q' \in Q, x \in X\} & if\ \delta^{-1}(q) \neq \emptyset \\ 0 & if\ \delta^{-1}(q) = \emptyset \end{cases}$$

For the FuzzyArdenARDS application, the automaton states Q are shown in Table 1.

Table 1. Definition of the fuzzy automaton states in FuzzyArdenARDS.

State	**Description**
Start	Initial state
Normal	Oxygenation is satisfactory, no additional effort needed.
Hypoxic	Oxygenation is too low.
Responding to high FiO2	Oxygenation was positively affected by high FiO2.
Not responding to high FiO2	High FiO2 did not have a desired effect.
Improved after hand bagging	Manual oxygenation through hand bagging has improved oxygenation.
Not improved after hand bagging	Hand bagging did not have the desired effect.

Note: FiO2, fraction of inspired oxygen.

As initial state $\tilde{q}_0$, the truth value for the *Start* state is 1, and 0 for all others. The set of input symbols X comprises {*adequate oxygenation, hypoxemia, high FiO2, low FiO2, rapidly improving oxygenation, slowly decreasing oxygenation*}. The set of fuzzy state transition rules δ is shown in Table 2.

Table 2. Definition of the fuzzy state transition rules in FuzzyArdenARDS.

Rule	**Begin state**	**Transition condition**	**End state**
1	Start	Adequate oxygenation	Normal
2	Start	Hypoxemia	Hypoxic
3	Normal	Hypoxemia	Hypoxic
4	Hypoxic	Low FiO2 ∧ Adequate oxygenation	Normal
5	Hypoxic	High FiO2 ∧ rapidly improving oxygenation	Responding to high FiO2
6	Hypoxic	High FiO2 ∧ hypoxemia	Not responding to high FiO2
7	Responding to high FiO2	Low FiO2 ∧ slowly decreasing oxygenation	Improved after hand bagging
8	Responding to high FiO2	Low FiO2 ∧ hypoxemia	Not improved after hand bagging
9	Not responding to high FiO2	Low FiO2 ∧ hypoxemia	Hypoxic
10	Not responding to high FiO2	High FiO2 ∧ adequate oxygenation	Responding to high FiO2
11	Improved after hand bagging	Adequate oxygenation	Normal
12	Improved after hand bagging	Hypoxemia	Hypoxic
13	Not improved after hand bagging	Hypoxemia	Hypoxic

Note: FiO2, fraction of inspired oxygen; ∧, fuzzy logical conjunction operator (minimum function).

Three parameters have been considered for P: time, oxygen saturation SaO2 (as a noninvasive alternative to measuring PaO2), and FiO2. Finally, rules for fuzzy sets based on time, SaO2 and FiO2, which jointly constitute f are presented in Table 3.

Table 3. Definition of the fuzzy automaton parameter-to-input symbol mapping in FuzzyArdenARDS.

Input symbol	Rule (Start of fuzzy region)
Adequate oxygenation	SaO2 above 97% (93%) for 5 minutes
Hypoxemia	SaO2 between 90% (87%) and 93% (97%) for 2 minutes
High FiO2	FiO2 above 60% for 30 seconds
Low FiO2	FiO2 below 60% for 30 seconds
Rapidly improving oxygenation	SaO2 increasing from 87–95% (85–99%) to 97–100% (93–100%) within 30–90 seconds
Slowly decreasing oxygenation	SaO2 above 96% (91%) steady or decreasing to 94% (89%) within 25 minutes

Note: SaO2, oxygen saturation; FiO2, fraction of inspired oxygen.

Data processing

The MLMs discussed in this report were created with the ARDENSUITE software [14,15]. The ARDENSUITE is a framework for medical knowledge representation and reasoning, which comprises an integrated development and test environment (IDE) and an ARDENSUITE server, including software modules for interconnecting with data sources (Figure 1).

