УДК 681.3.06

О ПРИЛОЖЕНИЯХ LP-СТРУКТУР В ТЕОРИИ ПРОГРАММИРОВАНИЯ

С. Д. Махортов

Воронежский государственный университет

В предыдущих работах автором были предложены алгебраические системы (**LP-**структуры), формализующие продукционно-логический вывод на основе теории решеток и бинарных отношений. Дальнейшие исследования показали применимость данной технологии в различных областях теории программирования. Цель настоящей статьи – более подробно указать возможности применения LP-структур. Перечисляются задачи в различных областях информатики, описание которых может быть сведено к продукционным системам, моделируемым LP-структурами. К таковым относятся: верификация и оптимизация баз знаний экспертных продукционных систем, представление математических знаний, исследование свойств императивных алгоритмов, автоматизация рефакторинга, упрощение множеств правил условных систем переписывания термов. Соответственно решение этих задач может осуществляться на основе LP-структур или их модификаций.

ВВЕДЕНИЕ

Алгебраические системы являются эффективным средством формального построения и исследования компьютерных программ, относящихся к различным парадигмам и технологиям [1—3]. Это касается и логического программирования, особенно в части представления теорий и знаний. Одним из эффективных подходов к формальному определению и исследованию логических систем является «алгебраизация логики» – алгебраическое описание модели представления знаний и правил вывода. Этот подход позволяет абстрактными математическими методами решать важные задачи, включая верификацию и оптимизацию баз знаний. Классической является теория Линденбаума—Тарского [4], рассматривающая логику высказываний как универсальную алгебру, операции которой соответствуют логическим связкам пропозиционального языка. В качестве примеров алгебраизации исчисления предикатов можно привести полиадические алгебры Халмоша [5] и цилиндрические алгебры [6] (см. также обзор [7]). Однако в силу своей универсальности общая алгебраическая логика не решает некоторых интересных частных проблем, связанных с широко распространенными на практике системами продукционного типа [8].

Для исследования таких систем в работах [9—10] автором были предложены алгебраические структуры, позволяющие рассматривать задачи формализации продукционно-логического вывода с точки зрения теории решеток и бинарных отношений. Идея состоит в определении двух отношений на общем множестве. Первое определяет решетку (задает на ней частичный порядок) и для текущей модели является постоянным. Второе порождается конкретной предметной областью и может подвергаться преобразованиям с целью оптимизации. Рассмотренные отношения названы логическими, поскольку обладают транзитивностью и монотонностью — свойствами, характерными для обычных схем продукционно-логического вывода. Решетку с заданным на ней логическим отношением можно назвать LP-структурой (Lattice- или Logical-Production Structure). При изучении этих структур были получены результаты, позволяющие автоматизировать решение ряда проблем для логических систем продукционного типа: эквивалентные преобразования, оптимизация, верификация, эффективный обратный логический вывод.

Дальнейшие исследования показали применимость данной технологии для целого ряда различных задач в теории программирования. Это объясняется тем фактором, что многие модели в информатике по сути имеют продукционный характер, а структуры представления информации подобно решеткам являются иерархическими.

Цель настоящей статьи — несколько более подробно, чем это было сделано автором в предыдущих исследованиях при разработке самих LP-структур, указать возможности их практического применения. В разделе 1 вводится необходимая терминология, связанная с бинарными отношениями, решетками и продукционными системами. В последующих разделах перечисляется ряд задач в различных областях информатики, описание которых может быть сведено к продукционным системам, моделируемым LP-структурами. Соответственно исследование этих задач может осуществляться на основе LP-структур или их модификаций.

1. ОСНОВНАЯ ТЕРМИНОЛОГИЯ

Бинарное отношение R на произвольном множестве F называется рефлексивным, если для любого $a \in F$ справедливо $(a,a) \in R$; транзитивным, если для любых $a,b,c \in F$ из $(a,b),(b,c) \in R$ следует $(a,c) \in R$. Известно, что существует замыкание R^* произвольного отношения R относительно свойств рефлексивности и транзитивности (рефлексивно-транзитивное замыкание).

Обратная задача — нахождение транзитивной редукции: по данному R ищется минимальное отношение R' такое, что его транзитивное замыкание совпадает с транзитивным замыканием R. Напомним, что для частично упорядоченных множеств различаются понятия минимального элемента (для него нет меньшего элемента) и наименьшего элемента (он меньше всех). В [11] приведен алгоритм построения транзитивной редукции ориентированных графов; показано, что эта задача вычислительно эквивалентна построению транзитивного замыкания, и доказана единственность транзитивной редукции ациклического графа.

Решеткой называется множество с частичным порядком $\leq$ («не больше», «содержится»), на котором для любой пары элементов определены операции $\wedge$ («пересечение») и $\vee$ («объединение»). Элемент $c = a \wedge b$ — это точная нижняя грань элементов a,b, т.е. наибольший элемент решетки, удовлетворяющий неравенствам $c \leq a$ и $c \leq b$. Соответственно $d = a \vee b$ — точная верхняя грань a,b, т.е. наименьший элемент решетки, для которого выполнено $a \leq d$ и $b \leq d$.

Решетка $\mathbb{F}$ называется ограниченной, если она содержит общие нижнюю и верхнюю грани — такие два элемента O,I, что $O \leq a \leq I$ для любого $a \in \mathbb{F}$; полной, если каждое ее подмножество имеет в $\mathbb{F}$ точные нижнюю и верхнюю грани. В полной решетке для любого подмножества элементов (конечного или бесконечного) можно определить (многоместные) пересечение и объединение. Для таких операций используются обозначения $\bigwedge$ и $\bigvee$ (с соответствующими индексами, если они нужны).

