УДК 303.732.4

СРАВНЕНИЕ СПОСОБОВ РЕАЛИЗАЦИИ ДЕРЕВА КЛАССИФИКАЦИИ ДЛЯ ПРЕДМЕТНОЙ ОБЛАСТИ, ПРИСУЩИХ РАЗНЫМ ЯЗЫКАМ ПРОГРАММИРОВАНИЯ – ЛОГИЧЕСКОМУ ЯЗЫКУ PROLOG И ОБЪЕКТНО-ОРИЕНТИРОВАННЫМ ЯЗЫКАМ

О.Б. ПОПОВА

Кубанский государственный технологический университет,
350002, Российская Федерация, Краснодар, ул. Московская, д. 2,
электронная почта: popova_ob@mail.ru

Сегодня повсеместно разрабатывают новые методы классификации, способы структурирования деревьев решений и их реализации современным способом. Специально разработанный для реализации баз данных у экспертных систем язык логического программирования считается наиболее удобным для представления систем поддержки принятия решений. Он был проверен на возможность использования для структуры дерева принятия решений, приближенной к естественному интеллекту. Так же в статье сравнивали эффективность реализации данной структуры объектно-ориентированным языком с эффективностью стандартной реализации классификационного дерева в Prolog через продукционные правила. Установлено, что эффективность использования продукционных правил внутри языка Prolog не сильно меняется при изменении способа представления классификационного дерева. Использование продукционных правил, которое представлено в Prolog, менее эффективно, чем использование структуры бинарного дерева, которое предполагает физическое перемещение по нему, используя систему указателей. Поэтому эффективнее использовать для реализации новых структур деревьев классификации объектно-ориентированные языки программирования.

Ключевые слова: логическое программирование, Prolog, предметная область, дерево классификации, продукционное правило.

Сегодня актуальной задачей является составление и решение задач классификации необходимой предметной области, когда формируется база знаний для системы поддержки принятия решений [1, 2]. Иногда приходится разрабатывать совершенно иные методы классификации или способы структурирования деревьев решений [1]. После чего следует реализовать разработанный метод и структуру современным способом. Для этого необходимо сравнить способы реализации деревьев классификации в языках программирования разного типа.

Специально разработанный для реализации баз данных у экспертных систем язык логического программирования считается наиболее удобным для представления систем поддержки принятия решений. Необходимо проверить

его на возможность использования для разработанной Поповой структуры дерева принятия решений [1-3], приближенной к естественному интеллекту. Так же необходимо сравнить эффективность реализации данной структуры объектно-ориентированным языком с эффективностью стандартной реализации классификационного дерева в Prolog через продукционные правила.

В Prolog «основой задач классификации является так называемое решающее дерево». Информация, которая будет представлена в виде знаний, сначала представляется эскизом классификационного дерева. В Prolog оно называется «***иерархической структурой классификационных правил типа «Если – То»***» [4]. Такое дерево не формируется в структуру, как это принято представлять в других языках программирования. Где составляются класс и методы, работающие с этой структурой. В описанных таким образом деревьях, между его элементами создаются связи, преимущественно с использованием указателей.

Классификационные правила языка Prolog позволяют создавать вопросно-ответные системы, системы поддержки принятия решений, основанные на известных методах классификации и построения деревьев решений. Такие правила очень эффективны, если:

- классификационное дерево имеет всего лишь несколько уровней;
- степень исхода узла очень большая;
- проще перечислить подряд все факты об объекте, чем структурировать знания о предметной области;
- перебор всех правил с первого по последний – это самое эффективное решение из возможных.

Семантика языка Prolog создавалась под те методы решения задач и информационные системы, которые были актуальны в момент появления данного языка. Поэтому Prolog и примеры реализации на нём указанных выше систем были на тот момент очень эффективными по сравнению с объектно-ориентированными языками.

Сегодня существуют другие, более эффективные способы представления деревьев решений, например, с использованием энтропии. Так же предлагаются более эффективные методы классификации, которые отражают современные методы структурирования информации, представления структур данных, которые удобны для реализации деревьев решений объектно-ориентированными языками. Это особенно актуально при проектировании программных продуктов, с использованием Windows Form.

Поэтому использование обычных классификационных деревьев с языком Prolog нельзя считать эффективным. При определённых условиях он может быть эффективен, если использовать новые концепции представления классификационных деревьев или деревьев принятия решений.

