УДК 004.9 **doi: 10.26906/SUNZ.2018.4.152**

В. Д. Хох

Центральноукраїнський національний технічний університет, Кропивницький, Україна

РОЗРОБКА СКРИПТОВОЇ МОВИ ЗАПИСУ ПРАВИЛ В ЕКСПЕРТНІЙ СИСТЕМІ НА ОСНОВІ НЕЧІТКОЇ ЛОГІКИ

Предметом вивчення у статті є процес розробки експертних систем із використанням апарату нечіткої логіки та продукційною базою знань. **Метою** є розробка скриптової мови для запису правил у базу знань експертної системи, що використовує апарат нечіткої логіки. Скриптова мова, повинна реалізувати мінімальні можливості для опису правил, а саме: опис змінних; запис та обробка логічних виразів; визначення просторів імен, – що дозволило би більш точно описати певну ситуацію; надати більш гнучкі засоби опису; виконувати логічні операції між різними типами даних; компенсувати неоднозначність висновків експертів, зважаючи на можливу недостовірність даних. **Завдання:** дослідити існуючі, поширені скриптові мови, що використовуються для запису правил у базі знань експертної системи, здійснити аналіз існуючих методів їх застосування; розробити власну мову опису правил – мову логічного програмування, інтерпретовану, інтегровану в механізми роботи експертної системи, однак таку, яку можливо виділити в самостійну. Отримані такі **результати:** Розроблено скриптову мову для запису правил у базу знань експертної системи, що використовує апарат нечіткої логіки. Скриптова мова реалізує мінімальні можливості для опису правил: опис змінних; запис та обробка логічних виразів; визначення просторів імен, що дозволяє: більш точно описати певну ситуацію; надати більш гнучкі засоби опису; виконувати логічні операції між різними типами даних; компенсувати неоднозначність висновків експертів, зважаючи на можливу недостовірність даних. Визначено напрямки подальших досліджень та роботи по вдосконаленню та розширенню області застосування розробленої скриптової мови. **Висновки.** Розроблено скриптову мову для запису правил у базу знань експертної системи, що використовує апарат нечіткої логіки. На даний момент розроблена скриптова мова реалізує мінімальні можливості для опису правил: опис змінних; запис та обробка логічних виразів; визначення просторів імен. Це дозволяє: більш точно описати певну ситуацію; надати більш гнучкі засоби опису; виконувати логічні операції між різними типами даних; компенсувати неоднозначність висновків експертів, зважаючи на можливу недостовірність даних. Планується реалізувати підтримку конструкцій, що дозволить використовувати скриптову мову окремо від експертної системи.

Ключові слова: експертна система, скриптова мова, нечітка логіка, інтелектуальна система, продукційна модель представлення знань.

Вступ

Загалом, теоретичної чи практичної відмінності між мовою програмування та скриптовою мовою - немає. Тим не менш, можливо виділити певні властивості і особливості за якими алгоритмічну мову починають називати скриптовою.

По-перше, скриптова мова, зазвичай, використовується і виконується в середині певної програми. Це обумовлюється тим, що скриптові мови, зазвичай інтерпретуються і не вимагають компіляції.

По-друге, скритові мови, часто розроблені і орієнтовані на вирішення певних, специфічних задач.

По-третє, скриптові мови найчастіше «дуже» високорівневі і не вимагають великої кількості часу для їх вивчення.

Найпоширеніша область використання скриптової мови це автоматизація певного процесу в середині вже існуючої програми/системи. Найчастіше, скриптова мова поставляється як вбудована мова в певну систему.

Розглянемо деякі з мов програмування, яким притаманні, в тій чи іншій мірі, вищезгадані властивості, а також наведемо приклади їх застосування в експертних системах (ЕС).

JavaScript – високорівнева, інтерпретована, мультипарадигмова мова програмування з динамічною типізацією. Одна з найвідоміших реалізацій стандарту ECMA-262. Мова для виконання обчислень та маніпулювання обчислювальними об’єктами в середовищі хосту. Використовує бінарну логіку для виконання логічних операцій [1]. Її використання було запропоновано у веб-базованій системі прийняття рішень на основі нечіткої логіки як засіб, що допоможе розвантажити сервер експертної системи, виконуючи частину розрахунків на стороні клієнта [2]. Також використовувалася для побудови SeTES – експертної системи з самонавчанням. Там JavaScript було використано для організації роботи інтерфейсу користувача та передачі даних за допомогою JSON до безпосередньо самої експертної системи, яка в свою чергу написана на Python [3].