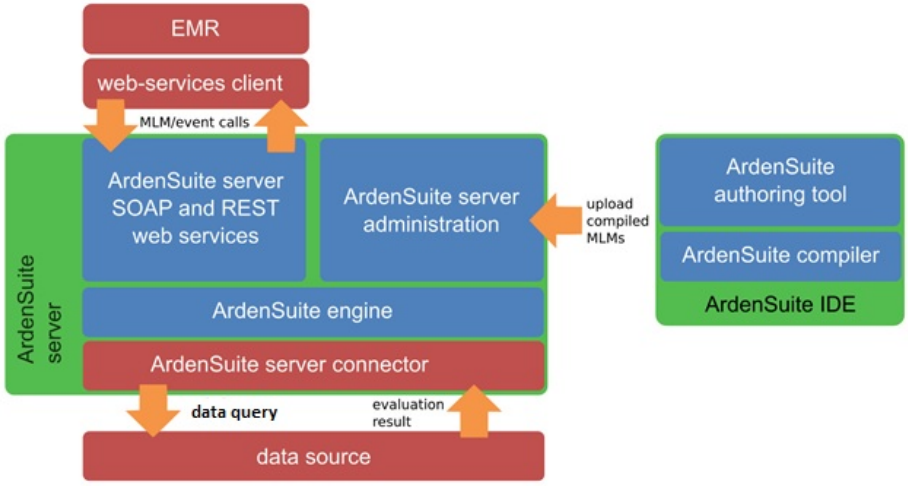

Figure 1. Graphic depiction of the ARDENSUITE framework. Image adapted from [15].

With the ARDENSUITE IDE, users can write and compile MLMs via the authoring tool. If test data are available, users can also immediately test the implemented MLMs. After compilation, the MLMs are uploaded to the administration module of the ARDENSUITE server. The administration module is a management tool for compiled Arden Syntax projects and supports functionalities such as the activation or deactivation of MLMs in an application, or MLM version management. The core element of the ARDENSUITE server is the ARDENSUITE engine, which executes compiled MLMs. To facilitate access to MLM functionalities by arbitrary clients, the server provides service-oriented access through a web-service component. Using this component, MLM calls and data exchange are facilitated through the Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) web-service standards. Web-service standards can also be used to connect the ARDENSUITE server with SOAP/REST-compatible external database sources through the ARDENSUITE server connector. With this module, external data sources can be accessed directly from within MLM files using query languages (such as SQL), which are then forwarded to the source data base management system using SOAP or REST web services.

Results

In this section, we discuss parts of the resulting MLM implementation of FuzzyArdenARDS. We implemented the core of the FuzzyArdenARDS application in two MLMs. In the first MLM, fuzzy sets are defined and truth values calculated using the SaO2 and FiO2 parameters. Each parameter is supplied as a list of values over the last half hour. The MLM is designed for semi-real-time processing, and is called every 30 seconds, thus analyzing 30 seconds of data at a time. Again, for the sake of brevity, we have confined the MLMs listed below to the knowledge category.

```
maintenance:   […]
library:       […]
knowledge:
  type:        data_driven;;
  data:        (sao2, fio2):= argument;;
  priority:    ;;
  evoke:       ;;
  logic:
    // Fuzzy set definitions
    fs_adeq_oxy := fuzzy set (0.93,0), (0.97,1);
    fs_hypoxemia := fuzzy set (0.87,0), (0.9,1), (0.93, 1), (0.97,0);
    fs_rio_begin := fuzzy set (0.85,0), (0.87,1), (0.95, 1), (0.99,0);
    fs_rio_end := fuzzy set (0.93,0), (0.97,1);
    fs_sdo_begin := fuzzy set (0.91,0), (0.96,1);
    fs_sdo_end := fuzzy set (0.89,0), (0.94,1);

    // Parameter analysis
    sao2_5mins := sao2 where they occurred within the past 330 seconds;
    ade_oxy := minimum (sao2_5mins in fs_adeq_oxy);

    sao2_2mins := sao2 where they occurred within the past 150 seconds;
    hypoxemia := minimum (sao2_2mins in fs_hypoxemia);

    fio2_30secs := fio2 where they occurred within the past 60 seconds;
    high_fio2 := minimum (fio2_30sec) > 60;
    low_fio2 := maximum (fio2_30sec) < 60;

    sao2_30secs := sao2 where they occurred within the past 30 seconds;
    for sao2_element in sao2_30secs do
      reference_list := sao2 where
        ((time of sao2_element) - (time of them)) is within 30 seconds to 90 seconds;
      for sao2_ref_element in reference_list do
        lmax_tv := sao2_ref_element is in fs_rio_begin and sao2_element is in fs_rio_end;
        rio := maximum (rio, lmax_tv);
      enddo;
    enddo;

    sdo := 0;
    for sao2_element in sao2_30secs do
      reference_list := sao2 where
        ((time of sao2_element) - (time of them)) is not greater than 25 minutes;
      for sao2_ref_element in reference_list do
        lmax_tv := sao2_ref_element is in fs_sdo_begin and sao2_element is in fs_sdo_end;
        sdo := maximum (sdo, lmax_tv);
      enddo;
    enddo;
    conclude true;;
```