Если сравнивать реализацию дерева по новой концепции на языке Prolog с реализацией дерева по той же концепции на любом другом объектно-ориентированном языке, то последняя из них будет более эффективной по числу выполненных операций (их будет намного меньше), по объёму каждой операции (каждая будет маленькой) и по реализации логического вывода с объяснением почему. Ниже докажем это на небольшом примере.

Пусть имеется классификационное дерево [4], представленное на рисунке 1. Запишем для него все продукционные правила.

Рисунок 1 – Классификационное дерево

Запишем для него все продукционные правила высокого уровня, используя обратную цепочку рассуждений:

1 animal_is(“тигр”) :– it_is(“млекопитающее”), it_is(“хищник”), positive(“имеет”, “рыжий цвет”), positive(“имеет”, “черные полосы”).

2 animal_is("гепард") :– it_is("млекопитающее"), it_is("хищник"), positive("имеет","рыжий цвет"), positive("имеет", "черные пятна").

3 animal_is("зебра") :– it_is("копытное"), positive("имеет","черные полосы").

4 animal_is("пингвин") :– it_is("птица"), negative("умеет", "летать"), positive("умеет", "плавать"), positive("имеет", "черно-белый цвет").

...

n animal_is("a") :– it_is("b"), negative("c", "d"), positive("e", "f"), positive("g", "h").

и продукционные правила низкого уровня через некоторые соподчиненные предикаты:

1 it_is("млекопитающее") :– positive("имеет", "шерсть"), positive("кормит", "детенышей молоком").

2 it_is("хищник") :– positive("имеет", "острые зубы"), positive("имеет", "когти"), positive("имеет", "глаза, направленные вперед").

3 it_is("копытное") :– it_is("млекопитающее"), positive("имеет", "копыта"), positive("жует", "жвачку").

...

n it_is("z") :– it_is("x"), positive("v", "m"), positive("k", "s").

Теперь предположим самый неэффективный случай поиска для этого дерева – поиск животного с именем "a". Это животное будет определено на n шаге рекурсивного выполнения продукционных правил высокого уровня. Так же может выполняться одно или несколько продукционных правил низкого уровня во время выполнения одного продукционного правила высокого уровня. Пусть среднее число выполняемых продукционных правил низкого уровня за одно выполнение продукционного правила высокого уровня равно $\boldsymbol{k}$. Тогда посчитаем эффективность используемого способа реализации для приведённого примера. Для этого примем за одно действие выполнение любого из элементов продукционного правила:

- it_is(“ ”, …)
- positive(" "," ",…)
- negative(" ", " ",…)

Так же определим число таких элементарных действий за выполнение одного продукционного правила высокого уровня. Как видно из продукционных правил, после знака «***если***» перечисляются подряд узлы классификационного дерева, которые расположены по пути правила между корнем дерева и узлом решения. Получим тогда число этих узлов так

$$N_{\text{узлов}} = h - 2, \quad (1)$$

где h – это высота дерева (будем считать дерево равновесным).

В продукционном правиле высокого уровня так же могут присутствовать описание свойств классифицируемого объекта элементами ***positive(" "," ",…)*** и ***negative(" ", " ",…)***. Пусть их в каждом таком продукционном правиле используется в среднем m элементов.

Следовательно, запишем общее число действий, которое необходимо совершить, чтобы определить животное с именем "a", используя формулу (1):

$$N_{Prolog} = n\,(N_{\text{узлов}} + m + k) = n\,(h - 2 + m + k). \quad (2)$$

Теперь проанализируем эффективность работы структуры бинарного дерева система вопросов и ответов, которая получена Поповой [5-12]. В каждом узле используется лишь одно свойство, которое способно разделить каждое подмножество на две части. Степень исхода узла, поэтому равна 2. Рассмотрим самый худший случай представления дерева – вырожденное левостороннее дерево, когда оно будет иметь высоту равную n (см. рис. 2).