Решетка $\mathbb{F}$ называется дистрибутивной, если в ней при любых a,b,c выполняются следующие равенства:

- $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$;
- $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$.

Под дополнением элемента a в ограниченной решетке $\mathbb{F}$ подразумевают элемент $a' \in \mathbb{F}$ такой, что $a \wedge a' = O$ и $a \vee a' = I$. Решетка, в которой любой элемент имеет дополнение, называется решеткой с дополнениями. Дистрибутивная решетка с дополнениями называется булевой. В булевой решетке каждый элемент имеет единственное дополнение

В соответствии с целями данной статьи мы не будем вводить строгое определение LP-структуры, ограничившись ее кратким общим описанием. Под LP-структурой мы подразумеваем алгебраическую систему, представляющую собой решетку с заданным на ней дополнительным бинарным отношением R. Продукционно-логическим отношением называется замыкание R относительно свойств рефлексивности, транзитивности и ряда других свойств, определяемых конкретными приложениями. Одно из таких свойств — дистрибутивность (полная или ограниченная). Неформально оно означает возможность логического вывода по частям и объединения его результатов на основе операций $\wedge$ и $\vee$. Таким образом, подобно транзитивному замыканию и редукции отношения на произвольном множестве, можно говорить о более сложных задачах нахождения логического замыкания и редукции отношения на решетке.

Последующие разделы статьи посвящены возможным применениям LP-структур к системам продукционного типа. В связи с этим введем также терминологию, связанную с экспертными продукционными системами. Указанные системы манипулируют множествами фактов и правил (продукций). *Факт* представляет собой единицу декларативной информации — некоторое суждение о внешнем мире или

состоянии системы. Стандартным представлением факта является триплет вида «объект.атрибут = значение» [12] (например, «термометр.температура = высокая»). В книге [13], описывающей практическую реализацию продукционных систем, триплет редуцируется к паре «объект = значение», при этом атрибут интегрируется в объект, т.е. «термометр.температура» и «термометр.изготовитель» просто считаются разными объектами. Мы в данной статье пойдем дальше, интегрируя в объект и само значение, считая факт независимым элементом общего множества фактов. Это упрощение на данном этапе делается с целью сосредоточиться на основных идеях применения LP-структур. В дальнейшем можно реализовать и более сложную конструкцию фактов, соответствующим образом скорректировав описание LP-структуры.

Во время своей работы продукционная система содержит так называемую *рабочую память*. Это некоторое подмножество фактов, которые на текущий момент считаются выполненными.

Правило (*продукция*) состоит из предпосылки и заключения. *Предпосылка* обычно представляет собой выражение над фактами (например, их конъюнкцию или дизъюнкцию). Предпосылка может быть выполненной (истинной) или невыполненной (ложной) при текущем состоянии рабочей памяти. Если предпосылка верна, то правило может быть применено. *Заключение* — это действие, которое можно осуществить, если верна предпосылка (например, добавить к рабочей памяти некоторый новый факт). *Применение правила* состоит в осуществлении действия заключения. Мы рассматриваем такие приложения, в которых заключение, как и предпосылка, является выражением над фактами, и соответствующее заключению действие интерпретируется как «считать истинным». Таким образом, в нашем случае применение правила означает некоторую модификацию рабочей памяти, обычно — пополнение за счет тех фактов, справедливость которых вытекает из истинности выражения в заключении правила.

Прямым выводом в продукционной системе называется процесс циклического применения правил к содержимому рабочей памяти (его исходное состояние задано в начале работы), и соответственно получение в результате новых фактов, которые считаются справедливыми. Прямой вывод может производиться до тех пор, когда получение новых фактов станет невозможным (при текущем содержимом рабочей памяти не окажется ни одной истинной предпосылки правила, заключение которого способно изменить рабочую память). *Обратный вывод* — это противоположный процесс. В нем по некоторому набору результирующих фактов — *гипотезе*, путем анализа правил в направлении от заключения к предпосылке, подтверждается или опровергается справедливость гипотезы при заданном исходном содержимом рабочей памяти. В процессе обратного вывода содержимое рабочей памяти также меняется.

Ниже мы в качестве возможных приложений теории LP-структур рассматриваем несколько видов продукционных систем, различающихся используемой структурой правила, точнее, выражения для его предпосылки и заключения.

2. ЭЛЕМЕНТАРНАЯ ПРОДУКЦИОННАЯ СИСТЕМА

Простейшую продукционную систему можно представить как множество элементарных фактов F и множество R правил $a \to b$, где $a, b \in F$. Элементарный факт — это не содержащее логических операций (элементарное) суждение. Правило $a \to b$ имеет продукционную семантику «если верно a, то верно b» (например, «если идет дождик, то взять зонтик»; «если температура высокая, то заболел»).

Алгебраический метод исследования такой системы сводится к изучению бинарного отношения R на множестве F. Очевидно, в силу особенностей данной продукционной системы R является рефлексивным и транзитивным. Действительно, для любого $a \in F$ можно сказать, что «если верно a, то верно a». Также при наличии в R двух правил $a \to b$ и $b \to c$ при выводе фактически действует и правило $a \to c$. Например, по правилам «если температура высокая, то заболел» и «если заболел, то на работу не ходить» можно сформировать правило «если температура высокая, то на работу не ходить». Соответствующую данному разделу алгебраическую систему можно считать вырожденным случаем LP-структуры с пустым базовым отношением частичного порядка.