In the above MLM code snippet, truth values are calculated for each of the symbols in *X*. First, all fuzzy sets are defined (lines 10-15). Then the truth value is calculated for the symbol *adequate oxygenation*. For this purpose, we first need to obtain SaO2 values for the last 5½ minutes; 30 seconds for evaluation and 5 minutes thereafter to calculate the truth value for all data in the evaluation phase according to the mapping in Table 3 (line 18). We then apply the data to the defined fuzzy set and take the minimum truth value as an aggregate result for the last 30 seconds (line 19). In a similar fashion, truth values are calculated for the symbols *hypoxemia* (lines 21-22), *high FiO2*, and *low FiO2* (lines 24-26). For the symbol *rapidly improving oxygenation*, a slightly different approach is needed. First, data elements for the first 30 seconds for evaluation are isolated (line 28). For each of the data elements, we then select prior elements or data elements within a period of 30-90 seconds before the timestamp of the analyzed elements (line 29-31). Data elements of both lists are applied pairwise to their respective fuzzy sets, and a logical conjunction of each pair is calculated (line 32-33). Finally, the maximum of all pairwise logical conjunctions is chosen as an aggregate value for the symbol *rapidly improving oxygenation* (line 34). In a similar fashion, a truth value is calculated for the symbol *slowly decreasing oxygenation* with a time period of 25 minutes.

The second MLM is used to perform the transitions. As parameters, a list of truth values for each state and a list of truth values for each input symbol are provided. Due to space constraints, we only show that part of the MLM implementation dealing with the transitions for the end states *Start*, *Normal*, and *Hypoxic*.

```
1   maintenance:   […]
2   library:       […]
3   knowledge:
4     type:  data_driven;;
5     data:
6       // States (array indices) enumerations:
7       AutomStates := OBJECT [Start, Normal, Hypoxic, RespHighFiO2,
8         NotRespHighFiO2, ImpAfterHandBagging, NotImpAfterHandBagging];
9       states := new AutomStates with 1 seqto 7;
10
11      // Inputs (array indices) enumerations:
12      AutomInputs := OBJECT [AdequateOxy, Hypoxemia, HighFiO2,
13        LowFiO2, RapidImpOxygenation, SlowDecOxygenation];
14      inputs := new AutomInputs with 1 seqto 6;
15
16      (state_tvs, input_tvs) := Argument;;
17
18    logic:
19      // Start state has no incoming transitions
20      state_start := 0;
21
22      // Normal state has three incoming transitions:
23      state_normal :=
24        // Rule 1: Start and adequate oxygenation
25        (state_tvs[states.Start] as truth value and
26         input_tvs[inputs.AdequateOxy] as truth value)
27        or
28        // Rule 4: Hypoxic and low FiO2 and adequate oxygenation
29        (state_tvs[states.Hypoxic] as truth value and
30         input_tvs[inputs.LowFiO2] as truth value and
31         input_tvs[inputs.AdequateOxy] as truth value)
32        or
33        // Rule 11: Improved after hand bagging and adequate oxygenation
34        (state_tvs[states.ImpAfterHandBagging] as truth value and
35         input_tvs[inputs.AdequateOxy] as truth value);
36
37        // Hypoxic state has five incoming transitions:
```

```
38        state_hypoxic :=
39          // Rule 2: Start and hypoxemia
40          (state_tvs[states.Start] as truth value and
41           input_tvs[inputs.Hypoxemia] as truth value)
42          or
43          // Rule 3: Normal and hypoxemia
44          (state_tvs[states.Normal] as truth value and
45           input_tvs[inputs.Hypoxemia] as truth value)
46          or
47          // Rule 9: Not responding to high FiO2 and low FiO2 and hypoxemia
48          (state_tvs[states.NotRespHighFiO2] as truth value and
49           input_tvs[inputs.LowFiO2] as truth value and
50           input_tvs[inputs.Hypoxemia] as truth value)
51          or
52          // Rule 12: Improved after hand bagging and hypoxemia
53          (state_tvs[states.ImpAfterHandBagging] as truth value and
54           input_tvs[inputs.Hypoxemia] as truth value)
55          or
56          // Rule 13: Not improved after hand bagging and hypoxemia
57          (state_tvs[states.NotImpAfterHandBagging] as truth value and
58           input_tvs[inputs.Hypoxemia] as truth value);
59
…
…
```

