УДК 004.021,004.42

© Аджит Абрахам, П. Н. Семченко, 2013

ЭКСПЕРТНЫЕ СИСТЕМЫ НА ОСНОВЕ ПРАВИЛ

Аджит Абрахам – доктор, профессор MIR Labs, e-mail: ajith.abraham@ieee.org (SNIRE, Оберн, США); *Семченко П. Н.* – асп. кафедры «Автоматика и системотехника», e-mail: pavelsemch@hotmail.com (ТОГУ)

В статье описывается, как экспертные системы на основе правил могут быть полезным для поддержки принятия решений. Представлены принципы работы экспертных систем на основе правил, в том числе и с расширениями на базе нечёткой логики. Проиллюстрированы архитектура нечёткой экспертной системы и механизмы нечёткого логического вывода.

This article describes how rule-based expert systems can be useful for decision-making support. The working principles of rule-based expert systems including expansions based on fuzzy logic are presented. The architecture of fuzzy expert system and fuzzy inference engines are illustrated.

Ключевые слова: нечёткая логика, экспертная система, эвристика, прогнозирование, основанные на правилах системы.

Введение#

Решением проблем является процесс поиска решения, когда путь, ведущий к решению, неочевиден. Даже если мы знакомы с несколькими методами решения проблем, в реальном мире многие проблемы не могут быть решены методами, с которыми мы знакомы. Удивительно, для некоторых сложных проблем никакая прямая техника решения не известна вообще. Для этих задач эвристические методы решения могут быть единственной альтернативой. Эвристика может быть упрощена в качестве стратегии, которая является мощной и общей, но не абсолютно гарантирующей обеспечение наилучшего решения. Эвристические методы используются для решения конкретных проблем. Предыдущий опыт и некоторые общие правила могли бы помочь легче найти хорошие эвристические методы. Люди часто используют эвристические методы при решении задач. Конечно, если эвристический метод не позволяет решить задачу, необходимо выбрать другой эвристический метод, или знать, что целесообразно отказаться от решения. Выбор случайных решений, использование «жадных алгоритмов», развитие основного эвристического метода для того, чтобы найти лучший эвристический

метод, являются только некоторыми из популярных подходов, используемых в эвристическом решении задач [1]. Эвристическое решение задачи состоит в нахождении набора правил или процедуры, которые находят удовлетворительное решение конкретной задачи. Хорошим примером является нахождение пути через лабиринт. Чтобы пройти путь к конечной цели, необходимо пошаговое движение. Очень часто делаются ложные ходы, но в большинстве случаев мы решаем задачу без особых затруднений. Для задачи лабиринта простое эвристическое правило может быть таким: *"выбирать направление, в котором, по-видимому, можно добиться успеха"*. Другим хорошим примером является задача календарного планирования, заключающаяся в том, чтобы составить график J_n независимых заданий $n = \{1,2,....N\}$ с использованием R_m различных средств $m = \{1,2,....,M\}$ с целью минимизации времени выполнения всех заданий и эффективного использовании всех средств. Каждое J_n задание имеет технические условия в качестве P_j циклов и средство $R_м$ имеет скорость S_i циклов за единицу времени. Каждое задание J_n должно выполняться с использованием средства R_m до своего завершения. Если C_j – это время, требущееся на завершение последнего задания j, тогда мы полагаем $C_{max} = max\ \{C_j\ ,\ j=1,...,N\}$ как период выполнения задания и ΣC_j как время выполнения всего объёма работ. Задача состоит в том, чтобы найти оптимальное расписание, которое оптимизирует время выполнения всего объёма работ и период выполнения задания. Несколько простых эвристических правил для достижения этой цели - это планирование выполнения самых коротких заданий на самых быстрых средствах (Shortest Job on the Fastest Resource – SJFR), которая сведет к минимуму ΣC_J или планирование самого долгого задания на самых быстрых средствах (Longest Job on the Fastest Resource – LJFR), которая сведет к минимуму $C_{max.}$ Минимизация ΣC_J порождает быстрое выполнение заданий средней продолжительности за счёт выполнения долгих заданий, требующих много времени, тогда как сведение к минимуму $C_{max,}$ порождает то, что никакая задача не занимает слишком много времени, однако в целом выполнение большинства заданий занимает не самое короткое время. Таким образом, минимизация C_{max} приведёт к максимизации ΣC_j, что делает задачу более интересной.

