УДК 004.021

РАЗРАБОТКА АЛГОРИТМА УПРАВЛЕНИЯ ТЕМПЕРАТУРНЫМ РЕЖИМОМ ДЛЯ ИНТЕЛЛЕКТУАЛЬНОЙ ЭНЕРГОСБЕРЕГАЮЩЕЙ СИСТЕМЫ МНОГОКВАРТИРНЫХ ЗДАНИЙ И СЛОЖНЫХ СООРУЖЕНИЙ

DEVELOPMENT OF THE ALGORITHM TEMPERATURE CONTROL SYSTEM FOR INTELLIGENT ENERGY-EFFICIENT APARTMENT BUILDINGS AND COMPLEX STRUCTURES

Морозова Татьяна Юрьевна
доктор техн. наук, профессор

Акимов Дмитрий Александрович
кандидат техн. наук, ведущий инженер

Кашкин Евгений Владимирович
ст. преподаватель

Работкин Владимир Дмитриевич
Аспирант.
Московский государственный университет приборостроения и информатики
Тел.: 8(909) 636-41-56
set@id-yug.com

T.Y. Morozova
Ph. D., Professor

D.A. Akimov
Ph. D., chief engineer

E.V. Kashkin
senior lecturer

V.D. Rabotkin
graduate student.
Moscow State University of Instrument Engineering and Computer Science
Tel.: 8(909) 636-41-56
set@id-yug.com

Аннотация. В статье представлен алгоритм управления температурным режимом для интеллектуальной энергосберегающей системы многоквартирных зданий и сложных сооружений.

Для программной реализации была выбрана платформа Microsoft.NET Framework 4.0. Программирование проводилось на языке C#. Среда разработки Microsoft Visual Studio 2010.

Разрабатывается база знаний, которая основана на базе данных, состоящей из нескольких связанных между собой таблиц и системы логического вывода, на основе резолюций.

Программа предназначена для управления системой задвижек тепловых контуров с помощью выдачи управляющей команды. Система принятия решения основывается на информации о текущей ситуации и призвана обеспечить энергосберегающий эффект при её использовании в системах управления температурным режимом промышленных объектов.

Ключевые слова: нейронная сеть, база знаний, нечеткая интерпретация, нечеткая импликация, логический вывод, продукционная система.

Annotation. The paper presents an algorithm for thermal management for intelligent energy-saving system of apartment buildings and complex structures.

For a software project has been selected platform Microsoft.NET Framework 4.0, in the programming language C #. Development environment Microsoft Visual Studio 2010.

Developing a knowledge base, which is based on a database consisting of several interrelated tables and inference systems based on the resolutions.

The program is designed for system control valves thermal system by issuing a control command. Decision-making system based on the information on the current situation and is designed to provide energy-saving effect when used in control systems for temperature control of industrial plants.

Keywords: neural network, knowledge base, fuzzy interpretation, fuzzy implication, inference, production system.

Программное обеспечение включают в себя следующие подсистемы. Интеллектуальное ядро системы управления и принятия решения на основе:

– обратимой нейронной сети и распределенной базы знаний, развернутая как клиент-серверная архитектура на облачном web-сервисе, которое обеспечивает

надёжность, простоту адаптации под новые условия и возможность дистанционно контролировать и настраивать любые параметры системы;

– статической базы знаний и программно-управляющего модуля в составе микроконтроллера, которые обеспечивают непосредственную выдачу команды управления заслонкой.

Функциями локального сервера и установленного программного обеспечения является обучение автономной программы на ПЗУ микроконтроллера. Облачная база знаний содержит интеллектуальное ядро анализа и самообучения на основе нечеткой нейронной сети.

База знаний микроконтроллера периодически должна обновляться за счет локального сервера.

В основе построения программной системы лежит метод искусственного интеллекта, основанный на использовании базы нечётких знаний, подсистемы логического вывода, адаптации и самообучения. Применение разработанной методики, позволяет управлять параметрами температурного режима в условиях неопределённости и быстроменяющейся внешней обстановки таких важных промышленных объектов как: промышленные предприятия, предприятия социальной сферы, многоквартирные здания, тепличные хозяйства, хранилища.

