5. Воронцов К. В., Потапенко А. А. Многокритериальная регуляризация вероятностных тематических моделей для улучшения интерпретируемости тем и определения числа тем // Диалог–2014.

6. Chemudugunta C., Smyth P., Steyvers M. Modeling general and specific aspects of documents with a probabilistic topic model // Advances in Neural Information Processing Systems. — MIT Press, 2006. – Vol. 19. – P. 241–248.

УДК 004.738.5

Возможности логического языка для решения задач на основе программной генерации фактов

О.Н. Половикова
АлтГУ, г. Барнаул

Возможности логического языка для решения задач на основе генерации процедуры правил

Логические языки нашли широкое применение во многих прикладных областях, в том числе и для решения специфических задач, где требуется применять узко настраиваемые модели знаний и аппарат получения новых знаний. Среди логических языков особое место занимает язык Пролог (Акторный пролог, Visual Prolog и др. среды), как широко используемый инструмент декларативного логического программирования. Такая применимость обусловлена поддержкой, помимо логического программирования, объектно-ориентированной парадигмы, программирования, управляемое шаблонами.

Программирования на языке логики, во-первых, привлекает к процессу разработки не только программистов и инженеров знаний, но и других специалистов с различными научными интересами, например, математиков. Во-вторых, определяет одно из перспективных направлений использования Пролог-систем – в качестве аппарата для решения логических задач. **В работе анализируется один из подходов построения новых элементов базы знаний – генерацию состояний.** Динамическое формирование множества возможных состояний объекта и способов его поведения позволяет выполнить поиск решения задачи в тех случаях, когда либо все дерево состояний хранить нецелесообразно исходя из специфики решения, либо предварительное построение и хранение такого дерева может привести к комбинаторному взрыву.

Кроме этого этап динамической генерации состояний будет ключевым в решении тех задачах, где пользователь программы управляет

ходом решения, выбор программой соответствующей ветви решения, зависит от цепочки ходов пользователя. Т.е. заранее неизвестны знания, на основе которых будет выстраиваться целевое построение. Программа в этом случаи хранит знания о способах получения множества состояний в виде совокупности правил (процедуры правил), а также знания об одном или нескольких начальных или переходных состоянии объекта или его поведения. При этом совокупность регламентирующих правил для перехода из одного состояния в другое может явно отсутствовать в постановке задачи, в этом случае этапе проектирования базы знаний программы их следует вывести и формализовать, либо получить знания о правилах динамической генерации самих правил. Поскольку можно построить и реализовать обобщенную модель знаний о знаниях предметной области решаемой задачи, то существует возможность динамического построения или генерации не только множества состояний по заранее сформированным правилам, но также и генерации самих правил по которым будут строиться совокупность состояний.

Следующим шагом рассмотрим основную концепцию (без демонстрации кода) построения **решения с генерацией фактов для базы знаний программы.**

Процесс генерации новых состояний можно связать с обучением программы, с динамическим формированием базы знаний. В основе лежит возможности программных сред для добавления фактов и правил к уже имеющимся, которые были получены в качестве результатов работы программы либо её этапов. В данном случае, генерирующим правилом является вся программа.

Продемонстрировать работу механизма генерации фактов с целью обучения программы можно на примерах элементарных вычислительных задач: поиск факториала или чисел Фибоначчи.

Процедура правил и фактов для вычисления чисел Фибоначчи основана на рекурсивном вызове данных процедур и декларируется базовыми средствами Пролог-системы. Полученное программой очередное число Фибоначчи добавляется в начало базы знаний в виде факта. При последующем запуске программы поиск решения осуществляет среди построенных фактов, если соответствующий факт не найден, тогда срабатывает рекурсивное правило. Если сработало правило, то поиск фактов (а затем и правила) выполняется с меньшим числовым значением параметра и новый построенный факт опять добавляется в начало базы знаний. Так как количество рекурсивных вычислений с каждым запуском программы сокращается, происходит экономия сте-

ковой памяти, которая используется для передачи значений переменных в рекурсивном вызове процедур правил.

Сохранение для последующего использования сгенерированных программой фактов требует привлечения дополнительных ресурсов памяти, но позволяет сократить время рекурсивного вычисления. Анализ апробированных на практике примеров решения с генерацией состояний и базы знаний, позволяет говорить об эффективности данного подхода за счет сокращения времени рекурсивного вычисления (сокращается вложенность рекурсии), а также сокращения памяти стека необходимого для работы программы.

Генерация фактов для вычислительного процесса демонстрирует подход построения решения, хотя решение вычислительных задач не является основной областью применения логических языков. Добавлять в базу знаний можно построенные на ходу факты, решая другие задачи, поиск решения которых можно задекларировать логическим языком.

Библиографический список

1. Цуканова Н.И., Дмитриева Т.А. Логическое программирование на языке Visual Prolog. – Горячая Линия – Телеком, 2008. – 144 с.

2. Половикова О.Н. Формализация процесса построения решения с использованием списков для класса логических задач в программах на языке Пролог // Известия Алтайского государственного университета. – Барнаул, 2011. – №1/1 (69). – С. 117–120.

УДК 681.3.08+519.2

Программные и комбинаторные методы анализа случайных изображений

А.Л. Резник, А.А. Соловьев, А.В. Торгов
ИАиЭ СО РАН, Россия, г. Новосибирск

Введение. Для решения многих задач распознавания образов и анализа изображений требуется знание априорной вероятности того, что в процессе генерации случайного поля, формируемого малоразмерными (в идеале – точечными) сигнальными источниками, не будет образовано ни одного локального сгущения. Под локальным сгущением в данном случае понимается компактный фрагмент поля, который имеет ограниченный размер и содержит количество сигнальных объектов, превосходящее некое наперед заданное пороговое значение. В работах [1, 2] показано, как двумерная задача, связанная с отысканием