В отличие от эвристических методов, алгоритмы являются простыми процедурами, которые гарантированно работают каждый раз. Например, некоторые ежедневные рутинные задачи могут быть сформулированы в формате алгоритма (например, запуск автомобиля). Однако для "решателя задач", чтобы он был более адаптивным, должны быть введены новые элементы или новые обстоятельства. Многие реальные проблемы в мире не могут быть сведены к алгоритмам, что приводит нас к поискам более эффективных методов [2].

Основанные на правилах системы#

Обычные компьютерные программы для решения задач создаются с использованием хорошо структурированных алгоритмов, структур данных и чётких стратегий рассуждений для поиска решений. Для трудных проблем, с

которыми связывают экспертные системы, более полезным является использование эвристики: стратегии, которые часто приводят к правильному решению, но также иногда могут привести к неверному результату. Обычные системы, основанные на правилах, используют экспертные знания для решения реальных жизненных проблем, которые обычно требуют человеческого интеллекта. Экспертное знание часто представляется в виде *правил* или как *данные* в компьютере. В зависимости от требований проблемы эти правила и данные могут повторно использоваться для решения проблем. Экспертные системы на основе правил играют важную роль в современных интеллектуальных системах и их применении в стратегическом целеполагании, планировании, проектировании, распределении средств по задачам, контроле неисправностей, диагностике и так далее. С техническим прогрессом в последнее десятилетие сегодняшние пользователи могут выбрать из десятков коммерческих пакетов программного обеспечения с дружественным графическим пользовательским интерфейсом [2]. Обычные компьютерные программы решают задачи, используя находящую решения логику, которая содержит очень небольшие знания помимо основного алгоритма для решения данной конкретной проблемы. Базовые знания часто вкладывается как часть программного кода, таким образом, программа должна быть перестроена при изменении знаний. Основанные на знаниях экспертные системы собирают небольшие фрагменты человеческих ноу-хау в базу знаний, чтобы производить рассуждения по проблеме, используя соответствующее знание. Важным преимуществом является то, что в области одной базы знаний различные проблемы могут быть решены с помощью той же программы без затрат на перепрограммирование. Кроме того, экспертные системы могут раскрыть ход рассуждений и хорошо справляются степенями достоверности и неопределенности, с которыми обычные алгоритмы не могут справиться [3].

Рис. 1. Архитектура простой экспертной системы

Основные компоненты экспертной системы изображены на рис. 1. База знаний хранит всю соответствующую информацию, данные, правила, случаи,

и отношения, используемые экспертной системой. База знаний может сочетать знания нескольких экспертов-людей. Правило – это условный оператор, который связывает имеющиеся условия с действиями или результатами. Фрейм – это другой подход, используемый для фиксирования и сохранения знаний в базе знаний. Это связывает объект или элемент к различным фактам или значениям. Представление, основанное на фреймах, идеально подходит для объектно-ориентированного представления. Экспертные системы с использованием фреймов для хранения знаний называют также экспертными системами, основанными на фреймах.

Цель механизма логического вывода – это получение информации и связей с базой знаний и предоставление ответов, прогнозов и предложений путём, который прошёл бы человек-эксперт. Механизм логического вывода должен найти верные факты, интерпретации и правила и скомпоновать их правильно. Широко используются два метода логического вывода. Обратный вывод представляет собой процесс, начинающийся с имеющегося заключения, и работает он в обратном порядке, чтобы обнаружить входные факты. Прямой вывод начинает работу с фактами и работает в прямом направлении до вывода заключений.

Средство разъяснения позволяет пользователю понять, как экспертная система пришла к определённым результатам. Основной целью средства пополнения базы знаний объекта является предоставление удобных и эффективных средств для сбора и хранения всех компонентов базы знаний.

Очень часто *пользовательский интерфейс* специализированного программного обеспечения используется для разработки, обновления и использования экспертных систем. Целью пользовательского интерфейса являетсяоблегчение использования экспертной системы разработчиками, пользователями и администраторами.

Механизм вывода в системах, основанных на правилах#

Основанная на правилах система состоит из *правил ЕСЛИ-ТО,* множества *фактов,* и некоторого *интерпретатора*, управляющего применением правил с учётом фактов. Эти *выражения правил ЕСЛИ-ТО* используются для разработки условных выражений, которые составляют полную базу знаний. Отдельное *правило ЕСЛИ-ТО* принимает вид «если *x* является *A*, то *y* является *B*», условная часть правила «если *x* является *A*» называется *антецедентом*, а часть правила, являющая собой следствие, «то *y* является *B*», называется *консеквентом* или следствием. Существуют два основных вида механизма логического вывода: *системы прямого вывода* и *системы обратного вывода*.