Python – високорівнева інтерпретована, об'єктно-орієнтована мова програмування з строгою динамічною типізацією [4]. Використовує бінарну логіку для виконання логічних операцій. Використовувалася для розробки SeTES, а саме для реалізації основної програми експертної системи – Python Runner, яка отримує та обробляє дані від користувачів після чого передає дані PHP-скриптам, які генерують HTML-сторінку з вихідними даними [3]. Також мова Python використовувався у експертній системі діагностики нещасних випадків на основі дерева рішень у реальному часі, де її було використано для реалізації модуля діагностики. Сама ж система розроблена для функціонування на комерційних електростанціях [5].

Prolog – мова логічного програмування, декларативна. Дана мова базується на невеликій кількості

базових механізмів включаючи зіставлення зразків, даних структурованих під дерева та автоматичному поверненню назад [6]. Прикладом використання Prolog`у може бути експертна система для аналізу правил брендмауера. В цій системі Prolog було використано для запису правил у базі знань, це було зроблено через зручність їх представлення засобами декларативної мови, оскільки такий запис дозволяє легко виразити знання без накладання інформації на обчислювані деталі [7]. Prolog як і вище розглянуті мови використовує бінарну логіку для виконання логічних операцій.

Bousi~Prolog – мова логічного програмування із синтаксисом як і у Prolog, але вона доповнена символом «~~», який використовується для опису подібності за допомогою рівняння подібності. Завдяки можливості використовувати подібності, Bousi ~ Prolog здатна використовувати нечітку логіку для виконання логічних операцій [8].

FASILL – мова логічного програмування першого порядку, виконує логічні операції з використанням нечіткої логіки. Правила в програмі FASILL мають ту ж роль, що й Prolog. Імплементовано у системі FLOPER, в свою чергу FLOPER реалізовано з використанням Sicstus Prolog версії 3.12.5, система доступна онлайн і використовується для різноманітних досліджень [9].

Метою даної роботи є розробка скриптової мови для запису правил у базу знань експертної системи, що використовує апарат нечіткої логіки.

Основна частина

В процесі розробки експертної системи для аудиту інформаційної безпеки в комп'ютерних системах та мережах [10], заснованої на нечіткій логіці постало питання про те, в якому форматі будуть записуватися правила в базі знань. Спочатку передбачався запис правил у вигляді продукцій, так як інші способи представлення знань були розглянуті [11] і відхилені через громіздкість, або через те, що передбачають роботу з макрознаннями, які не дуже раціонально використовувати в системі, робота якої проходить у досить формалізованому середовищі. Правило у вигляді імплікації не давало достатньої гнучкості, так як має вигляд: **ЯКЩО** *А* і *Б* і *С* і *Д* **ТОДІ** *Е*.

Такий вид правил хоч і зрозумілий, і простий все ж значно обмежує можливості для опису конкретного випадку. До того ж інформативна ємність такого правила невелика, в зв'язку з цим міркувань з боку експертної системи необхідно досить багато, що є досить відомою проблемою експертних систем з продукційною моделлю представлення знань. Вважається, що продукційна експертна система є непрацездатною, якщо в базі знань зберігатися тисяча або більше продукцій [12]. Ще одна проблема – це однобокість подання знань, передбачається, що конкретний випадок можна описати наявністю інформації за декількома фактами, відсутня можливість опису будь-яких винятків в правилі. До всього іншого, в розроблюваній експертній системі особлива увага приділяється гнучкості описів для того, щоб дати можливість реалізовувати більш творчі підходи для досягнення цілей – опис правил у вигляді продукцій зводять такі наміри до нуля. Особливістю функціонування експертної системи в області пентестінгу (тестування на проникнення, penetration-testing) є те, що деякі робочі рішення далеко не завжди очевидні, або ж задовольняють вимоги частково, тим не менш досягнення цих результатів надає можливість задовольнити вимоги в повному обсязі [13], тому бажано мати на виході кілька можливих пропозицій, а поруч деякий коефіцієнт релевантності щодо поточного контексту. Ще однією особливістю роботи експертної системи в цій галузі є те, що не завжди експерт може бути впевнений у результативності запропонованого рішення в повній мірі, хоча б тому, що не може бути впевненим у точності зібраної інформації, в разі тестування «чорного ящика» [14].

