SUMMARY

A great part of literature as well as offers for continued education on the use of (commercial) software concentrates on the aim to impart to the students how to use a certain program or package of programs and how these programs are structured. Therefore, in most cases illustrating examples are relatively simple and aim at the very beginners (see e. g. [2]). But such a type of examples does not give support a user mainly interested in solving his problems, but not being interested in becoming acquainted with the latest releases. In our opinion it is therefore important to put the problem into the centre of attention and only after that to «resort to the tool box». It is to be welcomed that an ever increasing part of textbooks deal with this aspect (see e. g. [3, 4, 5, 6]).

LITERATURE

1. Computer Based Teaching and Learning Links [Electronic resource]. – Mode of access : http://lorien.ncl.ac.uk/ming/Resources/cal/Cal.htm.
2. *Benker, H.* Mathematik mit dem PC / H. Benker. – Braunschweig : Vieweg, 1994. – 254 p.
3. *Dodson, C. T. J.* Experiments in Mathematics Using Maple / C. T. J. Dodson, E. A. Gonzalez. – Berlin : Springer, 1995. – 465 p.
4. *Varian, H. R.* Computational Economics and Finance. Modeling and Analysis with Mathematica / H. R. Varian (ed.). – New York : Springer, 1996. – 468 p.
5. *Vvedensky, D.* Partial Differential Equations with Mathematica / D. Vvedensky. – Wokingham : Addison-Wesley, 1993. – 465 p.
6. Mathematische Begriffe visualisiert mit Maple für Lehrer und Dozenten / T. Westermann [u. a.]. – Berlin : Springer, 2001. – 129 p.

ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ И ЕГО ИСПОЛЬЗОВАНИЕ ПРИ ОБУЧЕНИИ СТУДЕНТОВ-МАТЕМАТИКОВ

А. Е. Люлькин

Белорусский государственный университет
Минск, Беларусь
E-mail: lulkin@bsu.by

Рассматриваются особенности решения различных задач средствами логического программирования. Анализируются новые возможности современных систем логического программирования. Обсуждаются вопросы преподавания логического программирования студентам-математикам, в частности связь данного предмета с курсом математической логики. Приводится пример применения логического программирования для моделирования логических схем.

Ключевые слова: логическое программирование, логическое моделирование, Пролог.

Логическое программирование и созданные на его основе системы программирования, в частности Turbo Prolog, Visual Prolog и др., находят все более широкое применение как инструментальное средство для решения различных задач с использованием идей и методов теории искусственного интеллекта (ИИ). Логическое программирование позволяет избежать трудоемкой процедуры представления решения задачи в алгоритмической форме на языке программирования, как это делается в процедурном программировании. В этом случае решение задачи сводится к логическому выводу из описания исходной задачи в рамках некоторого логического исчисления. Процедура логического вывода реализуется в соответствующих системах программирования. Однако непосредственное применение логического программирования в ряде случаев затруднено, так как предполагает предварительное описание задачи в рамках исчисления предикатов, причем с ограничениями, присущими конкретной системе логического программирования. Такое описание позволяет определить искомое решение (цель) также в предикатной форме и свести решение к логическому выводу.

Изучение студентами-математиками логического программирования и его математической основы (исчисление предикатов первого порядка) дает возможность продемонстрировать непосредственное применение логического вывода для решения практических задач в отличие от использования исчисления предикатов для построения и анализа аксиоматических теорий.

Применение логического программирования при решении ряда задач [1–3] позволяет в десятки раз сократить длину программы по сравнению с процедурным программированием и избежать непосредственной реализации такой трудоемкой процедуры, как перебор с возвратом. В качестве примеров задач, которые эффективно решаются средствами логического программирования, можно привести следующие:

- обработка списков, в том числе сортировка, объединение, пересечение и др.;
- получение различных перестановок;
- работа с деревьями (возможность создания и обработки рекурсивных типов данных);
- анализ текста;
- разработка экспертных систем и др.

В настоящее время известны реализации логического программирования, например Visual Prolog, которые относятся к универсальным языкам программирования, так как позволяют эффективно решать практически любые задачи. Это достигается за счет обеспечения возможности работы с массивами (бинарные термы и встроенные предикаты для работы с ними в Visual Prolog), включения мощных библиотек предикатов различного назначения и др.