В *системе прямого вывода* начальные факты обрабатываются в первую очередь и используются в правилах для того, чтобы сделать новые выводы с учётом этих фактов.

В *системе обратного вывода* мы пытаемся обработать гипотезу (или решение/цель) первой и продолжаем поиски правил, которые позволили прийти к гипотезе.

В то время как происходит обработка, новые подцели также помечаются для проверки корректности. Системы прямого вывода в первую очередь работают с данными, в то время как системы обратного вывода работают с целью. Рассмотрим пример с следующим набором *правил ЕСЛИ-ТО*:

Правило 1: *Если* A *и* C, *то* Y
Правило 2: *Если* A *и* X, *то* Z
Правило 3: *Если* B, *то* X
Правило 4: *Если* Z, *то* D

Задача состоит в том, чтобы доказать, что D верно, учитывая, что A и B истинны. В соответствии с *прямым выводом*, начинаем с Правила 1 и идём вниз, пока не найдётся правило, которое активируется. Правило 3 является единственным, которое активировалось в первой итерации. После первой итерации, можно сделать вывод, что A, B и X являются истиной. Вторая итерация использует эту полезную информацию. После второй итерации активируется добавляющие Z в список истинных фактов Правило 2, в свою очередь помогающее активироваться Правилу 4, доказывающему, что D – истинно. Стратегия прямого вывода особенно подходит в ситуациях, когда сбор данных является дорогим, но имеется несколько фактов. Однако, особое внимание должно быть уделено тогда, когда эти правила создаются, с указанием предварительных условий как можно точнее, когда различные правила должны активироваться.

В методе обратного вывода обработка начинается с желаемой цели, а затем происходят попытки найти доказательства для подтверждения цели. Возвращаясь к тому же примеру, задаче доказать, что D верно, в первую очередь будет инициализировано правило, которое подтверждает D. Правило 4 также предоставляет подцель доказать, что Z истинно. Теперь Правило 2 вступает в игру, и, так как уже известно, что A истинно, новой подцелью является доказательство того, что X верно. Правило 3 предоставляет следующую подцель доказать, что B верно. Но то, что B истинно, является одним из входных утверждений. Поэтому можно сделать вывод, что X истинно, откуда следует, что Z является истиной, что в свою очередь также означает, что D – это истина. Обратный вывод является полезным в ситуациях, когда количество данных является потенциально очень большим, и где некоторые конкретные характеристики рассматриваемой системы представляет интерес. Если нет достаточных знаний, позволяющих прийти к заключению, или имеется некоторая определённая гипотеза для проверки на корректность, системы прямого вывода могут быть неэффективными. В принципе мы можем использовать тот же набор правил и для прямого, и для обратного вывода. В случае обратного вывода, основная проблема состоит в соответствии заключения правила некоторой цели, которая должна быть доказана; часть правила "тогда" (консеквент) обычно не выражается как действие для выполнения, но выражается

просто в качестве состояния, которое будет истиной, если условная часть (части) будет истиной [4].

Развитие экспертных систем #

Шаги в процессе развития экспертных систем включают определение фактических потребностей, приобретения знаний, построения компонентов экспертной системы, внедрение результатов и разработке процедур для технического обслуживания и анализа.

Приобретение знаний является наиболее важным элементом в развитии экспертной системы [5]. Знания могут быть получены путём опроса экспертам в этой области и/или обучения на опыте. Очень часто люди выражают знания в естественном языке (разговорный язык) или с помощью письма, или символических терминах. Есть несколько методов для извлечения человеческих знаний. Познавательный анализ работы (Cognitive Work Analysis, CWA) и познавательный анализ задачи (Cognitive Task Analysis, CTA) обеспечивают платформы для извлечения знаний. CWA – это технология для анализа, разработки и оценки человеко-машинных интерактивных системы [6]. CTA является методом для выявления познавательных навыков, психологических требований и потребностей для выполнения задач квалификации [7]. CTA сосредоточен на описании представления познавательных элементов, которые определяют целевую генерацию и принятие решений. CTA является надёжным методом для извлечения человеческих знаний, поскольку она основана на наблюдениях и интервью.