Адаптация системы основана на подстройки коэффициентов атомов правил и фактов. Коэффициенты меняются в зависимости от внешних условий с помощью модифицированного метода группового учета аргументов. Адаптация позволяет системе подстраиваться под текущие условия и выводить систему на более эффективный режим.

Самообучение основано на дописывании в базу знаний непротиворечивых новых правил или удалении старых. Самообучение основано на методе гиперрезолюции [1] с применением нейроподобных структур. Новые знания генерируются посредствам распознавания предыдущих ситуаций нейроподобными структурами и корректируются алгоритмами статистического анализа.

Нечеткие нейросетевые системы – это системы, основанные на нечетких продукционных правилах либо реляционные системы [2]. Нейро-нечеткий классификатор использует нечеткие связи между нейронами. Функционирование и тех, и других систем описывается с помощью композиционных правил нечеткого вывода, в основе которых лежат правила вывода.

Если A и $A \to B$ – выводимые формулы, то B также выводима.

Правило вывода, обычно называемое правилом отделения, позволяет от утверждения условного высказывания $A \to B$ и утверждения его основания A перейти к утверждению следствия B.

Нечеткая интерпретация переменных позволяет перейти к обобщенному правилу вывода.

посылка *если x есть A, то y есть B*

факт *x есть* A^*

заключение *y есть* B^*,

где $A, A^* \in F(X); B, B^* \in F(Y)$ – нечеткие числа, т.е. подмножества универсальных множеств $X \subseteq R$ и $Y \subseteq R$ соответственно.

Заключение B^* определяется на основе операции композиции.

Определим алгоритм нечеткого вывода в виде

$$\forall y \in Y \left(\mu_{B^*}(y) = \sup T \left\{ \mu_{A^*}(x), \mu_{A \to B^{(x,y)}} \right\} \right),$$

при этом T не зависит от оператора импликации, $\mu_{B^*}(y)$ – функция принадлежности.

Поскольку операции композиции и импликация могут быть определены не однозначно и должны быть определенным образом специфицированы, то выбор конкретных представлений определяет некоторый алгоритм, который реализует нечеткий логический вывод. В нечетких продукционных системах база знаний содержит совокупность правил:

R_1 : если x есть A_1, то y есть B_1
R_2 : если x есть A_2, то y есть B_2,
……
R_n : если x есть A_n, то y есть B_n,
x есть A^*
y есть B^*

Каждому правилу R_i соответствует импликация $A_i \to B_i$. Существуют несколько подходов к формированию функции принадлежности заключения. Вначале осуществляется агрегирование (процесс объединения элементов в одну систему) всех правил с помощью подходящего оператора агрегирования (*Agg*), в результате чего получается некоторое обобщенное правило

$$R = Agg(R_1, R_2, ..., R_n),$$

а затем применяется оператор композиции.

После получения нечетких множеств B^* к каждому из них применяется процедура дефазификации (преобразование нечеткого множества в четкое число), после чего осуществляется агрегирование.

Таким образом, процедуры агрегирования используются для:

– агрегирования нечетких продукционных правил в обобщенное правило,

– агрегирования нечетких выходных множеств в обобщенное выходное множество,

– агрегирования дефазифицированных значений выходной переменной, соответствующих каждому из правил.

Целью *фазификации* является установление взаимно однозначного соответствия между конкретным числовым значением и лингвистическим значением некоторой переменной. Фазификация преобразует четкие значения входных переменных в нечеткие множества, которые в дальнейшем наряду с базой правил используются системой нечеткого логического вывода. Это действие можно описать следующим образом:

$$A^* = fazzy(x_0),$$

где x_0 – значение входной переменной X, *fuzzy* – оператор фазификации, A^* – нечеткое подмножество области определения входной переменной X. По сути, фазификация – это процедура перевода числового («четкого») значения x_0 в нечеткий формат. Существуют две возможности, чтобы определить оператор *fuzzy*: каждому x_0 ставится в соответствие функция принадлежности вида

$$\mu_{Ai}(x) = \begin{cases} 1, \text{ если } x = x_0 \\ 0, \text{ если } x \neq x_0 \end{cases}$$

каждому x_0 ставится в соответствие унимодальное нечеткое число треугольного типа с функцией принадлежности

$$\mu_{Ai}(x) = \begin{cases} L(x), \text{ если } x < x_0 \\ 1, \text{ если } x = x_0 \\ R(x_0), \text{ если } x > x_0 \end{cases}$$

Дефазификация устанавливает связь между нечетким множеством и некоторым числовым значением, которое принадлежит области определения функции принадлежности нечеткого множества.