Таким чином, можна сформулювати основні проблеми експертної системи заснованої на продукційній базі знань, які послужили причиною створення власної мови опису правил:

– неінформативність продукцій;

– недостатня гнучкість описів;

– однобокість представлення знань;

– відсутність на виході коефіцієнта релевантності;

– відносна впевненість експерта в достовірності даних, з якими працює ЕС.

Для вирішення цих проблем було прийнято рішення розробити власну мову опису правил. Розроблена мова логічного програмування, інтерпретована, інтегрована в механізми роботи ЕС, однак її можливо виділити в самостійну. Вона здатна виконувати логічні операції І, АБО, НІ. Конструкція логічного виразу може бути, практично, будь-якої «глибини» і з залученням необхідної кількості змінних. Змінні в мові представлені трьома типами int, float та str. Однією з особливості мови є те, що змінні задаються у вигляді нечітких. У зв'язку з цим для опису змінної типу int необхідно вказати її ім'я, мінімальне значення, максимальне значення і тип характеристичної функції, те ж саме необхідно вказати і при описі змінної типу float.

Опис змінної типу str відрізняється від інших, так як задається у вигляді множини рядків із зазначенням коефіцієнта релевантності або ваги для кожного елемента множини. Або коефіцієнти будуть розраховуватися автоматично, для чого при описі вказується відповідна опція. В цілому опис змінної типу str виглядає наступним чином – встановлюється прапор true або false – чи буде користувач самостійно вказувати коефіцієнти для значень рядків, якщо так – то далі необхідно вказати розділовий символ, який буде розділяти значення рядка від коефіцієнта.

Синтаксис опису змінних виглядає таким чином:

• **int** *<ім'я_змінної>* [*<мінімальне_значення>*, *<максимальне_значення>*, *<номер_характеристичної_функції>*, *<простір_імен>*];

• **float** *<ім'я_змінної>* [*<мінімальне_значення>*, *<максимальне_значення>*, *<номер_характеристичної_функції>*, *<простір_імен>*];

• **str** < *ім'я_змінної* >[<*прапор*>, <*спец_символ*>, {"<*коефіцієнт*><*спец_символ*><*рядок*>", "<*коефіцієнт*><*спец_символ*><*рядок*>","<*коефіцієнт*><*спец_символ*><*рядок*>"…}, <*простір_імен*>];

Необхідно, уточнити про параметр: <номер_характеристичної_функції> – в інтерпретаторі встановлені три види характеристичних функцій, які дозволяють розрахувати ступінь релевантності змінної в залежності від контексту, графіки цих функцій наведено нижче, на рис. 1, 2, 3.