Приведем некоторые новые возможности логического программирования, реализованные в Visual Prolog [1]:

- реализована концепция объектно ориентированного программирования, что облегчает создание сложных программных систем;
- имеются обширные библиотеки предикатов, реализующих математические функции, средства системного программирования, средства для создания графических интерфейсов пользователя и др.;
- интегрированная среда разработки включает средства визуального программирования;
- возможность создания и эффективной работы с собственными базами данных;
- средства для работы с внешними базами данных, имеющими различную архитектуру;
- средства для создания распределенных приложений типа клиент/сервер;

• возможность подключения программ, написанных на других языках (С, ассемблер и др.).

Отметим также, что применение логического программирования позволяет быстро создавать прототипы систем различного назначения для экспериментального исследования и получения качественных оценок предлагаемых решений. В частности, автором создавались прототипы средств моделирования логических схем, с помощью которых оценивалась точность моделирования для различных модификаций методов логического моделирования.

В докладе рассматриваются вопросы использования логического программирования для решения задач моделирования и тестирования логических схем и организации на данной основе научно-исследовательской работы студентов механико-математического факультета БГУ. Выполнение НИРС в данном направлении позволяет более глубоко изучить основные модели и методы анализа логических схем, практические аспекты применения логического вывода для решения различных задач, а также самостоятельно освоить технологию логического программирования.

Построим предикатное описание логической схемы как объекта анализа и тестирования и на его базе будем решать различные задачи анализа и диагностики логических схем. Предикатное описание формулируется с учетом возможности его реализации на языке Пролог. Используемая модель в отличие от таких распространенных описаний дискретных устройств, как булевы функции, конечные автоматы, логические схемы и др., позволяет одинаково эффективно описывать функциональные элементы различной сложности на языке, близком к тому, который используется разработчиками цифровой аппаратуры.

Под конечным предикатом $P(x_1, ..., x_n)$ будем понимать функцию с областью значений {1,0} (или «истина» и «ложь»), а области значений аргументов функции представляют собой конечные множества $X_1, ..., X_n$, где $x_i \in X_i$, $i = 1, ..., n$, т. е. область определения предиката описывается декартовым произведением $X_1 \times X_2 \times ... \times X_n$.

Пусть $V_r = \{v_1, ..., v_r\}$ – алфавит, в котором описываются сигналы в линиях логической схемы. Если некоторый логический элемент схемы реализует функцию $y = f(x_1, ..., x_m)$, заданную в алфавите V_r, то функционирование данного элемента можно описать предикатом $p(x_1, ..., x_m, y)$ следующим образом:

$$p(x_1, ..., x_m, y) = 1 \Leftrightarrow y = f(x_1, ..., x_m),$$
$$p(x_1, ..., x_m, y) = 0 \Leftrightarrow y = f(x_1, ..., x_m).$$

Пусть входам схемы приписаны переменные $x_1, ..., x_n$, внутренним узлам – переменные $y_1, ..., y_t$. Тогда логическую схему можно представить в виде совокупности взаимосвязанных уравнений $y_i = f_i(x_{j1}, ..., x_{jki}, y_{l1}, ..., y_{lti})$, где f_i – функция, реализуемая i-м элементом; $\{x_{j1}, ..., x_{jki}\} \subseteq \{x_1, ..., x_n\}$, $\{y_{l1}, ..., y_{lti}\} \subseteq \{y_1, ..., y_t\}$ – переменные, описывающие значения сигналов на входах i-го элемента. Заменив функции $f_i(x_{j1}, ..., x_{jki}, y_{l1}, ..., y_{lti})$ предикатами $p_i(x_{j1}, ..., x_{jki}, y_{l1}, ..., y_{lti}, y_i)$ так, как было указано выше, мы получим описание логической схемы в виде совокупности предикатов.

Можно использовать также предикаты, описывающие зависимость значения сигнала в заданном узле схемы от значений сигналов на входах схемы, т. е. предикаты вида $p_{yi}(x_1, \ldots, x_n, y_i)$, которые описывают функции $y_i = \varphi_i(x_1, \ldots, x_n)$, реализуемые в узлах схемы. Легко видеть, что данные предикаты можно выразить через предикаты, описывающие функции, реализуемые элементами схемы.

