УДК 004

АЛГЕБРАИЧЕСКИЕ МОДЕЛИ ИЕРАРХИЙ ТИПОВ ДЛЯ ПРОЕКТИРОВАНИЯ И РЕФАКТОРИНГА

С.Д. Махортов[1], М.Д. Шурлин[2]

Воронежский государственный университет
[1]sd@expert.vrn.ru
[2]mshurlin@gmail.com

Аннотация

При проектировании и модернизации объектно-ориентированных информационных систем оказываются полезными алгебраические методы. Такие методы, в частности, могут служить основой для верификации и оптимизации программного кода. В настоящей работе рассматривается класс основанных на решетках алгебраических структур, описывающих иерархию типов в объектно-ориентированном программировании. Исследуются свойства таких структур, включая замкнутость, эквивалентность преобразований, существование логической редукции. Методология предназначена для верификации и модернизации иерархий типов, важным направлением которой является автоматизированное устранение избыточности кода.

Ключевые слова: *иерархия типов, алгебраическая система, проектирование, рефакторинг.*

Введение

Алгебраические системы предоставляют основу формальных исследований компьютерных программ в различных парадигмах [1–2]. Это относится и к логическому программированию, включая широко распространенные на практике системы продукционного типа [3]. В ряде работ (см. [4] и библиографию в ней) была предложена методология алгебраизации продукционно-логических систем на основе бинарных отношений и решеток. Получены теоретические результаты для обоснования эквивалентных преобразований, верификации и оптимизации таких систем. Решетка с заданным на ней дополнительным продукционным отношением названа LP-структурой (lattice production structure).

Исследования показали применимость данной методологии в различных областях теории программирования. В работе [5] впервые установлено, что отношения обобщения и агрегации типов обладают свойствами продукционно-логического вывода. В результате построен класс LP-структур для моделирования иерархий типов в целях рефакторинга – модернизации кода. В этой модели в LP-структуре из решеточных операций в качестве основной использовалась лишь операция объединения. Данное обстоятельство ограничило возможности теории формализацией единственного метода рефакторинга – *поднятия общих атрибутов* (общий обзор известных методов содержится в [6, 7]). Статья [8] развивает теорию LP-структур для иерархий типов. В результате «инвертирования» модели [5] получен новый метод рефакторинга. Суть данного метода, не упоминающегося в классическом перечне рефакторингов [6], состоит в замене нескольких атрибутов класса их общим потомком (*совмещение атрибутов*).

Настоящая работа посвящена алгебраической модели, объединяющей функциональность моделей [5] и [8]. Для нового вида LP-структур рассматривается стандартный круг вопросов: логическое замыкание, его архитектура, эквивалентные преобразования, логическая редукция и способ ее построения. Представленные результаты расширяют возможности исследования и модернизации иерархий типов.

Решение родственных задач алгебраическими методами анализа формальных понятий (FCA) представлены в [9], где элементам определенного множества классов предлагается в некотором смысле оптимально назначить наборы атрибутов – элементов другого независимого множества. В соответствии с выбранными назначениями формируется иерархия классов. В постановке, которая рассматривается авторами настоящей работы, в отличие от [9], атрибуты сами относятся к исследуемой иерархии классов (типов), что усложняет задачу и не оставляет возможности непосредственного применения методов FCA.

В разделе 1 работы приводятся необходимые базовые сведения из теории бинарных отношений и математических решеток. Основная часть раздела содержит предварительное обсуждение рассматриваемых задач на конкретных примерах, а также их приложений. Данное обсуждение служит иллюстрацией и отправной точкой для определения LP-структуры в разделе 2, где формулируются основные теоретические результаты. Их доказательства планируются к опубликованию в математическом издании. Тем не менее, ввиду двойственности построенной модели по отношению к рассмотренной в работе [5], общее представление о методике доказательств может быть получено в [5].

В разделе 3 обсуждаются вопросы компьютерной реализации LP-структур на решетках типов, включая оценки вычислительной сложности решаемых задач.

1 Обозначения и решаемые задачи

Необходимые для чтения статьи сведения о решетках содержатся в [10]. Решеткой называется частично упорядоченное множество F, в котором наряду с отношением $\leq$ («не больше», «содержится») определены также две двуместные операции $\wedge$ («пересечение») и $\vee$ («объединение»), вычисляющие соответственно точную нижнюю и верхнюю грани любой пары $a, b \in F$. Решетка называется ограниченной, если она содержит общие верхнюю и нижнюю грани – такие два элемента O, I, что $O \leq a \leq I$ для любого $a \in F$.

Рассмотрим иерархию типов F в объектно-ориентированной программной системе. Между парами типов могут существовать как минимум два вида связей – наследование и агрегация [6].