3. ТИПИЧНАЯ ПРОДУКЦИОННАЯ СИСТЕМА

Вряд ли можно назвать исчерпывающим знанием правило из предыдущего раздела «если заболел, то на работу не ходить». Более близкими к истине являлись бы следующие высказывания: «если заболел, сегодня рабочий день, время утреннее, то на работу не ходить, лечиться», «если заболел, сегодня рабочий день, время вечернее, то лечиться», «если заболел, сегодня выходной день, то лечиться».

Поэтому рассмотрим более сложную модель базы знаний, правила которой в качестве предпосылки и заключения могут содержать наборы элементарных фактов ($\{a\}$, $\{a,b\}$, $\{a,b,c\}$, ...). Общий вид продукции в данном случае таков: $\{a_1,...,a_n\} \to \{b_1,...,b_m\}$, где a_i и b_j — факты. Смысл такого правила состоит в следующем: если верны все $a_i, i = 1,...,n$, то верны и все $b_j, j = 1,...,m$.

В этой модели объектами бинарного отношения $\to$ являются не элементарные факты, а подмножества их исходного множества F. Состоящая из таких объектов математическая структура (обозначим ее $\mathbb{F}$) является частным случаем решеток — решеткой множеств (булеаном).

В данном разделе с учетом происхождения рассматриваемой модели мы будем использовать обозначения, принятые в теории множеств, а не теории решеток. Чтобы не путать с объектами простейшей логики п.1, элементы $\mathbb{F}$ будем обозначать большими латинскими буквами. Для любых A,B в $\mathbb{F}$ определены (теоретико-множественные) операции пересечения ($A \bigcap B$) и объединения ($A \bigcup B$). Кроме того, в $\mathbb{F}$ задан частичный порядок — отношение включения $\subseteq$.

Таким образом, на множестве $\mathbb{F}$ мы имеем базовое отношение $\subseteq$, задающее структуру решетки, и дополнительное отношение $\to$, алгебраически отражающее логические связи знаний конкретной предметной области. Ввиду усложнения модели, логические отношения в данной LP-структуре, кроме имевшихся в предыдущем разделе свойств рефлексивности и транзитивности, должны также обладать дополнительными свойствами. Свойство рефлексивности является теперь частным случаем свойства тавтологии включения: при $A \supseteq B$ имеет место $A \to B$. Действительно, если справедливо некоторое множество фактов A, то верно и любое его подмножество B. Кроме того, любую совокупность фактов можно выводить по частям: если $A \to B$ и $A \to C$, то $A \to B \bigcup C$. Это свойство логических отношений мы называем дистрибутивностью. Из него формально следует монотонность такой логики: поскольку $A \to A$, то из $A \to B$ по свойству дистрибутивности получаем $A \to A \bigcup B$. Это означает монотонное накопление знаний при применении правил.

В рассматриваемой модели продукционных систем более сложными являются не только сами объекты продукций (предпосылка и заключение), но и выводимые логические связи. Если в предыдущем разделе «лишние» правила можно было легко определить только на основе свойства транзитивности отношения, то в данном случае для этого требуется решить задачу нахождения логической редукции LP-структуры. Эта задача решена в работе [10]. Кроме того, методами работ [9-10] обратный вывод в продукционной системе можно интерпретировать как решение продукционно-логического уравнения на LP-структуре [14].

4. РАСШИРЕННАЯ ПРОДУКЦИОННАЯ СИСТЕМА

В рамках предыдущего раздела невозможно было бы использовать довольно естественное правило вида «если на улице дождик, то взять зонтик или отложить поход на завтра». Кроме того, хотя в той модели могло присутствовать правило «если дождика нет и прогноз благоприятный, то зонтик не брать», но в процессе работы системы хотелось бы также учитывать, что некоторые фигурирующие в этом правиле факты представляют отрицания фактов из предыдущего правила — нельзя же одновременно взять зонтик и пойти без него.

Таким образом, использования в предпосылках и заключениях правил конъюнкций элементарных фактов явно недостаточно. В литературе описываются продукционные системы [8], в правилах которых могут присутствовать сложные логические выражения, содержащие конъюнкции, дизъюнкции и отрицания. Конечно, теоретически в пропозициональных формулах (если относить к таковым выражения предпосылки и заключения) возможны и импликации, но каждую из них всегда можно исключить с использованием дизъюнкции и отрицания. Таким образом, мы рассматриваем

здесь базы знаний, продукции которых в качестве предпосылки и заключения содержат логические выражения над фактами, построенные с помощью конъюнкций, дизъюнкций и отрицаний. Данную логическую систему можно назвать продукционной системой нулевого порядка [15].

Для такого вида продукционных систем также может быть построена алгебраическая модель в виде LP-структуры. Ее решетка должна поддерживать указанные три операции пропозиционального языка. Таковой является булева решетка Линденбаума—Тарского [4]. Ее элементами являются формулы пропозиционального исчисления. Логическим операциям конъюнкции, дизъюнкции и отрицания соответствуют определенные в п.1 алгебраические операции $\wedge$, $\vee$ и $'$. Для любых двух формул $a, b \in \mathbb{F}$ справедливо $a \le b$, если из истинности формулы a следует истинность формулы b. Эквивалентные логические формулы отождествляются.