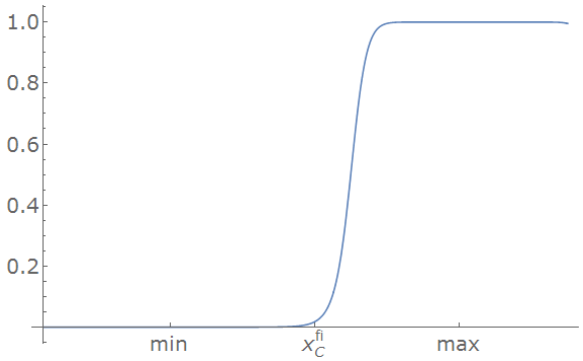

Рис. 1. Графік характеристичної функції при *fi* = 1

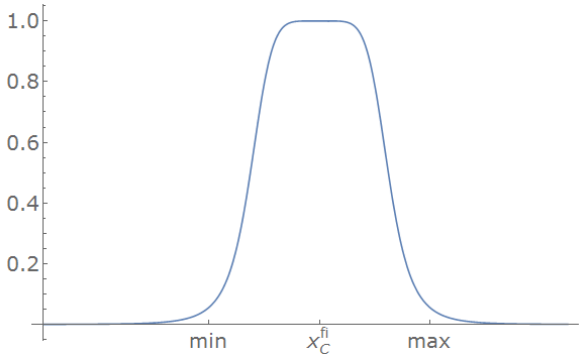

Рис. 2. Графік характеристичної функції при *fi* = 2

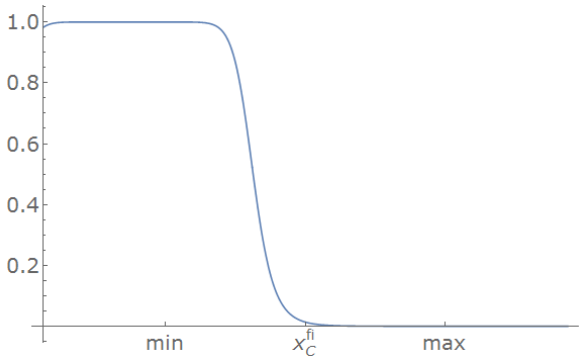

Рис. 3. Графік характеристичної функції при *fi* = 3

Загалом, характеристична функція у системі має наступний вигляд:

$$\mu A(x) = \frac{1}{1+\left(\frac{x - x_c^{fi}}{range}\right)^{sensibility}}, \qquad (1)$$

де *fi* – номер характеристичної функції; *sensibility* – значення чутливості системи; *range* – значення області істинності, тоді:

$$x_c^{fi} \in \left[\min, \frac{\min + \max}{2}\right], \text{ якщо } fi = 1, \qquad (2)$$

$$x_c^{fi} = \frac{\min + \max}{2}, \text{ якщо } fi = 2, \qquad (3)$$

$$x_c^{fi} \in \left[\frac{\min + \max}{2}, \max\right], \text{ якщо } fi = 3. \qquad (4)$$

Для *fi* = 1 графік функції зображено на рис. 1, для *fi* = 2 графік функції зображено на рис. 2, а для *fi* = 3 – на рис. 3.

Контекст визначається простором імен, одне правило (логічний вираз) може перебувати в одному просторі імен, але в правилі можуть бути змінні з різних. Це обробляється двигуном експертної системи для розширення її пошукових здібностей.

В цілому, правило обробляється в два етапи – на першому розраховується коефіцієнт релевантності для кожної змінної щодо поточного контексту. На другому етапі кожній змінній призначається еквівалент булевої змінної (true | false) і виконуються всі

логічні операції в правилі, на етапі прототипування вирішено, що змінна отримує еквівалент false в разі, якщо її контекст покинув зазначений діапазон в описі, а ступінь її релевантності дорівнює нулю, в іншому випадку змінна – true, а ступінь релевантності розраховується згідно характеристичної функції.

В кінці приводитися результат обчислення булевих логічних операцій разом з сумою всіх коефіцієнтів, релевантністю за правилом. Вирішено не розраховувати середнє значення за правилом, так як включення більшої кількості фактів (змінних) в правило може вказувати на більш точну оцінку поточного контексту, а відтак – підвищення коефіцієнту релевантності правила в цілому, за рахунок включення додаткових фактів в правило вважається справедливим.

Подібний підхід до опису правил здається більш ефективним, так як дозволяє більш точно описати певну ситуацію завдяки підвищенню інформативності за рахунок розгляду в кожному правилі не одного значення для кожного факту, а їх наборів/діапазонів. Гнучкість опису, підвищується за рахунок можливості використання набору з логічних операторів І, АБО, НЕ, а так само за рахунок можливості вказівки діапазону значень і можливості управління процесом розрахунку коефіцієнтів релевантності для кожного факту окремо. Правило більше не однобоке, так як вихід певного факту за рамки змінної призведе лише до падіння ступеня релевантності в цілому по правилу. І, незважаючи на те, що в такому випадку парсер нечіткої логіки відпрацює таку змінну як змінну зі значенням false, результат спрацьовування правила все одно надасть певну інформацію, яка може допомогти фахівцю доповнити картину контексту.

Використання скриптової мови для запису правил в розроблюваній системі дав можливість розраховувати для будь-якої змінної в правилі ступінь релевантності щодо поточного контексту, що дає можливість виконувати логічні операції між змінними різних типів, це дає можливість розраховувати коефіцієнти релевантності для правил з будь-яким набором даних, і дає можливість системі пропонувати на виході набір з рішень з різним ступенем впевненості в їх ефективності.

Завдяки можливості задавати в правилах змінні у вигляді діапазонів значень, експерт може компенсувати неоднозначність своїх висновків у зв'язку з можливою недостовірність деяких даних, з якими надалі може працювати система.