Truth values for the seven states in *Q* are calculated in the MLM shown above. First, to improve MLM readability we constructed objects that enumerate the indices of the state and input value lists supplied through the argument (lines 6-16). The calculation of truth values for each state starts thereafter. By definition, the *Start* state is only true at the first iteration of the automaton, followed by 0, as it has no incoming transitions. For each other state, the truth value is determined by a logical disjunction over all rules in Table 2 that have the respective state as an end state. The truth value for each rule, on the other hand, is calculated by a logical conjunction over the truth values of the state and transition conditions. As such, for the *Normal* state a logical disjunction is calculated over transition rule 1 (lines 25-26), rule 4 (lines 29-31), and rule 11 (lines 34-35). The truth values for the other states are calculated in a similar fashion.

Discussion

In the present report, we showed how fuzzy state monitors as defined in [8] could be implemented using fuzzy methods supported by Fuzzy Arden Syntax, a standard for computerized knowledge representation and processing. When designing a clinical knowledge base, knowledge engineers work closely together with clinicians to construct the rules. However, the translation process from natural language to a computerized knowledge representation may be prone to error. The average clinician may be unable to, or not interested in, the validation of source code. The original FuzzyARDS program was written in a dialect of the object-oriented language Smalltalk, which is neither considered a mainstream medium nor is easily understood by those untrained in its application. Given that Arden Syntax rules closely resemble natural language, clinicians can verify the implemented knowledge in MLMs (more or less) easily without in-depth knowledge of modern programming languages.

Within the context of fuzzy state monitors, we introduced fuzzy sets as mapping functions that map raw data to clinical linguistic concepts, thereby yielding a degree of compatibility between 0 and 1. Based on these degrees, together with degrees of applicability for each state in a fuzzy automaton, we showed that multiple transitions could occur simultaneously, resulting in a new automaton configuration comprising (again) degrees of applicability for each state in the fuzzy automaton.

The interpretation of the automaton results is an important aspect. It was not discussed here because it would exceed the scope of this report and is dependent on individual applications. In the case of FuzzyArdenARDS, six parameters need to be interpreted in a pairwise manner: *oxygenation state* (normal vs. hypoxic), *response to high FiO2* (response vs. no response), and *response to hand bagging* (improvement vs. no improvement). Given that values for each of

these states can be a DoC, a richer, a gradual interpretation follows from these pairwise interpretations. For instance, when the “normal” state has a DoC of 0.3 and the “hypoxic” state a DoC of 0.7, the patient’s current oxygenation state may be interpreted as moderately hypoxic.