В настоящее время существует большое число методов дефазификации. К основным относятся:

а) метод центра тяжести (CentreofGravity – COG)

$$y = \frac{\int_{\min}^{\max} x \cdot \mu(x) dx}{\int_{\min}^{\max} \mu(x) dx};$$

б) метод центра площади (CentreofArea – COA)

$$y = u : \int_{\min}^{u} \mu(x) dx = \int_{u}^{\max} \mu(x) dx;$$

в) метод левого модального значения (LeftMostMaximum – LM)

$$y = \min\{x_m\},$$

где x_m – модальное значение нечеткого множества.

г) метод правого модального значения (EightMostMaximum – RM)

$$y = \max\{x_m\},$$

где x_m – модальное значение нечеткого множества.

д) метод среднего модального значения (MiddleofMaximum– MOM)

$$y = \frac{\max(x_m) - \min(x_m)}{2},$$

где x_m – модальное значение нечеткого множества.

Метод левого и правого модальных значений предполагает, что значение выходной переменной определяется как мода нечеткого множества для соответствующей выходной переменной или наименьшая (наибольшая) из мод, если нечеткое множество имеет несколько модальных значении.

Необходимость этапа дефазификации обусловлена тем, что в современных системах управления устройства и механизмы способны воспринимать традиционные команды в форме количественных значений соответствующих управляющих переменных.

Выбор метода дефазификации– важный этап проектирования системы нечеткого управлении. Известно, что результат нечеткого логического вывода в значительной степени зависит от метода дефазификации.

Определение нечеткой импликации является одной из важнейших проблем нечеткого моделирования. Часто нечеткую импликацию определяют из ряда предположений и гипотез, которые должны выполняться в рамках данной модели. Так, например, для вербального задания нечеткой системы можно использовать импликацию Брауэра.

Наряду с интуитивным подходом определения импликации, который часто используется в приложениях, существует и аксиоматический подход. Например, можно импликацию рассматривать как нечеткое бинарное отношение *I* на истинностном пространстве [0,1]×[0,1], определяющее свойство импликации $P \rightarrow Q$ между нечеткими высказываниями *P* и *Q.* Вводятся аксиомы, постулирующие свойства импликации и отвечающие интуитивному представлению о природе нечеткого вывода.

Назовем *истинностной* любую функцию истинности, монотонно возрастающую до единицы и *ложностной* – любую функцию, монотонно убывающую от единичного значения, причем при всех положительных значениях принадлежности монотонность строгая.

Нечеткую импликацию можно рассматривать как операцию на [0,1], при этом в точках (0,0), (0,1), (1,0), (1,1) должно выполняться определение четкой импликации.

Выбор оператора импликации для нечетких регуляторов в значительной степени зависит от специфики прикладной задачи.

Выбор импликаций обусловлен тем, что они являются наиболее характерными представителями импликаций различных классов.

Эта взаимосвязь может быть описана наборами правил, являющихся импликациями. Левая часть этих правил представляет собой конъюнкцию подаваемых команд, а правая – команда операционной системе. Для одной и той же команды чаще всего

указывается несколько вариантов конъюнкций распознавания команд. Более того многие авторы [3–5] указывают различные наборы таких правил. Причем эти правила имеют различный уровень точности идентификации.

Программа реализует управление данными. При этом используется специальный алгоритм получения команды управления («Слабо приоткрыть», «Открыть», «Закрыть» и т.д.).

Потоки данных определяются процессом ввода и записи в базу данных информации (рис. 1).