Как и в предыдущем разделе, объектами продукционно-логического отношения $\to$ будут являться элементы решетки. Однако в нашем случае решетка имеет более сложную архитектуру. Чтобы адекватно отражать расширенную продукционную логику, логическое отношение должно удовлетворять большему количеству требований по сравнению с предыдущим разделом.

Таким образом, методы LP-структур могут быть применены к исследованию, преобразованию и оптимизации баз знаний продукционных систем с расширенной логикой.

5. ПРОДУКЦИОННАЯ СИСТЕМА ПЕРВОГО ПОРЯДКА

Существуют виды интеллектуальных систем, которые могут потребовать использования в правилах вывода наряду с логическими выражениями также кванторов общности и существования. Такие модели, в частности, могут возникать при разработке систем символьной математики. Большинство математических теорем имеют форму продукции вида «дано» $\to$ «требуется доказать», где предпосылка и заключение являются формулами логики предикатов первого порядка.

Рассмотрим продукционную систему, правила которой в качестве предпосылки и заключения могут содержать формулы исчисления предикатов первого порядка. Как и в предыдущем разделе, будем считать, что эти формулы не содержат импликаций (при необходимости их можно исключить). Будем называть ее продукционной системой первого порядка [16]. Соответственно возникает идея о распространении методов теории LP-структур на такую логику. Это дало бы аппарат для формальных преобразований математических знаний, а также их верификации и оптимизации.

Для этой цели потребуется некоторое алгебраическое описание логических кванторов, которое дополнило бы использованное нами в предыдущем разделе понятие булевой решетки. Как уже говорилось во введении, обзор имеющихся подходов к решению этой проблемы содержится в [7]. Мы воспользуемся алгебраизацией кванторов, предложенной в [4], поскольку она естественным образом расширяет нашу продукционную логику на решетке и содержит достаточные возможности для решения рассматриваемых нами задач.

Согласно этой теории, кванторы общности и существования соответственно моделируются в общем случае бесконечноместными операциями пересечения и объединения. Пусть $P(x)$ — некоторый предикатный символ, определенный на множестве X. Его можно представить совокупностью элементов решетки Линденбаума-Тарского, каждый из которых соответствует формуле $P(x)$ при конкретном значении $x \in X$. Тогда, в силу свойств логических операций, выражение $\forall x P(x)$ алгебраически интерпретируется операцией пересечения всех элементов решетки, соответствующих $P(x)$. Аналогично формула $\exists x P(x)$ может быть представлена объединением всех элементов решетки, порожденных предикатным символом $P(x)$.

Заметим, что не любая решетка допускает выполнение бесконечноместных операций. Чтобы они были возможны и корректны, можно использовать полные булевы решетки. Как говорилось в п. 1, в полной решетке для любого подмножества элементов (конечного или бесконечного) можно определить (многоместное) пересечение и объединение. Нашим формулам $\forall x P(x)$ и $\exists x P(x)$ в полной решетке будут соответствовать элементы $\bigwedge\limits_{x \in X} P(x)$ и $\bigvee\limits_{x \in X} P(x)$.

Приведенные соображения открывают возможности формальных исследований, преобразований и автоматической оптимизации продукционных баз знаний с логикой первого порядка.

В качестве иллюстрации возьмем пример из статьи [17], относящейся к области неклассических уравнений с частными производными. В этой статье в коэффициентах символов псевдодифференциальных операторов используется «весовая» вещественная функция α, существенно влияющая на разрешимость уравнений с этими операторами. В основе результатов работы [17] лежат следующие два условия на данную функцию.

1. $\alpha(t) = 0,\ t \le 0;\ \ \alpha(t) > 0,\ t > 0;$
$\alpha(t) = const,\ t \ge d > 0.$

2. $\alpha|_{R^+} \in C^{\infty}\left(\overline{R}^{+}\right);\ \exists M \ge 0 : \alpha \in C^{M+1}(R).$

Для построения соответствующей LP-структуры введем обозначения логических формул. Пусть $P_{11}(t)$, $P_{12}(t)$ и $P_{13}(t,c)$ — предикатные символы, соответствующие утверждениям $\alpha(t) = 0$, $\alpha(t) > 0$ и $\alpha(t) = c$. Тогда при каждом значении переменной t мы имеем элементы решетки $P_{11}(t)$, $P_{12}(t)$ и каждой паре значений t, c — элемент $P_{13}(t,c)$. Условию 1 соответствуетэлемент $\bigwedge\limits_{t\le 0} P_{11}(t) \wedge \bigwedge\limits_{t>0} P_{12}(t) \wedge \bigvee\limits_{c>0} \bigwedge\limits_{t\ge d>0} P_{13}(t,c)$. Так же для условия 2 введем символы $P_{21}(t)$ и $P_{21}(t,M)$, соответственно означающие «функция α бесконечно дифференцируема в точке t» и «функция α M раз непрерывно дифференцируема в точке t». Алгебраическая запись условия 2 будет выглядеть следующим образом: $\bigwedge\limits_{t\ge 0} P_{21}(t) \wedge \bigvee\limits_{M\ge 0} \bigwedge\limits_{-\infty<t<+\infty} P_{22}(t,M)$. Таким образом, большую часть утверждений работы [17] алгебраически можно представить продукциями, предпосылки которых содержат выражение вида $\bigwedge\limits_{t\le 0} P_{11}(t) \wedge \bigwedge\limits_{t>0} P_{12}(t) \wedge \bigvee\limits_{c>0} \bigwedge\limits_{t\ge d>0} P_{13}(t,c) \wedge$
$\wedge \bigwedge\limits_{t\ge 0} P_{21}(t) \wedge \bigvee\limits_{M\ge 0} \bigwedge\limits_{-\infty<t<+\infty} P_{22}(t,M)$.

