УДК 519.83

*М.А. Рязанов*

# Анализ существующих средств разработки экспертных систем

Экспертные системы (ЭС) появились в результате развития систем с искусственным интеллектом. Необходимость их создания была вызвана острой нехваткой специалистов-экспертов, которые смогли бы в любой момент квалифицированно отвечать на многочисленные вопросы в своей области знаний. Важную роль при создании ЭС играют инструментальные средства. Среди инструментальных средств для создания ЭС наиболее популярны такие языки программирования, как LISP и PROLOG, а также экспертные системы-оболочки (ЭСО): KEE, CENTAUR, G2 и GDA, AT_ТЕХНОЛОГИЯ, предоставляющие в распоряжение разработчика-инженера по знаниям широкий набор для комбинирования систем представления знаний, языков программирования, объектов и процедур.

Кратко остановимся на некоторых вышеприведенных языках программирования. LISP, как следует из его названия, предназначен для обработки списков, состоящих из атомов – абстрактных элементов, представляющих из себя формально неограниченные по длине цепочки символов. Они могут трактоваться как строки в более привычном понимании, числа или представлять собой некие логические структуры с вложенными на неограниченную глубину подсписками в виде иерархических деревьев. Для обработки списков используется функциональная модель, базирующаяся на теории Lambda-исчислений Черча. Фактически программа на LISP представляет из себя набор lambda-функций, при этом работа со списками осуществляется с помощью базового набора примитивов типа CAR/CDR (взять первый элемент списка, который сам может быть списком/получить список без первого элемента). Таких примитивов в минимальном наборе всего 13 штук. С их помощью и, главное, благодаря рекурсивной системе обработки информации LISP позволяет очень компактно описывать функции, для реализации которых на других языках программирования потребовались бы сотни и тысячи строчек кода. Такие задачи, как автоматическое доказательство теорем, понимание естественного языка и окружающего мира, логические исчисления, написание компиляторов, везде, где требуется обработка абстрактной структурной информации, как оказалось, очень удачно описываются и программируются на LISP. Из недостатков данного языка можно выделить сложность освоения и высокую стоимость средств разработки [1].

Программа на языке PROLOG состоит из набора фактов, определенных отношений между объектами данных (фактами) и набором правил (образцами отношений между объектами базы данных). Эти факты и правила вводятся в базу данных. Для работы программы пользователь должен ввести запрос – набор термов, которые все должны быть истинны. Факты и правила из базы данных используются для определения того, какие подстановки для переменных в запросе (называемые унификацией) согласуются с информацией в базе данных. Язык PROLOG, как интерпретатор, приглашает пользователя вводить информацию. Пользователь набирает запрос или имя функции. Выводится значение (истина – *yes*, или ложь – *no*) этого запроса, а также возможные значения переменных запроса, присвоение которых делает запрос истинным (т.е. унифицирует запрос). Хотя выполнение программы на языке PROLOG основывается на спецификации предикатов, оно напоминает выполнение программ на языках LISP или ML. К недостаткам данного языка можно отнести отсутствие механизма прямого вывода [2].

Несмотря на все достоинства вышеперечисленных языков создания экспертных систем, на сегодняшний день на первое место выходит новая разработка – среда CLIPS. Название языка CLIPS – аббревиатура от C Language Integrated Production System. Язык был разработан в Центре космических исследований NASA (NASA's Johnson Space Center) в середине 1980-х гг. и во многом сходен с языками, созданными на базе LISP, в частности OPS5 и ART.

Для создания экспертных систем, как и в любой другой среде, в CLIPS используются две основные конструкции: правила и факты. Факты могут быть как упорядоченные так и неупорядоченные, их называют шаблонами или фреймами. Мощный инструмент создания правил включает в себя возможность задания приоритета правил, которое позволяет пользователю назначать приоритет для своих правил. Объявляемый приоритет должен быть выражением, имеющим целочисленное значение из диапазона от –10000 до +10000. Выражение, представляющее приоритет правила, может использовать глобальные переменные и функции.