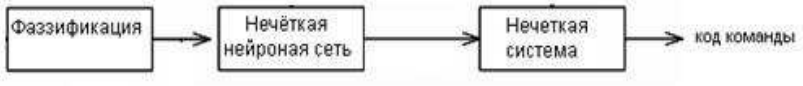

Рисунок 1 – Схема потоков данных

Блок-схема алгоритма программы для микроконтроллера представлена на рисунке 2.

Рисунок 2 – Блок-схема алгоритма программны для микроконтроллера

Для обучения системы используется отдельный алгоритм, представленный на рисунке 3.

Структура облачного WEB-сервиса программного обмена с микроконтроллером показана на рисунке 4.

В разрабатываемой системе в роли решателя выступает WEB-сервис, расположенный на сервере приложений. Данный WEB-сервис отвечает за перебор правил, представленных в базе знаний, а также вывод из последовательности знаний заключения – искомой команды управления, которая должна быть передана на сторону клиента. На вход решателя подаётся команда, являющаяся результатом первичного рас-

познавания команды интеллектуального ввода на стороне клиента. Работа решателя основана на использовании последовательности программных блоков IF-ELSE, в совокупности с базой знаний образующих продукционную систему. Данная продукционная система предполагает стратегию обработки информации «от данных к цели» [6]. При этой стратегии образцом для поиска служит набор условий, передаваемых со стороны клиентской части системы. Процесс поиска решения оканчивается, если в базу данных поступает утверждение, являющееся решением, или выполняется продукционное правило, предписывающее прекращение поиска [7].

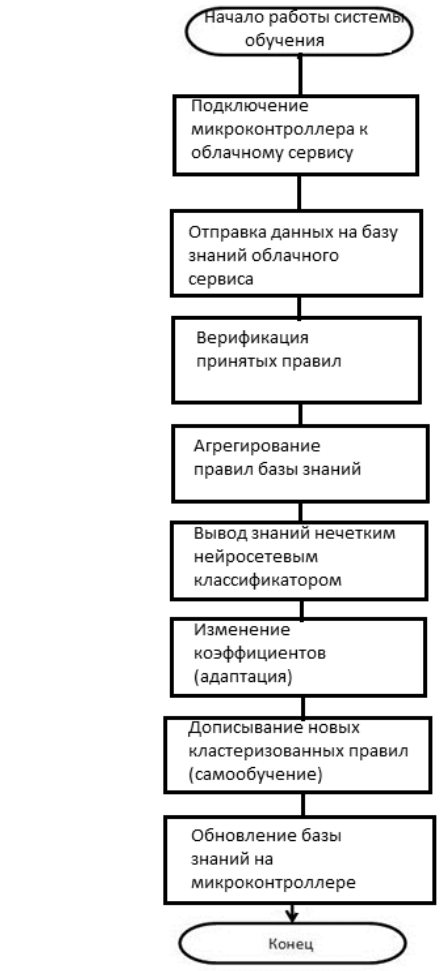

Рисунок 3 – Алгоритм обучения системы

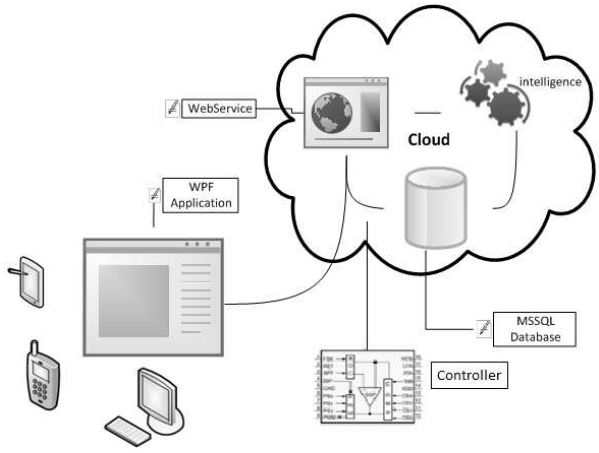

Рисунок 4 – Структура облачного WEB-сервиса программного обмена с микроконтроллером

База знаний в системе основана на базе данных, состоящей из нескольких связанных между собой таблиц и системы логического вывода на основе резолюций [8].