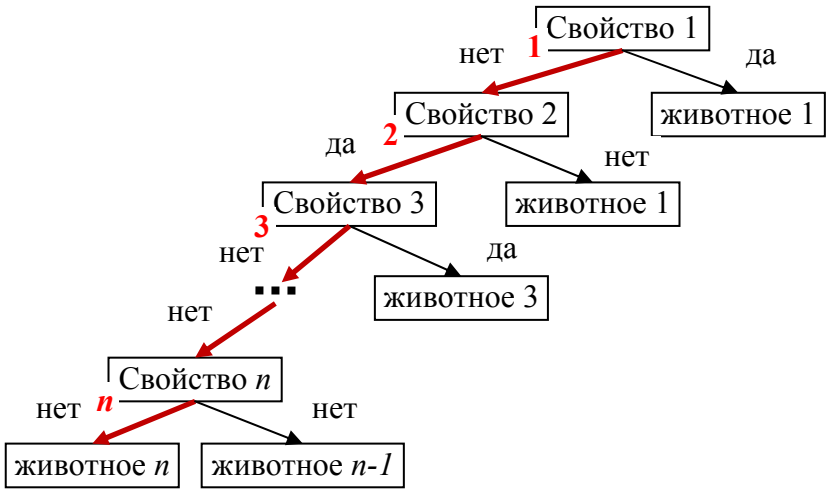

Рисунок 2 – Бинарное дерево системы вопросов и ответов

При определении одного из животных *n* или *n-1* будет совершено самое большое количество сравнений, то есть элементарных действий в дереве. При этом другие пути не будут проверяться, так как это заложено в сам алгоритм. Тем самым достигается экономия по выполненным действиям. Перемещение по дереву организовано физически, в процессе опроса пользователя. Из рисунка 2 (см. красные стрелки, которые образуют путь решения по этому дереву) видно, что их количество будет равно высоте этого дерева:

$$N_{Tree} = h = n. \quad (3)$$

Теперь посчитаем во сколько раз новый способ представления дерева классификации и его реализация эффективнее реализации его продукционными правилами в Prolog, разделив формулу (2) на формулу (3):

$$\frac{N_{\Pr olog}}{N_{Tree}} = \frac{n\,(h - 2 + m + k)}{n} = (h - 2 + m + k). \quad (4)$$

Указанная в формуле (4) величина, как раз характеризует используемые в Prolog особенности – суммарное число фактов, которые используются в одном продукционном правиле высокого уровня и низкого уровня для определения соответствия текущего условия. Даже если $m = k = 1$, а высота дерева $h = 2$ (это самый простой случай задачи на Prolog), то новый метод будет эффективен в два раза. При увеличении сложности задачи (высота дерева и число фактов увеличиваются) эффективность использования нового метода представления дерева классификации увеличивается в $n > 2$ раз. Поэтому использование продукционных правил, которое представлено в Prolog, менее эффективно чем использование структуры бинарного дерева, которое предполагает физическое перемещение по нему, используя систему указателей.

Проверим, как изменится эффективность использования продукционных правил в Prolog, если использовать новые методы представления деревьев классификации.

Используем для этого равновесное дерево (см. рис. 4) с *n* листьями (решениями). В нём так же используется предположения:

- в промежуточном узле лишь одно свойство, которое проверяется;

- степень исхода узла равна 2;
- переход влево вниз по дереву– это ответ «да»;
- переход вправо вниз по дереву– это ответ «нет».

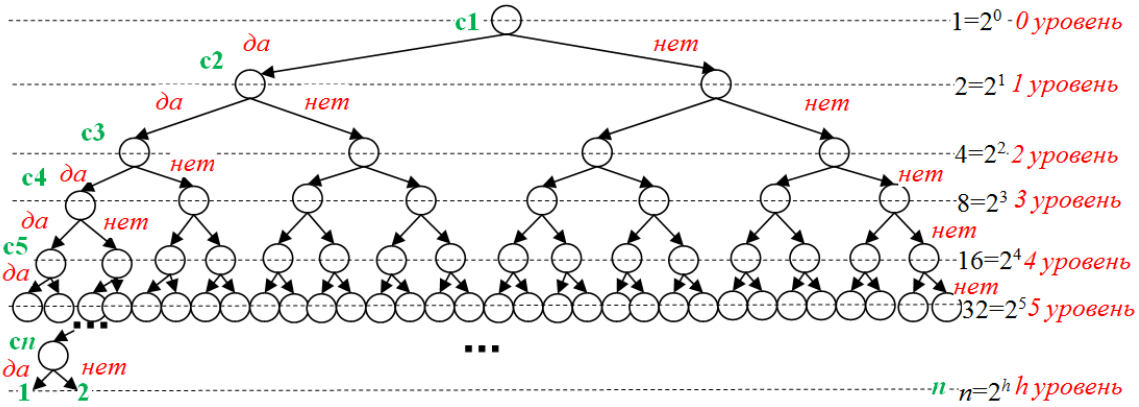

Рисунок 4 – Новый метод представления деревьев классификации

Запишем для дерева рисунка 4 все продукционные правила высокого уровня, используя обратную цепочку рассуждений:

1 animal_is(“1”) :– positive(“c1”), positive(“c2”), positive(“c3”), positive(“c4”), positive(“c5”) …, positive(“c*n*”).