Отношение наследования порождает на F частичный порядок: если тип b – потомок a (соответственно a – предок b), то $b \leq a$. Требуется, чтобы для любых $a, b \in F$ были определены две «решеточные» операции: пересечение $a \wedge b$ – наибольший общий потомок; объединение $a \vee b$ – наименьший общий предок a, b.

На решетке F рассмотрим второе, соответствующее агрегации, отношение R: если экземпляр типа a в качестве атрибута содержится в определении типа b, то $(b, a) \in R$. Оба отношения ($\leq$ и R) имеют общую семантику: в каждом случае, $b \leq a$ или $(b, a) \in R$, тип b получает возможности типа a в виде доступа к его атрибутам. Ясно, что это общее отношение «обладания набором возможностей» (обозначим его $\xleftarrow{R}$) обязано быть рефлексивным и транзитивным. Обсудим другие свойства введенных отношений.

Пусть для элементов $a, b_1, b_2 \in F$ справедливо $b_1 \leq a$, $b_2 \leq a$. Тогда по определению решетки [9] имеем $b_1 b_2 \leq a$. Это естественное для отношения $\leq$ свойство (согласно [4]) называется $\vee$-дистрибутивностью. Посмотрим, что будет означать обладание этим же свойством для отношения $\xleftarrow{R}$. Пусть $b_1 \xleftarrow{R} a$ и $b_2 \xleftarrow{R} a$, то есть каждому типу b_1 и b_2 доступны возможности типа a. Тогда в силу предполагаемой $\vee$-дистрибутивности имеем $b_1 b_2 \xleftarrow{R} a$. Последнее означает, что тип $b_1 b_2$ также обладает возможностями типа a. С точки зрения проектирования типов это не обязательно. Однако, если более одного типа-наследника (в данном случае – b_1 и b_2) содержат одинаковые атрибуты, то, согласно принципу рефакторинга [6], целесообразно «поднять» общие атрибуты, то есть поместить один такой атрибут в об-

щий тип-предок b_1b_2, после чего каждый b_1 и b_2 получит возможности a в порядке наследования. В рассмотренной ситуации $\vee$-дистрибутивность отношения $\xleftarrow{R}$ содержит решение актуальной задачи – устранение дублирования кода.

Указанное свойство исследовано в работе [5]. В частности, показано, что отношение $\xleftarrow{R}$, обладая свойством транзитивности вне зависимости от контекста, не может во всех ситуациях удовлетворять свойству $\vee$-дистрибутивности, иначе это приведет к некорректным результатам. Продемонстрируем отмеченное обстоятельство на примерах.

Пример 1. Для иллюстрации рассмотрим пример: $b \xleftarrow{R} a$ и $a \xleftarrow{R} a$. При $\vee$-дистрибутивности $\xleftarrow{R}$ выполнялось бы $ba \xleftarrow{R} a$. Согласно принципам объектно-ориентированного программирования, тип $b \vee a$ не имеет права что-либо знать о своих наследниках. По этой причине 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 (то есть $b \le a$). В остальных вариантах соотношение $ba \xleftarrow{R} a$ окажется некорректным.

Пример 2. Существуют ситуации, когда выполнение $\vee$-дистрибутивности отношения $\xleftarrow{R}$ теоретически возможно, однако нецелесообразно с точки зрения качества кода. Пусть при $b_1 \xleftarrow{R} a$, $b_2 \xleftarrow{R} a$ элементы b_1b_2 и a имеют непустое пресечение: $(b_1b_2)a=d\neq 0$, причем $d< b_1b_2$ и $d<a$. Если в данном случае допустить $(b_1b_2, a)\in R$, то окажется, что тип d обладает возможностями типа a одновременно по двум линиям, а именно – как его потомок и как потомок типа b_1b_2, также имеющего возможности a. Недостаток такого кода состоит в его избыточности – разрыв связи $d < a$ (при $a \neq I$) не приведет к потере функциональности системы типов.

Пример 3. Другая подобная ситуация – наличие конфликта. Пусть имеет место $(b_1,a),(b_2,a),(b_3,a)\in R$, причем элементы b_1, b_2, b_3, b_1b_2, b_2b_3 попарно различны и $(b_1b_2)(b_2b_3)=b_2$. 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_1b_2 и b_2b_3, что также ухудшит код.

В работе [5] формально описаны ситуации подобных коллизий и построена LP-структура с ограниченным свойством $\vee$-дистрибутивности. Данное свойство сохраняется лишь для $\vee$-дистрибутивных и так называемых неконфликтных в R троек (b_1, b_2, a) элементов решетки. Принятая стратегия предполагает отказ от «поднятия» общих атрибутов при наличии описанных ситуаций (невыполнение $\vee$-дистрибутивности).