The limitations of the present report are worthy of note. First, the study is limited to a single application, namely FuzzyArdenARDS. Other fuzzy state monitors and even other types of fuzzy automata need to be implemented with Fuzzy Arden Syntax to ensure the syntax is equipped to support a variety of fuzzy applications and methodologies. Second, the fuzzy sets that we implemented for the FuzzyArdenARDS automaton were “two-dimensional”. In other words, they only implemented fuzzy regions for one parameter. As some rules were defined over two parameters (SaO2 and a time duration), three-dimensional fuzzy sets that provide more accurate modeling for these rules will have to be implemented in the future. Finally, as of yet the program has only been tested on retrospectively collected data.

We reported on the first steps in implementing fuzzy state monitors with Fuzzy Arden Syntax. In the future, we plan to study and address the aforementioned limitations and continue to improve the use of Arden Syntax for real-time monitoring.

Conclusion

The native support of fuzzy methods allows the intuitive implementation of a fuzzy state monitor for clinical application with Fuzzy Arden Syntax.

References

1. Gaines BR, Kohout LJ. The logic of automata. Int J Gen Syst. 1976;2(4):191-208.
2. Zadeh LA. Fuzzy sets. Inform Control. 1965;8(3):338-53.
3. Klir G, Yuan B. Fuzzy sets and fuzzy logic: Theory and applications. 1st ed. New Jersey: Prentice Hall; 1995.
4. Doostfatemeh M, Kremer SC. New directions in fuzzy automata. Int J Approx Reason. 2005;38(2):175-214.
5. Adlassnig K-P. A survey on medical diagnosis and fuzzy subsets. In: Gupta MM, Sanchez E, editors. Approximate reasoning in decision analysis. Amsterdam: North-Holland Publishing Company; 1982. p. 203-17.
6. Health Level Seven International. HL7 Arden V2.9-2013: The Arden Syntax for Medical Logic Systems Version 2.9. [Internet]. 2013 [cited 2017 Mar 6]. Available from: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=290
7. Steltzer H, Trummer B, Höltermann W, Kolousek G, Fridrich P, Lewandowski K, Adlassnig K-P, Hammerle AF. Wissensbasierte Diagnostik und Therapieempfehlung mit Methoden der Fuzzy-Set-Theorie bei Patienten mit akutem Lungenversagen (ARDS). Anasthesiol Intensivmed Notfallmed Schmerzther. 1999;34(4):218-23.
8. Steimann F, Adlassnig K-P. Clinical monitoring with fuzzy automata. Fuzzy Sets Syst. 1994;61(1):37-42.
9. Hripcsak G. Writing Arden Syntax medical logic modules. Comput Biol Med. 1994;24(5):331-63.
10. Samwald M, Fehre K, de Bruin J, Adlassnig K-P. The Arden Syntax standard for clinical decision support: Experiences and directions. J Biomed Inform. 2012;45(4):711-8.
11. Vetterlein T, Mandl H, Adlassnig K-P. Fuzzy Arden Syntax: A fuzzy programming language for medicine. Artif Intell Med. 2010;49(1):1-10.
12. The ARDS Definition Task Force*. Acute respiratory distress syndrome: The Berlin definition. JAMA. 2012;307(23):2526-33.
13. Zadeh LA. The concept of a linguistic variable and its application to approximate reasoning. In: Fu KS, Tou JT, editors. Learning systems and intelligent robots. Boston, MA: Springer US; 1974. p. 1-10.
14. Medexter Healthcare. ArdenSuite – Medical Knowledge Representation and Rule-Based Inference Software with Arden Syntax. [Internet]. 2015 [cited 2017 Mar 6]. Available from: http://www.medexter.com/component/jdownloads/send/3-public-articles/6-ardensuite-for-emrs
15. Adlassnig K-P, Fehre K. Service-Oriented Fuzzy-Arden-Syntax-Based Clinical Decision Support. Indian J Med Inform. 2014;8(2):75-9.