2 animal_is(“1”) :– positive(“c1”), positive(“c2”), positive(“c3”), positive(“c4”), positive(“c5”) …, negative (“c*n*”).

...

n animal_is("*n*") :– negative(“c1”), negative(“c2”), negative(“c3”), negative(“c4”), negative(“c5”) …, negative (“c*n*”).

Продукционные правила низкого уровня использоваться не будут. Как видим, число элементарных операций в одном продукционном правиле будет равно *h*, которое соответствует высоте равновесного дерева. Так как число продукционных правил равно *n*, то количество всех операций при определении животного *n* будет равно:

$$N_{Tree\text{-}Prolog} = n\ h, \quad (5)$$

где h – высота дерева, которую можно вычислить из формулы (см. рис. 4)

$$n=2^h. \quad (6)$$

Прологарифмируем обе части уравнения (6) и получим выражение для *h*:

$$h = \log n. \quad (7)$$

Подставим уравнение (7) в уравнение (5) и получим:

$$N_{Tree\text{-}Prolog} = n \log n. \quad (8)$$

Теперь сравним способ обычной реализации классификационного дерева с предложенным методом Поповой через продукционные правила. Для этого разделим формулу (2) на формулу (8) и проанализируем результат:

$$\frac{N_{Pro\log}}{N_{Tree-Pro\log}} = \frac{n\,(h - 2 + m + k)}{n \log n} = \frac{(h - 2 + m + k)}{\log n}. \quad (9)$$

Поскольку величина $h - 2$ (высота классификационного дерева обычного способа реализации) примерно соотносится с высотой дерева нового метода реализации $h = \log n$, то можно говорить об увеличении эффективности ***на*** какую-то величину $m + k$ (смотри формулу (9)), а не ***во сколько-то раз***, как в приведённом выше примере. Поэтому можно сделать вывод, что эффективность использования продукционных правил внутри языка Prolog не сильно меняется при изменении способа представления классификационного дерева.

Таким образом, чтобы увеличить эффективность рабочей программы, необходимо изменить способ реализации дерева классификации внутри Prolog, отказавшись от применения продукционных правил в существующем сейчас виде, или сменить язык программирования. В языке Prolog существует представление бинарного дерева, которое позволяет сортировать данные, как реализация двоичного справочника. В такой реализации довольно просто представлено заполнение справочника данными и сам вывод данных из дерева [13]. Отдельные компоненты работы с деревом очень просты и понятны, по сравнению со способами реализации других языков.

В Prolog используется «следующее рекурсивное определение бинарного дерева: дерево либо пусто, либо состоит из корня, а также левого и правого поддеревьев, которые в свою очередь также являются деревьями» [13]:

```
1 DOMAINS
2 tree=empty;tr(i,tree,tree)  /* дерево либо пусто, либо состоит из корня
                                 (целого числа), левого и правого поддеревьев,
```

также являющихся деревьями */

Предикат, «позволяющий добавить в двоичный справочник новое значение» [13]. «При этом результирующее дерево должно получиться двоичным деревом. Предикат будет иметь три аргумента. Первым аргументом будет добавляемое значение, вторым – дерево, в которое нужно добавить данное значение, третьим – результат вставки первого аргумента во второй. Решение, конечно, будет рекурсивным» [13].

```
  /* вставляем X в пустое дерево, получаем дерево с X в корневой
  вершине, пустыми левым и правым поддеревьями */
1 tree_insert(X,empty,tr(X,empty,empty)).
   /* вставляем X в дерево со значением X в корневой вершине, оставляем
      исходное дерево без изменений */
2 tree_insert(X,tr(X,L,R),tr(X,L,R)):-!.
   /* вставляем X в дерево с большим X элементом в корневой вершине,
      значит, нужно вставить X в левое поддерево исходного дерева */
3 tree_insert(X,tr(K,L,R),tr(K,L1,R)):- X<K,!, tree_insert(X,L,L1).
   /* вставляем X в дерево с меньшим X элементом в корневой вершине,
      значит, нужно вставить X в правое поддерево исходного дерева */
4 tree_insert(X,tr(K,L,R),tr(K,L,R1)):- tree_insert(X,R,R1).
```