Большинство экспертных систем разрабатываются с использованием специализированных программных средств, называемых оболочками. Эти оболочки оснащены механизмом вывода (обратного вывода, прямого вывода, или тот и другой вместе), и требуют знаний, которые будут введены в соответствии с указанным форматом. Один из самых популярных оболочек широко используется во всём правительстве, промышленности и научных кругах CLIPS [8]. CLIPS – это средство разработки экспертных систем, которое предоставляет полную среду для конструирования правила и/или объектно-ориентированных экспертных систем. CLIPS обеспечивает единый инструмент для обработки широкого спектра знаний с поддержкой трёх различных парадигм программирования: основанной на правилах, объектно-ориентированный и процедурный характер. CLIPS написан на C для мобильности и скорости, и был установлен на различных операционных системах без изменений в коде.

Нечёткие экспертные системы#

Мир информации окружён неопределенностями и неточностями. Человеческий процесс рассуждений может обрабатывать неточные, неопределённые и расплывчатые концепции соответствующим образом. Как правило, человеческое мышление, рассуждения и процесс восприятия не могут быть выражены точно. Эти типы опыта редко могут быть выражены или измерены с

помощью статистических или вероятностных теорий. Нечёткая логика обеспечивает основу для моделирования неопределенности, человеческого мышления, рассуждений и процесса восприятия. Нечёткие системы были впервые представлены Лотфи Заде, профессором Калифорнийского университета в Беркли, в его основополагающей работе, опубликованной в 1965 [9].

Нечёткие экспертные системы – это просто экспертные системы, которые использует набор нечётких функций принадлежности и правил вместо булевой логики, для рассуждения о данных [10]. Правила в нечёткой экспертной системе, как правило, находятся в форме, аналогичной следующей:

Если A *низкое* и B *высокое*, то X = *среднее*

где A и B являются входными переменными, а X является выходной переменной. Здесь низкое, высокое, среднее являются нечёткими множествами, определённые на A, B и X соответственно. Антецендент (условие правила) описывает, в какой степени применимо правило, в то время как консеквент правила присваивает функции принадлежности к одной или нескольким выходным переменным.

Пусть X – это пространство объектов, а x – это общий элемент пространства X. Классическое множество A, $A \subseteq X$, определено как совокупность элементов или объектов $x \in X$, таких что x может либо принадлежать, либо не принадлежать A. Нечёткое множество A на пространстве X определено как множество упорядоченных пар:

$$A = \{(x, \mu_A(x)) \mid x \in X\} \quad (1)$$

где $\mu_A(x)$ называется функцией принадлежности (membership function, MF) на нечётком множестве A. Функция принадлежности ставит в соответствие каждый элемент пространства X степени принадлежности (или значению принадлежности) между 0 и 1. Очевидно, что выражение (1) является простым расширением определения классического множества, в котором характеристическая функция может иметь значения между 0 и 1.

Пересечением двух нечётких множеств A и B в общем определяется функцией T: [0,1] × [0,1]→ [0,1], которая объединяет в одно целое две степени принадлежности следующим образом:

$$\mu_{A\cap B}(x) = T(\mu_A(x), \mu_B(x)) = \mu_A(x) \bar{*} \mu_B(x) \quad (2)$$

где $\bar{*}$ является бинарным оператором функции T. Этот класс нечётких операторов пересечения, как правило, называется операторами T-нормы [11]. Четыре наиболее часто используемых оператора T-нормы следующие:

Минимум: $T_{\min}(a,b) = \min(a,b) = a \wedge b$ (3)

Алгебраическое произведение: $T_{ap}(a,b) = ab$ (4)

Ограниченное произведение: $T_{bp}(a,b) = 0 \vee (a + b - 1)$ (5)

$$\text{Сильное произведение: } T_{dp}(a,b)=\begin{cases} a, if\ b=1 \\ b, if\ a=1 \\ 0, if\ a,b<1 \end{cases} \quad (6)$$