Как известно, в основанных на решетках алгебраических системах операции объединения и пересечения порождают двойственные свойства (например, законы де Моргана). Подобная закономерность имеет место и в стандартных LP-структурах [4], где наряду с $\vee$-дистрибутивностью бинарных отношений рассмотрено симметричное свойство – $\wedge$-дистрибутивность. Выясним, что означает свойство $\wedge$-дистрибутивности применительно к отношению $\xleftarrow{R}$.

Предположим, что для элементов $a_1,a_2,b\in F$ выполнено $b \le a_1$, $b \le a_2$. Тогда по определению решетки справедливо $b \le a_1a_2$. Такое свойство частичного порядка $\le$ на решетке называется $\wedge$-дистрибутивностью [4]. Распространим его на отношение $\xleftarrow{R}$. Пусть $b \xleftarrow{R} a_1$ и $b \xleftarrow{R} a_2$, то есть тип b обладает возможностями типов a_1 и a_2. В силу предполагаемой $\wedge$-дистрибутивности получим $b \xleftarrow{R} a_1a_2$. Таким образом, тип b также обладает возможностями типа a_1a_2. Как и выше, с точки зрения проектирования типов последнее соотношение обязательным не является. Его семантика такова: если тип имеет два или более различных атрибута, то эти атрибуты можно заменить единственным атрибутом, относящимся к ближайшему общему потомку типов исходных атрибутов. Нетрудно увидеть, что такая реорганизация делает определение типа-контейнера более компактным с сохранением его функциональности.

Данный метод рефакторинга был назван «совмещением атрибутов» [8]. Он применим лишь для иерархий типов с множественным наследованием, однако его актуальность оправдана популярностью языка C++.

Выясним далее вопрос о том, насколько в данной алгебраической модели типов свойство $\wedge$-дистрибутивности отношения $\xleftarrow{R}$ универсально, и не окажутся ли его ограничения двойственными по отношению к описанным выше ограничениям $\vee$-дистрибутивности.

Пример 4. Для иллюстрации рассмотрим случай двойственности по отношению к примеру 1: $b \xleftarrow{R} a$ и $b \xleftarrow{R} b$. При $\wedge$-дистрибутивности отношения $\xleftarrow{R}$ должно быть выполнено $b \xleftarrow{R} ba$. Аналогично исходному примеру 1, тип b не имеет права что-либо знать о своем типе-наследнике $b \wedge a$. По этой причине тип b, являясь предком типа $b \wedge a$, может обладать его возможностями лишь в случае $b \le a$, иначе соотношение $b \xleftarrow{R} ba$ окажется некорректным.

Пример 5. Пусть при $b \xleftarrow{R} a_1$, $b \xleftarrow{R} a_2$ элементы b и a_1a_2 имеют объединение $b(a_1a_2)=d \neq I$, причем $b<d$ и $a_1a_2<d$. Если в данном случае допустить $(b, a_1a_2) \in R$, то окажется, что тип b обладает возможностями другого типа d одновременно по двум линиям, а именно – и как его потомок, и как контейнер типа a_1a_2, также имеющего возможности d в порядке наследования. Недостатки такого кода (подобно примеру 2) заключаются в его избыточности – разрыв связи $a_1a_2<d$ не приведет к потере функциональности системы типов.

Пример 6. Рассмотрим конфликтную ситуацию, двойственную по отношению к примеру 3. Пусть $(b, a_1), (b, a_2), (b, a_3) \in R$, причем элементы a_1, a_2, a_3, a_1a_2, a_2a_3 попарно различны и $(a_1a_2)(a_2a_3)=a_2$. Тогда пары $(b, a_1), (b, a_2)$ «конфликтуют» с парами $(b, a_2), (b, a_3)$. Этот факт означает, что если в обоих случаях совместить атрибуты (применив свойство $\wedge$-дистрибутивности), то тип b получит возможности типа a_2 одновременно посредством двух атрибутов – a_1a_2 и a_2a_3, что также ухудшит код.

В работе [8] формально описаны ситуации, в которых отношение $\xleftarrow{R}$ не сохраняет свойство $\wedge$-дистрибутивности, и построена LP-структура с его ограничением. Данное свойство выполняется только для $\wedge$-дистрибутивных и так называемых неконфликтных в R троек (b, a_1, a_2) элементов решетки. Принятая стратегия предполагает отказ от «совмещения» общих атрибутов при наличии описанных ситуаций (невыполнение $\wedge$-дистрибутивности). Теоретически возможны и другие подходы, более тонко учитывающие особенности конкретных систем.