Висновки

Розроблено скриптову мову для запису правил у базі знань експертної системи, що використовує апарат нечіткої логіки.

На даний момент, розроблена скриптова мова реалізує мінімальні можливості для опису правил:

– Опис змінних.

– Запис та обробка логічних виразів.

– Визначення просторів імен.

Це дозволяє:

– Більш точно описати певну ситуацію.

– Надати більш гнучкі засоби опису.

– Виконувати логічні операції між різними типами даних

– Компенсувати неоднозначність висновків експертів, зважаючи на можливу недостовірність даних.

Планується реалізувати підтримку конструкцій, що дозволить використовувати скриптову мову окремо від експертної системи.

СПИСОК ЛІТЕРАТУРИ

1. Ecma International. ECMA- 262 / Ecma International. – Geneva: Ecma International, 2018. – 805 c. – (Ecma International).
2. Vahid R. A Novel Web-based Human Advisor Fuzzy Expert System / R. Vahid, H. Mahdi. // Journal of Applied Research and Technology. – 2013. – №11. – С. 161–168.
3. George J. M. SeTES, a Self - Teaching Expert System for the analysis, design and prediction of gas production from unconventional resources / J. Moridis George. // RPSEA. – 2011. – №712. – С. 1–116.
4. The Making of Python [Електронний ресурс] // Artima. – 2003. – Режим доступу до ресурсу: https://www.artima.com/intv/pythonP.html.
5. Andressa6 S. N. Accident diagnosis system based on real-time decision tree expert system / S. N. Andressa, P. A. João, S. Roberto. // American Institute of Physics. – 2017. – С. 1–6.
6. Ivan B. Prolog Programing for Artificial Intelegence / Bratko Ivan. – Harlow: Pearson education limited, 2012. – 697 c. – (Pearson). – (4; кн. 4).
7. Eronen P. An expert system for analyzing firewall rules / P. Eronen, J. Zitting. // Helsinki University of Technology. – 2001. – С. 1–8.
8. Pascual J. Bousi ~ Prolog: a Prolog Extension Language for Flexible Query Answering / J. Pascual, R. Clemente, J. Gallardo-Casero. // Electronic Notes in Theoretical Computer Science. – 2009. – №248. – С. 131–147.
9. S. Escobar (Ed.): XIV Jornadas sobre Programaci´on y Lenguajes, PROLE 2014, Revised Selected Papers EPTCS 173, 2015, pp. 71–86, doi:10.4204/EPTCS.173.6 A Fuzzy Logic Programming Environment for Managing Similarity and Truth Degrees / J.Pascual, M. Gin´es, P. Jaime, V. Carlos. // ornadas sobre Programaci´on y Lenguajes. – 2015. – С. 71–86.
10. Хох В.Д. Експертна система для автоматизації аудиту інформаційної безпеки комп'ютерних систем та мереж / В.Д. Хох, Є.В. Мелешко // Збірник тез Шостої міжнародної науково-технічної конференції "ITSEC", м. Київ, 17-19 травня 2016. – Київ: Національний авіаційний університет. – 2016. – С. 17-18.
11. Хох В.Д. Дослідження методів побудови експертних систем / В.Д. Хох, Є.В. Мелешко, М.С. Якименко // Збірник наукових праць "Системи управління, навігації та зв'язку". Випуск 4(40). – Полтава: ПНТУ ім. Ю. Кондратюка. – 2016. – С. 48-52.
12. Болотова І. С. Системы искуственного интелекта: модели и технологии, основанные на знаниях / І. С. Болотова. – Москва: Финансы и Статистика, 2012. – 664 с. – (ФГАУ ГНИИ ИТТ "Информатика").

13. AN OVERVIEW OF PENETRATION TESTING / G. B.Aileen, Y. Xiaohong, B. C. Bei-Tseng, J. Monique. // International Journal of Network Security & Its Applications (IJNSA),. – 2011. – №6. – С. 19–38.
14. Technical Guide to Information Security Testing and Assessment / S.Karen, S. Murugiah, C. Amanda, O. Angela. // National Institute of Standards and Technology. – 2008. – С. 24–34.