Как и пересечение, оператор нечёткого объединения в общем описывается функцией S: [0,1] × [0,1] → [0,1], которая объединяет в одно целое две степени принадлежности следующим образом:

$$\mu_{A\cup B}(x)=S(\mu_A(x),\mu_B(x))=\mu_A(x)\overline{+}\mu_B(x) \quad (7)$$

где $\overline{+}$ является бинарным оператором S. Этот класс нечётких операторов пересечения, как правило, называется операторами Т-конормы (или *S-нормы*) [11]. Четыре наиболее часто используемых оператора Т-конормы следующие:

$$\text{Максимум: } S_{\max}(a,b)=\max(a,b)=a\vee b \quad (8)$$

$$\text{Алгебраическое объединение: } S_{as}(a,b)=a+b-ab \quad (9)$$

$$\text{Ограниченное объединение: } S_{bs}(a,b)=1\wedge(a+b) \quad (10)$$

$$\text{Сильное объединение: } S_{ds}(a,b)=\begin{cases} a, if\ b=0 \\ b, if\ a=0 \\ 1, if\ a,b>0 \end{cases} \quad (11)$$

И оператор пересечения, и оператор объединения сохраняют некоторые свойства классических операций на множествах. В частности, они являются ассоциативными и коммутативными.

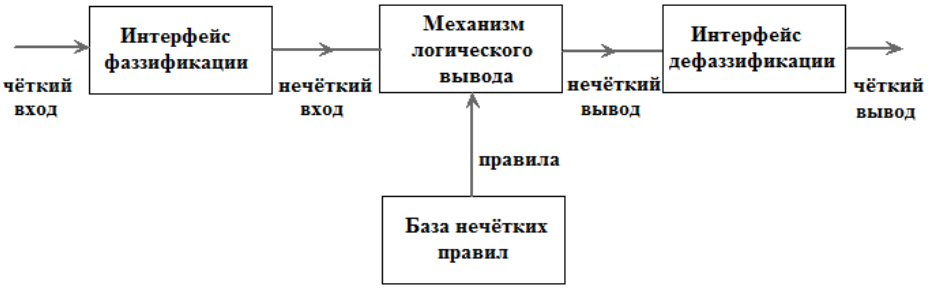

Рис. 2. Базовая архитектура нечеткой экспертной системы

Рис. 2 иллюстрирует основную архитектуру нечёткой экспертной системы. Основными компонентами являются интерфейс фаззификации, база нечётких правил (база знаний), механизм логического вывода (логика принятия решений) и интерфейс дефаззификации. Входные переменные преобразуются в нечёткие посредством функций принадлежности, определённых на входных переменных и применяемых к фактическим значениям входных переменных для установления степени принадлежности для каждого антецендента правила. Нечеткие правила *если-то* и нечеткие рассуждения являются основой нечётких экспертных систем, которые является наиболее важными инструмен-

тами моделирования на основе теории нечётких множеств. База нечётких правил выражена в форме правил *если-то*, в которых антецеденты и консеквенты имеют лингвистические переменные. Совокупность этих нечётких правил формирует базу правил для системы нечёткой логики. При использовании подходящей процедуры логического вывода вычисляется степень истинности для антецедента каждого правила, а затем применяется к консеквенту каждого правила. Это приводит к получению одного нечёткого подмножества, которое будет присвоено каждой выходной переменной для каждого правила. Кроме того, при использовании подходящей процедуры составления, все нечёткие подмножества, присвоенные каждой выходной переменной, объединяются вместе, чтобы сформировать единственное нечёткое подмножество для каждой выходной переменной. Наконец, дефаззификация применяется для преобразования нечёткого вывода в чёткий вывод.

В сущности, механизм нечёткого логического вывода может принимать либо нечёткий, либо чёткий ввод, выходов, но на выход всегда выдаются нечёткие множества. Задача дефаззификации – извлечение чёткого вывода, который наилучшим образом представляет нечёткое множество. С чётким вводом и выводом, нечёткий механизм логического вывода осуществляет нелинейное сопоставление от своего входного пространства к выходному, используя множество нечётких правил *если-то*.

В дальнейшем следует описание двух наиболее распространённых в различных приложениях популярных механизмов нечёткого логического вывода. Различия между этими двумя механизмами нечёткого логического вывода находятся в консеквентах их нечётких правил, и, таким образом, их объединяющие и дефаззифицирующие процедуры отличаются соответствующим образом.