Проблемы возникают при реализации перемещения по дереву решений, автоматизации внесения не числовых данных в дерево и так далее. Тут возникают комплекс проблем, которые связаны с общеизвестным недостатком Prolog [14]. В нём «создание сколько-нибудь сложных и практически полезных Пролог-программ в полностью декларативном стиле практически невозможно, программист вынужден прибегать к процедурным приёмам, что приводит к резкому возрастанию сложности создания и отладки программ, а также плохой контролируемости промежуточных результатов» [15].

Поэтому временная простота представления некоторых компонент покрывается сложностью их совмещения и сочетания при реализации сложных структур данных, которые могут дать серьёзный прирост эффективности при

решении задач классификации. Такая задача может носить свойство олимпиадной или серьёзной научной проблемы, решение которой обусловлено желанием продолжать программировать на языке Prolog.

Работа выполнена при финансовой поддержке Российского гуманитарного научного фонда № 16-03-00382 от 18.02.2016 в рамках темы “Мониторинг исследовательской деятельности образовательных учреждений в условиях информационного общества”.

ЛИТЕРАТУРА

1. Popova O., Popov B., Karandey V., Evseeva M. Intelligence amplification via language of choice description as a mathematical object (binary tree of question-answer system) // Procedia – Social and Behavioral Sciences. – 2015. – V. 214. – P. 897–905.

2. Системный анализ процесса выбора метода оптимизации информационной системы: монография / О.Б. Попова, Б.К. Попов, В.И. Ключко; ФБГОУ ВПО «Кубан. гос. технол. ун-т». – Краснодар: Издательский Дом – Юг, 2012 – 135 с.

3. Попова О.Б., Попов Б.К. Интеллектуальная информационная система выбора «Оптимэль». Патент на изобретение RUS № 2564641 от 27.05.2014.

4. Язык Пролог в задачах и примерах, 2016. http://cs.petrsu.ru/studies/se/2006/webres/01/files/ai/prolog/p4.html#_Toc161423012

5. Попова О.Б. Системный подход к исследованию процесса оптимизации. Деп. в ВИНИТИ №83-В2010 от 17.02.2010.

6. Попова О.Б., Попов Б.К., Ключко В.И. Анализ связей в реальной и технической системах процесса оптимизации // Международный журнал экспериментального образования. – 2013. – №10-2. – С. 405-408.

7. Попова О.Б., Попов Б.К., Ключко В.И. Анализ связей в реальной и технической системах процесса оптимизации // Международный журнал экспериментального образования. – 2013. – №10-2. – С. 405-408.

8. Попова О.Б. Структура технической системы процесса выбора метода оптимизации. Деп. в ВИНИТИ №243-В2012 от 25.05.2012.

9. Попова О.Б. Системный анализ и управление процессом оптимизации. Деп. в ВИНИТИ №256-В2010 от 07.05.2010.

10. Попова О.Б. Участие процесса оптимизации в развитии сложных технических систем. Деп. в ВИНИТИ №257-В2010 от 07.05.2010.

11. Попова О.Б., Попов Б.К. Связи в исследуемой системе процесса оптимизации. Деп. в ВИНИТИ №112-В2012 от 22.03.2012.

12. Попова О.Б., Попов Б.К. Анализ процесса оптимизации. Определение понятий. Деп. в ВИНИТИ №111-В2012 от 22.03.2012.

13. INTUIT. Национальный открытый университет. Основы программирования на языке Пролог, 2016. http://www.intuit.ru/studies/courses/44/44/lecture/1327?page=1

14. Пролог (язык программирования). Материал из Википедии — свободной энциклопедии, https://ru.wikipedia.org/wiki/Пролог_(язык_программирования)

15. Себеста Р.У. Основные концепции языков программирования = Concepts of programming languages. – 5-е изд. – М.: Вильямс, 2001.

REFERENCES

1. Popova O., Popov B., Karandey V., Evseeva M. Intelligence amplification via language of choice description as a mathematical object (binary tree of question-answer system) // Procedia – Social and Behavioral Sciences. – 2015. – V. 214. – R. 897–905.