6. ПРОДУКЦИОННАЯ МОДЕЛЬ ИМПЕРАТИВНЫХ АЛГОРИТМОВ

Применение вышеизложенной теории ограничено системами с *монотонным* выводом, при котором осуществляется монотонное накопление знаний. Вместе с тем существуют практические задачи искусственного интеллекта, предполагающие не только накопление, но и модификацию получаемых знаний. Эти задачи рассматриваются в ряде работ (например, [18—20]). В данном разделе мы к ним добавим еще одну интересную область — анализ поведенческих свойств и преобразования императивных алгоритмов. Можно сказать, что во время их работы информация не только накапливается, но и часто замещается.

Приведем фрагмент программы на Паскале, вычисляющий максимальный элемент целочисленного массива.

```
var
A: array[1..N] of integer;
Max, i: integer;
begin
 Max := A[1]; i := 2;
 while i <= N do
 begin
  if A[i] > Max then Max := A[i]; i := i + 1
 end
end;
```

Используемые в программе переменные с их значениями можно считать фактами некоторой базы знаний. Тогда инициализация любой переменной может рассматриваться как добавление нового факта к базе знаний. Присвоение же инициализированной переменной нового значения можно интерпретировать как модификацию имеющегося факта.

Попробуем для приведенной выше императивной программы записать эквивалентную логическую программу — исходный набор фактов и совокупность правил логического вывода. Для этой цели аналогично п.3 будем использовать простейший синтаксис вида

если <Факт 1>, <Факт 2>, ..., <Факт K>
то <Действие 1> , <Действие 2>, <Действие M>;

Итак, предположим, что исходная база знаний состоит из следующих фактов: значения всех переменных не определены (*null*), кроме одной вспомогательной переменной *начало*, имеющей значение *да*. Соответствующая логическая программа может иметь вид

если начало = да **то** Max = A[1], i = 2, цикл = да, начало = нет;
если цикл = да, i <= N, A[i] > Max **то** Max = A[i], i = i + 1;
если цикл = да, i <= N, A[i] <= Max **то** i = i + 1;
если цикл = да, i > N **то** цикл = нет;

Возможно, используемый синтаксис правил требует уточнений, а приведенная логическая программа не оптимальна. Для нас существенно то, что она имеет декларативный характер и

дает эквивалентный результат независимо от порядка записи правил.

Классическое понятие решетки оказывается недостаточно выразительным для моделирования немонотонного продукционно-логического вывода. В связи с этим автором были введены некоммутативные решетки как обобщение классических решеток [21]. В новых решетках операция объединения выполняется с частичным замещением одного из операндов. Для построения соответствующих LP-структур на этих решетках рассмотрены логические бинарные отношения. Описана возможность применения этой модели к формальному исследованию логических свойств императивных алгоритмов [22].

7. ПРОДУКЦИОННАЯ МОДЕЛЬ ИЕРАРХИИ

Еще одна группа возможных применений рассматриваемой методики базируется на использовании решетки типов. Как известно, структура типов объектно-ориентированной системы также образует математическую решетку [18]. Для двух типов объединением считается их ближайший общий тип-предок, пересечением — ближайший общий тип-потомок. Семантика частичного порядка и основных операций на такой решетке существенно отличается от случая решетки множеств. J.Sowa доказал [23], что решетки типов не изоморфны решеткам множеств. Поэтому LP-структуры на решетках типов и возможности их применения требуют дополнительных рассмотрений.

Рассмотрим некоторую иерархию типов $\mathbb{F}$ в объектно-ориентированном программировании. Между парами типов могут существовать как минимум два вида связей — наследование (тип наследует атрибуты типа-предка) и агрегация (тип содержит в качестве атрибута представителя другого типа) [24]. На множестве $\mathbb{F}$ введем отношение частичного порядка: если тип b является наследником a (соответственно a — предком b), то $b \le a$. Для любых $a, b \in \mathbb{F}$ определены две операции: пересечение $a \wedge b$ — наибольший общий потомок и объединение $a \vee b$ — наименьший общий предок a, b (первая операция актуальна в системах с множественным наследованием). Для ограниченности полученной решетки добавим к $\mathbb{F}$ два специальных элемента: I — универсальный тип (общий предок, имеется во многих современных системах) и O — фиктивный потомок всех типов.

На решетке $\mathbb{F}$ рассмотрим второе (соответствующее агрегации) отношение R: если объект типа a в качестве атрибута содержится в типе b, то $(b, a) \in R$. Оба отношения ($\le$ и R) имеют общую семантику: в каждом случае, $b \le a$ или $(b, a) \in R$, тип b получает возможности типа a в виде доступа к его атрибутам. Семантически ясно, что это общее отношение «обладания набором возможностей», которое мы обозначаем $\xleftarrow{R}$, обязано быть рефлексивным и транзитивным.