База данных состоит из следующих таблиц:

Климатическая зона – таблица, содержащая данные о среднемесячных суточных температурах для каждой климатической зоны промышленного объекта.

Помещения–таблица, содержащая параметры конкретных помещений промышленного объекта.

Правила – таблица, содержащая правила для базы знаний.

Пользователи – таблица, содержащая данные о пользователях.

Параметры – таблица, содержащая информацию настраиваемых локальных и глобальных параметрах.

Рисунок 5 – Состав программной системы

Основные функции реализуют следующие программные процедуры и функции:

Get_customers_using_Soap12ServiceClient() – Получениепараметровпо протоколу SOAP

Get_customers_using_XmlServiceClient()-ПолучениепараметровчерезXML

Get_customers_using_JsonServiceClient()- ПолучениепараметровJSON

```
usingNUnit.Framework;
usingSakila.ServiceModel.Version100.Operations.SakilaService;
usingSakila.ServiceModel.Version100.Types;
usingServiceStack.Service;
usingServiceStack.ServiceClient.Web;
usingServiceStack.UsageExamples.Support;
namespaceServiceStack.UsageExamples
{
  [TestFixture]
  publicclassUsingServiceClients:TestBase
  {
    [Test]
  publicvoidGet_customers_using_Soap12ServiceClient()//
```

```
Получение параметров SOAP
    {
  using(IServiceClientclient=newSoap12ServiceClient(base.WsSyncReplyUri))
    {
    varrequest=newGetCustomers{CustomerIds=newArrayOfIntId{CustomerId}};
      varresponse=client.Send<GetCustomersResponse>(request);

      Assert.AreEqual(1,response.Customers.Count);
      Assert.AreEqual(CustomerId,response.Customers[0].Id);
    }
  }

    [Test]
    publicvoidGet_customers_using_Soap11ServiceClient()
    {
      using(IServiceClientclient=newSoap11ServiceClient(base.BasicHttpSyncReplyUri))
      {
        varrequest=newGetCustomers{CustomerIds=newArrayOfIntId{CustomerId}};
        varresponse=client.Send<GetCustomersResponse>(request);

        Assert.AreEqual(1,response.Customers.Count);
        Assert.AreEqual(CustomerId,response.Customers[0].Id);
      }
    }

    [Test]
  publicvoidGet_customers_using_XmlServiceClient() // ПолучениепараметровXML
    {
      using(IServiceClientclient=newXmlServiceClient(base.XmlSyncReplyBaseUri))
      {
        varrequest=newGetCustomers{CustomerIds=newArrayOfIntId{CustomerId}};
        varresponse=client.Send<GetCustomersResponse>(request);

        Assert.AreEqual(1,response.Customers.Count);
        Assert.AreEqual(CustomerId,response.Customers[0].Id);
      }
    }

    [Test]
   publicvoidGet_customers_using_JsonServiceClient()//
ПолучениепараметровJSON
    {
      using(IServiceClientclient=newJsonServiceClient(base.JsonSyncReplyBaseUri))
      {
        varrequest=newGetCustomers{CustomerIds=newArrayOfIntId{CustomerId}};
        varresponse=client.Send<GetCustomersResponse>(request);

        Assert.AreEqual(1,response.Customers.Count);
        Assert.AreEqual(CustomerId,response.Customers[0].Id);
      }
    }

  }
}
```

Все обращения возможны через WEB–инерфейс, реализованный на ASP.NET (рис. 6).