CLIPS поддерживает семь различных стратегий разрешения конфликтов: стратегия глубины (depth strategy), стратегия ширины (breadth strategy), стратегия упрощения (simplicity strategy), стратегия

усложнения (complexity strategy), LEX (LEX strategy), МЕЛ (MEA strategy) и случайная стратегия (random strategy). По умолчанию в CLIPS установлена стратегия глубины. Текущая стратегия может быть установлена командой set-strategy (которая переупорядочит текущий план решения задачи, базируясь на новой стратегии). Каждая из данных стратегий позволяет пользователю выбрать способ выбора одного из правил при активации множества правил.

Кратко остановимся на некоторых из них.

*Стратегия глубины.* Только что активированное правило помещается выше всех правил с таким же приоритетом. Например, допустим, что факт-А активировал правило-1 и правило-2, факт-Б – правило-3 и правило-4, тогда, если факт-А добавлен перед фактом-Б, в плане решения задачи правило-3 и правило-4 будут располагаться выше, чем правило-1 и правило-2. Однако позиция правила-1 относительно правила-2 и правила-3 относительно правила-4 будет произвольной.

*Стратегия ширины.* Только что активированное правило помещается ниже всех правил с таким же приоритетом. Например, допустим, что факт-А активировал правило-1 и правило-2 и факт-Б активировал правило-3 и правило-4, тогда, если факт-А добавлен перед фактом-В, в плане решения задачи правило-1 и правило-2 будут располагаться выше, чем правило-3 и правило-4. Однако позиция правила-1 относительно правила-2 и правила-3 относительно правила-4 будет произвольной.

*Стратегия упрощения.* Между всеми правилами с одинаковым приоритетом только что активированные правила размещаются выше всех активированных правил с равной или большей определенностью (specificity). Определенность правила вычисляется по числу сопоставлений, которые нужно сделать в левой части правила. Каждое сопоставление с константой или заранее связанной с фактом переменной добавляет к определенности единицу. Каждый вызов функции в левой части правила, являющийся частью условных элементов :, = или test, также добавляет к определенности единицу. Логические функции and, or и not не увеличивают определенность правила, но их аргументы могут сделать это. Вызовы функций, сделанные внутри функций, не увеличивают определенность правила.

И сравнение заранее связанной переменной ?x с константой, и вызовы функций numberp, < и > добавляют единицу к определенности правила. В итоге получаем определенность, равную 5. Вызовы функций and и + не увеличивают определенность правила.

*Стратегия усложнения.* Между правилами с одинаковым приоритетом только что активированные правила размещаются выше всех активированных правил с равной или меньшей определенностью.

*Стратегия LEX.* Между правилами с одинаковым приоритетом только что активированные правила размещаются с использованием одноименной стратегии, впервые использованной в системе OPS5. Для определения места активированного правила в плане решения задачи используется «новизна» образца, который активировал правило. CLIPS маркирует каждый факт или объект временным тегом для отображения относительной новизны каждого факта или объекта в системе. Образцы, ассоциированные с каждой активацией правила, сортируются по убыванию тегов для определения местоположения правила. Активация правила, выполненная более новыми образцами, располагается перед активацией, осуществленной более поздними образцами. Для определения порядка размещения двух активаций правил поодиночке сравниваются отсортированные временные теги для этих двух активаций, начиная с наибольшего временного тега. Сравнение продолжается до тех пор, пока не останется одна активация с наибольшим временным тегом. Эта активация размещается выше всех остальных в плане решения задачи.

К основным достоинствам данного языка также можно отнести свободное распространение, мультиплатформенность, полную открытую документацию и объектно-ориентированное расширение CLIPS Object-Oriented Language (COOL).

## Библиографический список

1. Джексон, П. Введение в экспертные системы / П. Джексон. – 3-е изд. – М., 2007.

2. Братко, И.М. Алгоритмы искусственного интеллекта на языке PROLOG / И.М. Братко. – М., 2005.