Обсудим еще одно свойство введенных отношений. Пусть для элементов $a, b_1, b_2 \in \mathbb{F}$ справедливо $b_1 \le a$, $b_2 \le a$. Тогда в соответствии с определением решетки имеем $b_1 \vee b_2 \le a$. Это естественное для отношения $\le$ свойство мы называем $\vee$-дистрибутивностью. Посмотрим, что будет означать обладание этим же свойством для отношения $\xleftarrow{R}$. Пусть $b_1 \xleftarrow{R} a$ и $b_2 \xleftarrow{R} a$, т.е. каждый тип b_1 и b_2 обладает возможностями типа a. Тогда в силу предполагаемой $\vee$-дистрибутивности имеем $b_1 \vee b_2 \xleftarrow{R} a$. Это значит, что тип $b_1 \vee b_2$ также обладает возможностями типа a. С точки зрения проектирования типов это не обязательно. Однако в случае, если более одного наследника (b_1 и b_2) содержат одинаковые атрибуты, по принципам рефакторинга [25] нужно «поднять» общие атрибуты, т.е. поместить один такой атрибут в общий тип-предок $b_1 \vee b_2$, после чего каждый b_1 и b_2 получит возможности a в порядке наследования. В данном случае $\vee$-дистрибутивность отношения $\xleftarrow{R}$ означает решение важной задачи — устранение дублирования кода.

Подчеркнем, что мы рассматриваем здесь обобщенную постановку в плане «распределения возможностей» между типами: вариант, когда b_1 и b_2 по каким-либо практическим соображениям содержат в виде атрибутов представителей типа a под различными идентификаторами, нами в расчет не принимается. На практике тип может содержать много атрибутов, но не все они одинаково существенны при построении иерархии.

Заметим далее, что отношение $\xleftarrow{R}$, обладая свойством транзитивности вне зависимости от контекста, не может аналогично во всех ситуациях удовлетворять $\vee$-дистрибутивности, иначе это приведет к некорректным результатам. Для иллюстрации рассмотрим пример

$b \xleftarrow{R} a$ и $a \xleftarrow{R} a$. При дистрибутивности $\xleftarrow{R}$ выполнялось бы $b \vee a \xleftarrow{R} a$. По идеологии ООП тип $b \vee a$ не имеет права что-либо знать о своих наследниках. Поэтому в данной ситуации $b \vee a$, являясь общим предком типов a и b, может обладать возможностями типа a лишь в случае, если он совпадает с a (т.е. $b \le a$). В остальных вариантах соотношение $b \vee a \xleftarrow{R} a$ будет некорректным.

Существуют ситуации, когда выполнение $\vee$-дистрибутивности отношения $\xleftarrow{R}$ теоретически возможно, но нецелесообразно с точки зрения качества кода. Пусть при $b_1 \xleftarrow{R} a, b_2 \xleftarrow{R} a$ элементы $b_1 \vee b_2$ и a имеют непустое пресечение: $(b_1 \vee b_2) \wedge a = d \neq O$, причем $d < b_1 \vee b_2$ и $d < a$. Если в данном случае допустить $(b_1 \vee b_2, a) \in R$, то окажется, что тип d обладает возможностями типа a одновременно по двум линиям: как его наследник и как наследник типа $b_1 \vee b_2$, также имеющего возможности a.

Другая подобная ситуация — наличие конфликта. Пусть имеет место $(b_1, a), (b_2, a), (b_3, a) \in R$, причем $b_1 \vee b_2 \neq b_2 \vee b_3$. Тогда пары $(b_1, a), (b_2, a)$ «конфликтуют» с парами $(b_2, a), (b_3, a)$: если в обоих случаях «поднять» атрибуты, то тип b_2 унаследует атрибут типа a одновременно от двух различных предков — $b_1 \vee b_2$ и $b_2 \vee b_3$, что также ухудшит код.

Одна из возможных здесь стратегий предполагает отказ от «поднятия» общих атрибутов при наличии подобных ситуаций (невыполнение $\vee$-дистрибутивности). Возможны и другие подходы, более тонко учитывающие особенности конкретной системы.

На основе приведенных здесь соображений можно дать строгое определение логического отношения на ограниченной решетке. Оно будет отражать свойство «обладания набором возможностей» в иерархии типов. В силу вышеназванных соображений его свойство дистрибутивности зависит от контекста. Это, в частности, порождает немонотонность логического вывода на таких LP-структурах.

Логическое замыкание произвольного отношения на решетке типов предоставляет все такие пары (b, a), что в типе b доступны возможности типа a. Логическая редукция даст иерархию с минимальным эквивалентным набором связей и в определенном смысле минимальным дублированием кода. Этот формализм позволяет проводить автоматизированные исследования иерархий типов, включая эквивалентные преобразования и верификацию. Он может служить основой для практической реализации (или модернизации) типов.

Решение родственных проблем алгебраическими методами (FCA) предлагается в [26]: элементам некоторого множества классов требуется оптимально назначить наборы атрибутов — элементов другого независимого множества. В соответствии с выбранными назначениями формируется иерархия классов. В нашей постановке, в отличие от [26], атрибуты сами относятся к исследуемой иерархии классов (типов), что усложняет задачу и вряд ли оставляет возможность непосредственного применения формального концептуального анализа.

Также отметим, что мощный аппарат исследования иерархий типов представляют описанные в [27—28] многоуровневые упорядоченно-сортные алгебры. Однако, как справедливо сказано в [19], разработка этих алгебр еще не завершена. Также автору не удалось обнаружить исследований, посвященных редукции многоуровневых упорядоченно-сортных алгебр, которая предусматривала бы вышеуказанное автоматическое устранение дублирования кода.