2. Sistemnyy analiz protsessa vybora metoda optimizatsii informatsionnoy sistemy: monografiya / O.B. Popova, B.K. Popov, V.I. Klyuchko; FBGOU VPO «Kuban. gos. tekhnol. un-t». – Krasnodar: Izdatelskiy Dom – Yug, 2012 – 135 s.

3. Popova O.B., Popov B.K. Intellektualnaya informatsionnaya sistema vybora «Optimel». Patent na izobretenie RUS № 2564641 ot 27.05.2014.

4. Yazyk Prolog v zadachakh i primerakh, 2016. http://cs.petrsu.ru/studies/se/2006/webres/01/files/ai/prolog/p4.html#_Toc161423012

5. Popova O.B. Sistemnyy podkhod k issledovaniyu protsessa optimizatsii. Dep. v VINITI №83-V2010 ot 17.02.2010.

6. Popova O.B., Popov B.K., Klyuchko V.I. Analiz svyazey v realnoy i tekhnicheskoy sistemakh protsessa optimizatsii // Mezhdunarodnyy zhurnal eksperimentalnogo obrazovaniya. – 2013. №10-2. – S. 405-408.

7. Popova O.B., Popov B.K., Klyuchko V.I. Analiz svyazey v realnoy i tekhnicheskoy sistemakh protsessa optimizatsii // Mezhdunarodnyy zhurnal eksperimentalnogo obrazovaniya. – 2013. – №10-2. – S. 405-408.

8. Popova O.B. Struktura tekhnicheskoy sistemy protsessa vybora metoda optimizatsii. Dep. v VINITI №243-V2012 ot 25.05.2012.

9. Popova O.B. Sistemnyy analiz i upravlenie protsessom optimizatsii. Dep. v VINITI №256-V2010 ot 07.05.2010.

10. Popova O.B. Uchastie protsessa optimizatsii v razvitii slozhnykh tekhnicheskikh sistem. Dep. v VINITI №257-V2010 ot 07.05.2010.

11. Popova O.B., Popov B.K. Svyazi v issleduemoy sisteme protsessa optimizatsii. Dep. v VINITI №112-V2012 ot 22.03.2012.

12. Popova O.B., Popov B.K. Analiz protsessa optimizatsii. Opredelenie ponyatiy. Dep. v VINITI №111-V2012 ot 22.03.2012.

13. INTUIT. Natsionalnyy otkrytyy universitet. Osnovy programmirovaniya na yazyke Prolog, 2016. http://www.intuit.ru/studies/courses/44/44/lecture/1327?page=1

14. Prolog (yazyk programmirovaniya). Material iz Vikipedii-svobodnoy entsiklopedii, https://ru.wikipedia.org/wiki/Prolog_(yazyk_programmirovaniya)

15. Sebesta R.U. Osnovnye kontseptsii yazykov programmirovaniya = Concepts of programming languages. 5-e izd. M.: Vilyams, 2001.

COMPARISON OF THE WAYS OF IMPLEMENTATION THE CLASSIFICATION TREE OF THE SUBJECT AREA – INHERENT DIFFERENT PROGRAMMING LANGUAGES - LOGICAL LANGUAGE PROLOG AND OBJECT ORIENTED LANGUAGES

O.B. POPOVA

Kuban State Technological University,
2, Moskovskaya st., Krasnodar, Russian Federation, 350002,
e-mail: popova_ob@mail.ru

Today everywhere develop new methods of classification, ways of structuring trees of decisions and their realization in the modern way. Specifically designed for the implementation of the database for the expert systems, logic programming language is considered the most convenient for presentation of decision support systems. He was tested

for the ability to use for the structure of decision-making tree, close to the natural intelligence. Also in the article are compared the effectiveness of the implementation of this structure by object-oriented language with the efficiency of the standard implementation of the classification tree in the Prolog by production rules. It was found that the efficiency of production rules inside Prolog language is not much changed with a change the method of representing of the classification tree. The use of production rules, which is represented in Prolog, is less effective than the use of a binary tree structure, which involves physical movement on it, using a pointer system. Therefore, efficient to use new structures for implementing the classification tree the object-oriented programming languages.

Key words: logic programming, Prolog, subject area, the classification tree, production rule.