Рецензент: д-р техн. наук, проф. С. Г. Семенов,
Національний технічний університет "Харківський політехнічний інститут", Харків
Received (Надійшла) 19.06.2018
Accepted for publication (Прийнята до друку) 15.08.2018

Разработка скриптового языка записи правил в экспертной системе на основе нечёткой логики

В. Д. Хох

Предметом изучения в статье является процесс разработки экспертных систем с использованием аппарата нечеткой логики и продукционной базы знаний. **Целью** является разработка скриптового языка для записи правил в базу знаний экспертной системы, использующей аппарат нечеткой логики. Скриптовой язык, должен реализовать минимальные возможности для описания правил, а именно: описание переменных; запись и обработка логических выражений; определения пространств имен, - что позволило бы более точно описать определенную ситуацию; предоставить более гибкие средства описания; выполнять логические операции между различными типами данных; компенсировать неоднозначность выводов экспертов, принимая во внимание возможную недостоверность данных. **Задача:** исследовать существующие, распространенные скриптовые языки, используемые для записи правил в базе знаний экспертной системы, осуществить анализ существующих методов их применения; разработать собственный язык описания правил - язык логического программирования, интерпретированный, интегрированный в механизмы работы экспертной системы, однако такой, который можно выделить в самостоятельный. Получены следующие **результаты:** Разработан скриптовый язык для записи правил в базу знаний экспертной системы, использующий аппарат нечеткой логики. Скриптовой язык реализует минимальные возможности для описания правил: описание переменных; запись и обработка логических выражений; определения пространств имен, что позволяет более точно описать определенную ситуацию; предоставить более гибкие средства описания; выполнять логические операции между различными типами данных; компенсировать неоднозначность выводов экспертов, принимая во внимание возможную недостоверность данных. Определены направления дальнейших исследований и работы по совершенствованию и расширению области применения разработанного скриптового языка. **Выводы.** Разработан скриптовой язык для записи правил в базу знаний экспертной системы, использующей аппарат нечеткой логики. На данный момент разработанный скриптовой язык реализует минимальные возможности для описания правил: описание переменных; запись и обработка логических выражений; определение пространств имен, что позволяет более точно описать определенную ситуацию; предоставить более гибкие средства описания; выполнять логические операции между различными типами данных; компенсировать неоднозначность выводов экспертов, принимая во внимание возможную недостоверность данных. Планируется реализовать поддержку конструкций, что позволило бы использовать скриптовый язык отдельно от экспертной системы.

Ключевые слова: экспертная система, скриптовый язык, нечеткая логика, интеллектуальная система, продукционная модель представления знаний.

Development of script language for notation rules in expert system based on fuzzy logic

V. D. Khokh

The **subject matter** is the process of developing expert systems using the apparatus of fuzzy logic and product knowledge base. The **goal** is to develop a scripting language for writing rules into the knowledge base of an expert system using fuzzy logic. A scripting language that implements the minimal possibilities for describing rules, namely: the description of variables writing and processing of logical expressions; definition of namespaces. That would allow: more accurately describe a specific situation; provide more flexible means of description; perform logical operations between different types of data; compensate the ambiguity of the conclusions of experts, considering the possible unreliability of the data. The **tasks** to be solved are: to explore existing, distributed scripting languages, languages that are used to notate the rules into the knowledge base of the expert system, to analyze the existing methods of their application. Develop own language for notating the rules. The language of logical programming, interpreted, integrated into the mechanisms of the ES, but one that can be separated into an independent one. The following **results** were obtained: the script language has been developed to notate rules into the knowledge base of the expert system using the fuzzy logic apparatus. The scripting language implements the minimal possibilities for describing rules: the description of variables writing and processing of logical expressions; definition of namespaces. That allows more accurately to describe a certain situation; provide more flexible means of description; perform logical operations between different types of data; to compensate the uncertainty of the expert's conclusions, considering the possible unreliability of the data. Directions for further research and work on improving and expanding the scope of the developed scripting language are determined. **Conclusions**. A script language has been developed to notate rules into the knowledge base of the expert system based on the fuzzy logic apparatus. At the moment, the developed scripting language implements the minimal possibilities for the description of rules: the description of variables writing and processing of logical expressions; definition of namespaces. This allows more accurate description of a specific situation; provide more flexible means of description; perform logical operations between different types of data; to compensate the ambiguity of the experts' conclusions, considering the possible unreliability of the data. It is planned to implement support for constructs that will allow using the scripting language separately from the expert system.

Keywords: expert system, script language, fuzzy logic, intellectual system, production model of knowledge representation.