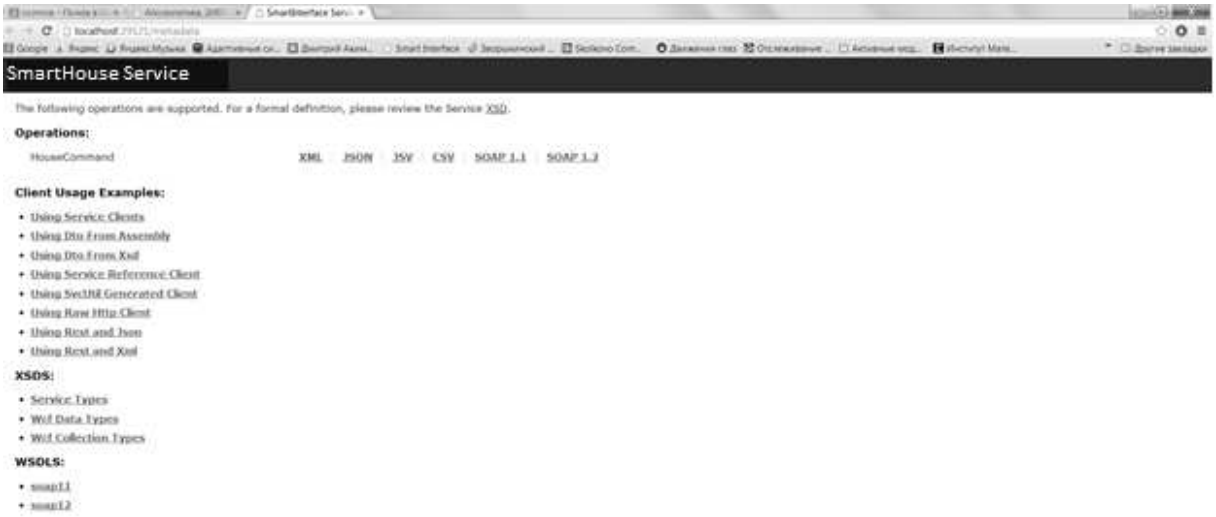

Рисунок 6 – WEB–инерфейс

Передача праметров раелизуется отдельной группой классов [9]. В классе указаны необходимые переменные и методы записи и извлечения параметров в пакет:

```
using System;
usingSystem.Collections.Generic;
usingSystem.Linq;
usingSystem.Runtime.Serialization;
usingSystem.Text;

namespaceCommandsRepository
{
 [DataContract(Namespace = "")]
publicclassSmartHouseProperties
 {
 [DataMember]
publicintClimateZone { get; set; }
 [DataMember]
publicintRoomType { get; set; }
 [DataMember]
publicdecimal Walls { get; set; }
 [DataMember]
publicdecimal Volume { get; set; }
 [DataMember]
publicdecimal Windows { get; set; }
}
}
```

Нечеткий нейросетевой классификатор как набор классов и методов самообучения и адаптации. Нейроподобная структура включает 30 нейронов.

```
typeFuzzyNeuron = classname: string; // Название классо нечеткого нейрона
input: array[0..29,0..29] ofinteger; // входной массив 30x30
output:integer; // выходная команда
memory:array[0..29,0..29] ofinteger; // хранит опыт о предыдущем состоянии
end;
```

При построении сети идет создание массива нейронов, которые образуют структуру нейросетевого классификатора.

```
For i:=0 to 32 do
begin
neuro_web[i]:=Neuron.Create;
neuro_web[i].output:=0; //
end;
ifneuro_web[i].name=s then
begin //В нужном нейроне обновляем память
for x:=0 to 29 do
begin
for y:=0 to 29 do
begin
p.Complex [x,y]:=FuzyF (neuro_web[i].memory[x,y],neuro_web[i].memory[x,y],
neuro_web[i].memory[x,y]); //Записываем новое значение памяти
end;
end;
```

Самообучение реализуется с помощью памяти предыдущего состояния нейрона. Результат передаётся в ПЗУ микроконтроллера. ПЗУ содержит локальную базу правил в шестнадцатеричным формате.

Настройки хранятся в конфигурационном файле, содержащий информацию о подключении к WEB-сервису.

```
<?xmlversion="1.0"?>
<configuration>
  <system.web>
    <httpHandlers>
      <addpath="*"type="ServiceStack.WebHost.Endpoints.ServiceStackHttpHandler
Factory, ServiceStack"verb="*"/>
    </httpHandlers>
    <compilationdebug="true"/></system.web>
  <system.webServer>
    <validationvalidateIntegratedModeConfiguration="false"/>
    <handlers>
      <add-
path="*"name="ServiceStack.Factory"type="ServiceStack.WebHost.Endpoints.ServiceStack
HttpHandlerFactory,      ServiceStack"verb="*"preCondition="integratedMode"resourceType=
"Unspecified"allowPathInfo="true"/>
    </handlers>
  </system.webServer>
  <system.serviceModel>
    <serviceHostingEnvironmentaspNetCompatibilityEnabled="true"/>
    <standardEndpoints>
      <webHttpEndpoint>
        <!--Configure the WCF REST service base address via the global.asax.cs file
and the default endpoint
via the attributes on the <standardEndpoint> element below-->
        <standardEndpoint-
name=""helpEnabled="true"automaticFormatSelectionEnabled="true"/>
      </webHttpEndpoint>
    </standardEndpoints>
  </system.serviceModel>
</configuration>
```