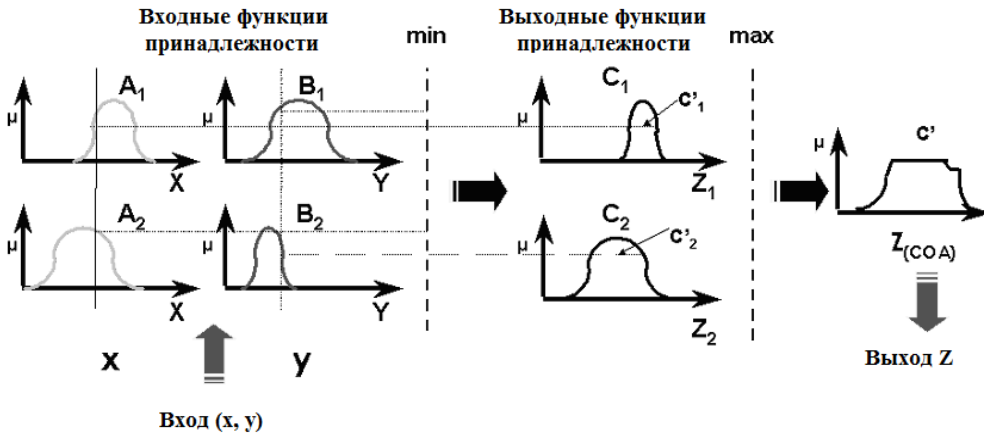

Рис. 3. Система нечёткого вывода Мамдани с использованием min и max операторов для Т-нормы и Т-конормы

В соответствии с изображённым на Рис. 3 механизмом нечёткого логического вывода Мамдани [12], антецеденты и консеквенты правила определяются нечёткими множествами и имеют следующую структуру:

$$if\ x\ is\ A_1\ and\ y\ is\ B_1\ then\ z_1 = C_1^{'} \qquad (12)$$

Существует несколько методов дефаззификации. Однако, наиболее широко используемый метод дефаззификации использует способ получения центра тяжести пространства следующим образом:

$$\text{Центр тяжести пространства } Z_{COA} = \frac{\int_Z \mu_A(z)\, z\, dz}{\int_Z \mu_A(z)\, dz}, \qquad (13)$$

где $\mu_A(z)$ является объединённым выводов функции принадлежности.

Такаги, Сугэно, и Канг [13] предлагают схему логического вывода, в которой заключение нечёткого правила состоит из взвешенной линейной комбинации чёткого ввода, а не нечёткого множества Сущность механизма нечёткого логического вывода Такаги-Сугэно изображена на рис. 4, а правила имеют следующую структуру:

$$if\ x\ is\ A_1\ and\ y\ is\ B_1, then\ z_1 = p_1 x + q_1 y + r_1 \qquad (14)$$

где p_1, q_1 и r_1 являются линейными параметрами. Нечёткий контроллер Такаги-Сугэно-Канга (TSK) обычно нуждается в меньшем количестве правил, потому как их вывод уже является линейной функцией от входа, а не неизменным нечётким множеством.

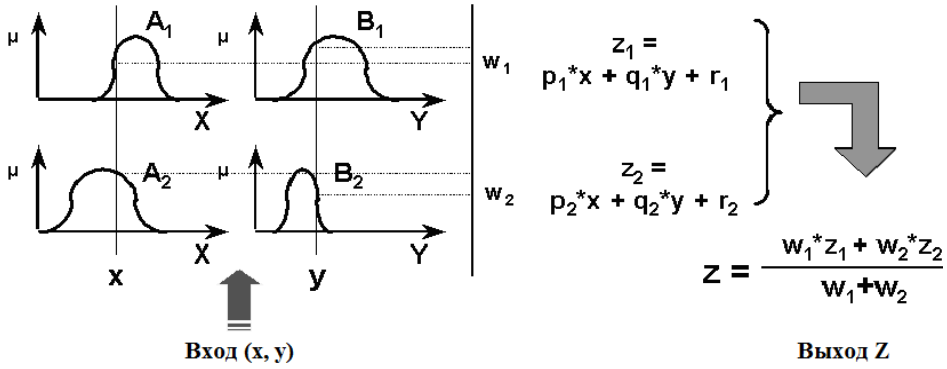

Рис. 4. Механизм нечёткого логического вывода Такаги-Сугэно с использованием оператора минимального произведения в качестве Т-нормы

Популярной оболочкой, позволяющей описывать нечёткие экспертные системы, является FuzzyCLIPS [3]. Данная модификация дополняет возможности CLIPS, предоставляя возможности нечёткой аргументации. Расширения для нечёткой логики полностью интегрированы с фактами и предполо-

жениями ядра CLIPS, что даёт возможность представлять и манипулировать нечёткими фактами и правилами.