8. УСЛОВНАЯ ЭКВАЦИОНАЛЬНАЯ ТЕОРИЯ КАК ПРОДУКЦИОННАЯ СИСТЕМА

Еще одним из возможных направлений применения идей и методов LP- структур являются некоторые проблемы эквациональных теорий и систем переписывания термов (СПТ) [29—30]. Эти системы служат математической основой многих исследований в различных областях теории программирования. С их помощью, в частности, решаются задачи символьного упрощения алгебраических выражений [31], автоматического доказательства теорем [32], верификации программ [33] и др.

Важными задачами, связанными с СПТ, являются эквивалентные преобразования и оптимизация их множеств правил. В то время как для обычных СПТ подобные проблемы уже решались в ряде работ [34—35], для условных СПТ они, по-видимому, еще остаются открытыми. Этот факт можно объяснить более сложной структурой правил условных СПТ. Для обычных систем задача минимизации множества правил в конечном счете сводится к транзитив-

ной редукции некоторого бинарного отношения («элиминация транзитивности»). Для условных систем, как и в предыдущих разделах, можно говорить о более сложной задаче нахождения логической редукции.

При определении системы переписывания термов отправной точкой обычно является эквациональная теория, множество правил которой состоит из равенств. Совокупность правил переписывания получается путем «ориентации» равенств и, возможно, их пополнения для достижения свойства конфлюэнтности. Аналогичный подход используется и для условных СПТ [36]. Поскольку обычно именно эквациональная теория является критерием эквивалентности систем переписывания, исследование в этом плане условных СПТ можно начать с рассмотрения эквивалентности условных эквациональных теорий.

Предположим, что эквациональная теория (аналогично [36]) содержит набор позитивно-условных правил вида $s_1 = t_1, \ldots, s_n = t_n : s = t$. Смысл такого правила в следующем: если имеют место все равенства термов $s_i = t_i, i = 1, \ldots, n$, то выполнено и $s = t$. Если вместо термов рассматривать независимые элементы, то задача упрощения множества правил может быть сведена к описанным в п. 3 LP-структурам. Этот же метод применим и в случае равенств между термами, но оптимизация может оказаться лишь частичной.

Рассмотрим пример, полученный модификацией примера из [36]:

1) $x + y = z : s(x) + y = s(z)$;
2) $x + y = z : g(x) + y = g(z)$;
3) $s(x) + y = s(z), g(x) + y = g(z) : f(x) = f(z)$;
4) $x + y = z : f(x) = f(z)$.

Последнее правило здесь является «лишним», т.к. выводится из первых трех. Этот факт может быть обнаружен методами п.3, поскольку для этого не требуется учета особенностей термов. Заменим теперь второе правило следующим:

2) $h(x + y) = h(z) : g(x) + y = g(z)$.

Теперь формальное применение продукционного вывода не поможет исключить «лишнее» правило 4), если не учесть связь $x + y = z : h(x + y) = h(z)$, которая обусловлена свойствами термов.

Таким образом, можно ввести новую LP-структуру — алгебраическую модель условной эквациональной теории. Эта модель должна учитывать не только обычные логические связи, но и взаимозависимости между термами, обусловленные применениями функций и подстановками. Множество правил можно задать как бинарное отношение на решетке — булеане равенств вида $\{s_i = t_i\}$. На этой решетке необходимо ввести дополнительные операции, отражающие использования функциональных символов и подстановок в термах. Такая модель даст теоретическую основу для автоматической минимизации множества правил условной эквациональной теории.

ЛИТЕРАТУРА

1. *Подловченко Р.И.* Иерархия моделей программ // Программирование. — 1981, 2. — С. 3—14.

2. *Нигиян С.А., Аветисян С.А.* О семантике бестиповых функциональных программ // Программирование. — 2002, 3. — С. 5—14.

3. *Замулин А.В.* Алгебраическая семантика императивного языка программирования // Программирование. — 2003, 6. — С. 51—64.

4. *Расёва Е., Сикорский Р.* Математика метаматематики. Пер. с англ. — М.: Наука, 1972. — 591 с.

5. *Halmos P. Algebraic logic,* IV: Equality in polyadic algebras // Trans. Amer. Math. Soc. 86 (1957), P. 1—27.

6. *Henkin L., Tarski A.* Cylindric algebras // Proc. of Symposia in Pure Mathematics II (1961), Lattice theory, P. 83—113.

7. *Nemeti I.* Algebraization of Quantifier Logics; an Overview // Studia Logica L, nos 3/4, (1991) 485—569.

8. *Davis R., King J.* An overview of production systems // Machine Intelligence, vol 8, Ellis Horwood Limited, Chichester, P. 300—332 (1977).

9. *Махортов С.Д.* Логические отношения на решетках // Вестник ВГУ. Серия физика, математика. — Воронеж, 2003, 2. — С. 203—209.

10. *Махортов С.Д.* О редукции логических отношений на решетках // Вестник факультета ПММ: Вып. 5. — Воронеж: ВГУ, 2004. — С. 172—179.

11. *Aho A.V., Garey M.R., Ulman J.D.* The transitive reduction of a directed graph. SIAM J. Computing 1:2, P. 131—137 (1972).