Приведенный способ описания логической схемы совокупностью предикатов отличается от описаний, предложенных автором ранее [4], компактностью (ранее для представления функции, реализуемой логическим элементом, использовались r предикатов, где r – мощность алфавита моделирования; в приведенном описании каждая функция задается од-

ним предикатом), а также возможностью эффективного решения проблемы локальности переменных при задании условий истинности предикатов или правил.

Аналогично может быть выполнено предикатное описание логических элементов с возможностью внесения константных неисправностей. Как известно, для представления значения сигнала в некоторой линии, которой соответствует переменная y_i, с возможностью внесения константных неисправностей в данную линию можно использовать обобщенную переменную $y`_i$. При этом переменная $y`_i$ вычисляется следующим образом: $y`_i = y_i\varphi_i^0 \vee \varphi_i^1$. Здесь булевы переменные φ_i^0 и φ_i^1 используются для внесения неисправностей «константа 0» и «константа 1» соответственно; $\varphi_i^0 = 0$, если вносится неисправность «константа 0», иначе $\varphi_i^0 = 1$; $\varphi_i^1 = 1$, если вносится неисправность «константа 1», иначе $\varphi_i^1 = 0$. Не допускается, чтобы одновременно $\varphi_i^0 = 0$ и $\varphi_i^1 = 1$.

Если некоторый логический элемент реализует функцию $y = f(x_1, ..., x_m)$, то функцию, описывающую данный элемент с возможностью внесения константных неисправностей на входы и выходы элемента, можно представить в следующем виде:

$$y = f(x_1\varphi_1^0 \vee \varphi_1^1, ..., x_m\varphi_m^0 \vee \varphi^1{}_m)\varphi_y^0 \vee \varphi_y^1,$$

где переменные $x_1, ..., x_m$ заменены обобщенными переменными

$$x`_1 = x_1\varphi_1^0 \vee \varphi_1^1, ..., x`_m = x_m\varphi_m^0 \vee \varphi_m^1,$$

а переменные φ_y^0 и φ_y^1 используются для внесения константных неисправностей на выход элемента. Так же, как и в случае функции, реализуемой логическим элементом в исправном состоянии, опишем функцию

$$y` = f`(x_1, ..., x_m, \varphi_1^0, ..., \varphi_m^0, \varphi_1^1, ..., \varphi_m^1, \varphi_y^0, \varphi_y^1),$$

реализуемую элементом с неисправностью, предикатом

$$P`(x_1, ..., x_m, \varphi_1^0, ..., \varphi_m^0, \varphi_1^1, ..., \varphi_m^1, \varphi_y^0, \varphi_y^1, y).$$

Предикатное описание логической схемы с возможностью внесения константных неисправностей на входы схемы, входы и выходы логических элементов представляет собой совокупность предикатов, описывающих входы схемы и логические элементы с возможностью внесения неисправностей. Отметим здесь, что переменные φ_i^0 и φ_i^1 можно ставить в соответствие только тем входам логических элементов, которые непосредственно следуют после разветвления питающих их входов схемы или выходов других элементов. Если разветвление отсутствует, то, очевидно, достаточно рассматривать неисправности лишь на соответствующем входе схемы или выходе логического элемента.

В докладе приводится пример предикатного описания заданной схемы. Найдены и обоснованы условия, которым должны удовлетворять описания предикатов, поставленных в соответствие функциональным элементам, при которых исключаются повторяющиеся решения при использовании механизма логического вывода, реализованного в ПРОЛОГЕ.

ЛИТЕРАТУРА

1. *Адаменко, А.* Логическое программирование и Visual Prolog / А. Адаменко, А. Кучуков. – СПб. : БХВ-Петербург, 2003. – 992 с.
2. *Братко, И.* Алгоритмы искусственного интеллекта на языке PROLOG / И. Братко. – М. : Вильямс, 2004. – 640 с.
3. *Стерлинг, Л.* Искусство программирования на языке Пролог / Л. Стерлинг, Э. Шапиро. – М. : Мир, 1990. – 580 с.
4. *Люлькин, А. Е.* Асинхронное моделирование КМОП-структур на переключательном уровне средствами логического программирования / А. Е. Люлькин // Известия РАН. Теория и системы управления. – 2001. – № 5. – С. 799–804.