FuzzyCLIPS может иметь дело с чёткими, нечёткими и комбинированными аргументациями, что позволяет свободно смешивать нечёткие и нормальные условия в правилах и фактах экспертной системы. Система использует два понятия мягких вычислений: нечёткость и неопределённость. FuzzyCLIPS предоставляет полезную среду для разработки приложений, использующих нечёткую логику в системах принятия решений.

Существующая на данный момент реализация FuzzyCLIPS версии 6.10d может использоваться при разработке распределённых информационных систем поддержки принятия решений, таких как клиент-серверная среда динамических экспертных систем [14]. Действительно, дополнения для нечёткой логики в системе FuzzyCLIPS хорошо документированы [15] и существенно расширяют возможности инженеров по знаниям при разработке экспертных систем в различных предметных областях. Адаптация пользовательского интерфейса FuzzyCLIPS вместе с обеспечением преемственности последней версии обычного ядра CLIPS для серверной компоненты клиент-серверной среды динамических экспертных систем возможна с помощью двух методов: либо необходима разработка библиотеки по технологии OCX для подключения к программным продуктам, либо необходимо создать модуль по использованию имеющегося в пакете FuzzyCLIPS интерфейса командной строки при разработке компонент клиент-серверной среды динамических экспертных систем.

Заключение

Экспертные системы, основанные на правилах, находят применение в большом количестве прикладных областей. Важным преимуществом нечёткой экспертной системы является то, что знания выражаются в лёгкой для понимания форме языковых правил. Мы можем организовать извлечение данных нечёткой экспертной системы с помощью нейронной сети обучения, эволюционных вычислений или других методов адаптации. Мы можем предположить, что увидим значительный рост применения данных технологий в будущем.

В настоящее время широко распространяются технологии, основанные на использовании нечёткой логики. Нечёткая логика является основой для систем поддержки принятия решений и существенно сокращает дистанцию между человеческим языком и компьютерными формальными языками.

Библиографические ссылки

1. *Michalewicz Z., Fogel D.B.* How to Solve It: Modern Heuristics. Springer Verlag, Germany, 1999.

2. *Abraham A.* Rule Based Expert Systems. Handbook for Measurement Systems Design ed. London: John Wiley and Sons Ltd., 2005. 909-919 pp.

3. *Ignizio J.P.* Introduction to Expert Systems: The Development and Implementation of Rule-Based Expert Systems. McGraw-Hill, Inc., 1991.

4. *Giarratano J.C., Riley G.D.* Expert Systems: Principles and Programming. Course Technology, 2004.

5. *Donald W.A*. A Guide to Expert Systems. Addison-Wesley, 1986.

6. *Niwa K., Sasaki K., and Ihara H.* An Experimental Comparison of Knowledge Representation Schemes, in Principles of Expert Systems. New York: IEEE Press, 1988. 133-140 pp.

7. *Vicente K.J.* Cognitive Work Analysis: Towards Safe, Productive, and Healthy Computer-Based Work. USA: Lawrence Erlbaum Associates, Inc. Press, 1999.

8. *Militallo L.G., Hutton R.J.B.* Applied Cognitive Task Analysis (ACTA): A Practitioner's Toolkit for Understanding Cognitive. 11th ed. Vol 41. Ergonomics, 1998. 1618-1642 pp.

9. *CLIPS*: A Tool for Building Expert Systems [Электронный ресурс] URL: http:// clipsrules.sourceforge.net/ (дата обращения: 24.11.2012).

10. *Zadeh L.A.* Fuzzy Sets, Information and Control. Vol 8. 1965. 338-353. pp.

11. *Schneider M., Langholz G., Kandel A., and Chew G.* Fuzzy Expert System Tools. USA: John Wiley & Sons, 1996.

12. *Jang J.S.R., Sun C.T., and Mizutani E.* Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. USA: Prentice Hall Inc, 1997.

13. *Mamdani E.H., Assilian S.* An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller // International Journal of Man-Machine Studies. 1975. Vol. 7. No. 1. pp. 1-13.

14. *Takagi T., Sugeno M.* Fuzzy idendfication of systems and its applications of modeling and control. IEEE Transactions of Systems. Man and Cybernetics. 1985. 116-132 pp.

15. *Воронин В.В., Семченко П.Н.* Концепция клиент-серверной среды динамических экспертных систем // Информатика и системы управления. - 2010. - № 3(25).

16. *Orchard B.* FuzzyCLIPS Version 6.10d User's Guide. Canada. 2004.