Рынок для реализации разработанного программного обеспечения для энергосберегающей системы управления температурным режимом имеет большое количество разнообразных потребителей. Обычные пользователи получат удобное средство управления климат контролем с обеспечением энергосберегающих функций. Промышленные предприятия, военные, научные центры, тепличные хозяйства, хранилища получат уникальную систему управления различными объектами и процессами.

Литература:

1. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. – М. : Наука, 1983. – 358 с.

2. Нильсон Н.Дж. Принципы искусственного интеллекта. – М. : Радио и связь, 1985. – 323 с.

3. Мендельсон Э. Введение в математическую логику. – М. : Либроком, 2010. – 320 с.

4. Рассел С., Норвиг П. Искусственный интеллект: современный подход. – М. : Вильямс, 2006, – 292 с.

5. Zadeh L.A. Fuzzy sets // Information and Control. – 2011. – V. 8. – № 3. – P. 338–353.

6. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений. – М. : Мир, 1976. – 166 с.

7. Кофман А. Введение в теорию нечетких множеств. – М. : Радио и связь, 1982. – 432 с.

8. Никонов В.В. Логический формализм в автоматизированных системах управления качеством данных // Промышленные АСУ и контроллеры. – 2012. – № 9. – С. 18–25.

9. Морозова Т.Ю. Немонотонный вывод в задачах принятия управленческих решений // Сборник научных трудов. – М. : МГУПИ, 2010. – С. 99–112.

References:

1. Chen Ch., Li P. Matematicheskaya logika I avtomaticheskoe dokazatelstvo teorem [Symbolic Logic and Mechanical Theorem Proving]. – M. : Nauka [Moscow: Publishing house «Science»], 1983. – 358 p.

2. Nilson N.D. Principy iskusstvennogo intellekta [Principles of artificial intelligence]. – M. : Radio I svyuz [Moscow: Publishing house «Radio and Communications»], 1985. – 323 p.

3. Mendelson E. Vvedenie v matematicheskuyu logiku [Introduction to mathematical logic]. – M. : Librokom [Moscow: Publishing house «Librokom»], 2010. – 320 p.

4. Rassel S., Norvik P. Iskusstvennyi intellect: sovremennyi podhod. [Artificial Intelligence: a Modern Approach]. – M. : Vilyums [Moscow: Publishing house « Vilyums »], 2006. – 292 p.

5. Zadeh L.A. Fuzzy sets // Information and Control. – 2011. – V. 8. – № 3. – P. 338–353.

6. Zade L. Ponyutie lingvisticheskoi peremennoi i ego primenenie k prinyutiyu priblizhenyh reshenii [The concept of a linguistic variable and its application to the adoption of approximate solutions]. – M. : Mir [Moscow: Publishing house «World»], 1976. – 166 p.

7. Kofman A. Vvedenie v teoriyu nechetkih mnojestv [Introduction to the theory of fuzzy sets]. – M. : Radio I svyuz [Moscow: Publishing house «Radio and Communications»], 1982. – 432 p.

8. Nikonov V.V. Logicheskii formalism v avtomatizirovannyh sistemah upravleniy kachestvom dannyh [Logical formalism in automated systems for managing data quality]. Promyshlennye ASU i kontrollery [Industrial automation control systems and controllers]. – 2012. – № 9. – P. 18–25.

9. Morozova T.Y. Nemonotonnyi vyvod v zadachah prinyatiya ypravlencheskih reshenii. [Nonmonotonic inference in problems of decision making]. – 2010. – P. 99–112.