12. *Doorenbos R. B.* Production Matching for Large Learning Systems. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-22942., Carnegie Mellon University (1995).

13. *Сойер Б., Фостер Д.Л.* Программирование экспертных систем на Паскале. Пер. с англ. — М.: Финансы и статистика, 1990. — 191 с.

14. *Махортов С.Д.* Логические уравнения на решетках // Вестник ВГУ. Серия физика, математика. — Воронеж, 2004, 2. — С. 170—178.

15. *Махортов С.Д.* Об алгебраической интерпретации продукционной логики нулевого порядка //

Вестник ВГУ. Серия системный анализ и информационные технологии. — Воронеж, 2007, 1. — С. 56—63.

16. *Махортов С.Д.* Продукционная логика первого порядка и ее алгебраическая интерпретация // Системы управления и информационные технологии. — 2007, № 3(29). — С. 21—26.

17. *Глушко В.П., Махортов С.Д.* Операторы неглавного типа и задача с косой производной // Успехи мат. наук. — 1986, т. 41, вып. 4. — С. 202—203.

18. *Тейз А., Грибомон П. и др.* Логический подход к искусственному интеллекту: от классической логики к логическому программированию. Пер. с франц. — М.: Мир, 1990. — 432 с.

19. *Вагин В.Н., Головина Е.Ю, Загорянская А.А., Фомина М.В.* Достоверный и правдоподобный вывод в интеллектуальных системах / Под ред. В.Н. Вагина, Д.А. Поспелова. — М.: Физматлит, 2004. — 704 с.

20. *Ашинянц Р.А.* Логические методы в искусственном интеллекте. — М.: МГАПИ, 2001. — 204с.

21. *Махортов С.Д.* Некоммутативные решетки и немонотонные логические отношения. // Вестник ВГУ. Серия физика, математика. — Воронеж, 2006, 1. — С. 166—173.

22. *Махортов С.Д.* О сравнении возможностей императивного и логического программирования // Вестник ВГУ. Серия системный анализ и информационные технологии. — Воронеж, 2006, 1. — С. 78—86.

23. *Sowa J. F.* Conceptual Structures: Information Processing in Mind and Machine. Reading, MA: Addison-Wesley, 1984.

24. *Фаулер М., Скотт К.* UML. Основы: Пер. с англ. — СПб: Символ-Плюс, 2002. — 192 с.

25. *Mens T., Tourw'e T.* A Survey of Software Refactoring // IEEE Trans. on Software Engineering. — V. 30(2), February 2004, P. 126—139.

26. *Godin R., Valtchev P.* Formal Concept Analysis-Based Class Hierarchy Design in Object-Oriented Software Development // Formal Concept Analysis, B.Ganter, G.Stumme, R.Wille Eds. Lecture Notes In Computer Science. —Springer Berlin/Heidelberg, 2005, v. 3626. P. 304—323.

27. *Erwing M.* Specifying Type Systems with Multi-Level Order-Sorted Algebra // 3rd Int. Conf. On Algebraic Method. and Software Technol., 1993, P. 179—188.

28. *Erwing M., Guting R.H.* Explicit Graphs in a Functional Model for Spatial databases // Report 110, FernUniversity Hagen. — 1991. Revised Version. 1993.

29. *Schmitt, P.H.* A Survey of Rewrite Systems. In Proceedings of the 1st Workshop on Computer Science Logic (October 12 — 16, 1987). E. Börger, H. K. Büning, and M. M. Richter, Eds. Lecture Notes In Computer Science, vol. 329. Springer-Verlag, London, 235—262.

30. *Dershowitz, N.* A Taste of Rewrite Systems. In Functional Programming, Concurrency, Simulation and Automated Reasoning: international Lecture Series 1991—1992, Mcmaster University, Hamilton, Ontario, Canada P. E. Lauer, Ed. Lecture Notes In Computer Science, vol. 693. Springer-Verlag, London, 199—228.

31. *Buchberger B., Loos R.* Algebraic Simplification. In: Computer Algebra — Symbolic and Algebraic Computation, B. Buchberger, G. E. Collins, R. Loos (ed.), Springer-Verlag, Vienna — New York, 1982, P. 11—43.

32. *Hsiang J.* Refutational theorem proving using term-rewriting systems // *Artif. Intell.* — 1985. — Vol. 25. — P. 255—300.

33. *Воробьев С.Г.* Условные системы подстановок термов и их применение в проблемно-ориентированной верификации программ. Автореферат диссерт. к.ф.-м.н. Новосибирск, ВЦ СО АН СССР, 1987.

34. *Metivier, Y.* About the Rewriting Systems Produced by the Knuth-Bendix Completion Algorithm. Information Processing Letters 16, 1983.

35. *Toyama, Y.* On Equivalence Transformations for Term Rewriting Systems. In Proceedings of the 1983 and 1984 RIMS Symposia on Software Science and Engineering II E. Goto, K. Araki, and T. Yuasa, Eds. Lecture Notes In Computer Science, vol. 220. Springer-Verlag, London, 1986, 44—61.

36. *Dershowitz N., Okada M., Sivakumar G.* Canonical Conditional Rewrite Systems. In Proceedings of the 9th international Conference on Automated Deduction (May 23 — 26, 1988). E. L. Lusk and R. A. Overbeek, Eds. Lecture Notes In Computer Science, vol. 310. Springer-Verlag, London, 1988, 538—549.

Статья принята к опубликованию 25 октября 2007 г.