А. Е. Люлькин

ИСПОЛЬЗОВАНИЕ ЛОГИЧЕСКОГО ПРОГРАММИРОВАНИЯ ПРИ ОБУЧЕНИИ СТУДЕНТОВ

Рассматриваются возможности логического программирования как инструментального средства реализации современных информационных систем на основе методов искусственного интеллекта. Показывается полезность изучения логического программирования студентами и организации на данной основе научно-исследовательской работы студентов.

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 задачи:

– 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.

Отметим также, что применение логического программирования позволяет быстро создавать прототипы систем различного назначения для экспериментального исследования и получения качественных оценок предлагаемых решений.

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 $x_i \, X_i$, 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 одним предикатом), а также возможностью эффективного решения проблемы локальности переменных при задании условий истинности предикатов.

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.

Литература

1. *Адаменко, А.* Логическое программирование и Visual Prolog / А. Адаменко, А. Кучуков. СПб.: БХВ-Петербург, 2003.

2. *Братко, И.* Алгоритмы искусственного интеллекта на языке PROLOG / И. Братко. М. : Вильямс, 2004.

3. *Стерлинг, Л.* Искусство программирования на языке Пролог / Л. Стерлинг, Э. Шапиро. М. : Мир, 1990.

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.

Люлькин Аркадий Ефимович, доцент кафедры веб-технологий и компьютерного моделирования механико-математического факультета Белорусского государственного университета, кандидат технических наук, доцент, lulkin@bsu.by