Анализируя примеры 4–6 в сравнении с соответствующими примерами 1–3, нетрудно прийти к выводу о двойственном характере ограничений свойств $\wedge$-дистрибутивности и $\vee$-дистрибутивности отношения $\xleftarrow{R}$, которые необходимы для адекватного моделирования иерархий типов. Отмеченный факт служит подтверждением естественности разрабатываемых алгебраических моделей, и, в частности, вводимых ограничений дистрибутивности.

Заметим, что с помощью LP-структур рассматривается некоторая обобщенная постановка задач «распределения возможностей» между типами. Варианты, когда типы по каким-либо практическим соображениям содержат в виде атрибутов представителей одних и тех же типов с различными идентификаторами, в расчет не принимаются. На практике тип может содержать много атрибутов, но не все они одинаково существенны при построении иерархии. Кроме того, алгебраические модели в большей степени предназначены для автоматизированного рефакторинга, чем автоматического, то есть окончательное решение о конкретных преобразованиях типов остается за программистом.

2 Основные результаты

На основе представленных выше соображений в данном разделе определяется понятие логического бинарного отношения на ограниченной решетке. Оно отражает свойство «обладания набором возможностей» в иерархии типов.

Логическое замыкание произвольного бинарного отношения на решетке предоставляет все такие пары (b, a), что в типе b доступны возможности типа a. Решив задачу построения логического замыкания, можно автоматизировать верификацию системы типов. К таким задачам, в частности, относится исследование LP-структуры на наличие циклов. Исходя из предметной области, можно также формулировать правила, которым должна удовлетворять система типов, и контролировать их выполнение. Например, возможна проверка системы на наличие необходимых или отсутствие запрещенных логических связей.

Логическая редукция строит иерархию с минимальным эквивалентным набором связей. В моделируемой системе типов данное преобразование соответствует автоматическому применению упомянутых выше двух методов рефакторинга.

Определение 1. Бинарное отношение на решетке называется ограниченно дистрибутивным, если оно ограниченно $\vee$-дистрибутивно [5] и ограниченно $\wedge$-дистрибутивно [8].

Определение 2. Отношение называется логическим, если оно содержит $\leq$, транзитивно и ограниченно дистрибутивно. Логическим замыканием R называется наименьшее логическое отношение, содержащее R.

Два отношения R_1 и R_2, определенные на общей решетке, называются эквивалентными, если их логические замыкания совпадают. Логической редукцией отношения R называется минимальное эквивалентное ему отношение R_0. При этом не требуется вложения $R_0 \subset R$. Можно говорить о некотором отношении R_0 как логической редукции вообще, не относя этот факт к какому-либо другому отношению R. Это означает, что при изъятии любой пары «меньшее» отношение не будет эквивалентно R_0.

Теорема 1. Для произвольного бинарного отношения R на решетке логическое замыкание существует.

Теорема 2. Пусть R – бинарное отношение на решетке. Тогда каждая из следующих операций на R приводит к эквивалентному отношению:

добавление или исключение пары (b,a), если $b \leq a$;

добавление пары (b_1b_2, a), если тройка (b_1,b_2, a) $\vee$-дистрибутивна и неконфликтна в R;

добавление пары (b, a_1a_2), если тройка (b, a_1,a_2) $\wedge$-дистрибутивна и неконфликтна в R;

добавление или исключение пары (b,a) при наличии пар $(b, c),(c, a) \in (R \cup \leq)$, где a,b,c попарно различны.

В [4] и других работах для стандартных структур показано, что логическое замыкание отношения R совпадает с транзитивным замыканием другого отношения $\check{R} \supseteq R$, построенного в виде некоторого многообразия над R. Этот факт позволяет свести ряд вопросов, касающихся логических отношений, к соответствующим проблемам транзитивных отношений. В частности, построение логического замыкания или редукции можно осуществить с помощью быстрых алгоритмов (типа Уоршолла) [11]. В рамках модели, основанной на ограниченной дистрибутивности, также удается разделить процесс построения логического замыкания на этапы дистрибутивного и транзитивного замыканий.

Для отношения R на решетке рассмотрим отношение $\check{R}$, построенное последовательным выполнением следующих шагов:

для каждой неконфликтной $\vee$-дистрибутивной тройки $(b_1, b_2, a)(b_1 \neq b_2)$ добавить к исходному отношению пару (b_1b_2, a);

для каждой неконфликтной $\wedge$-дистрибутивной тройки $(b, a_1, a_2)(a_1 \neq a_2)$ добавить к исходному отношению пару(b, a_1a_2);

к полученному отношению добавить отношение $\leq$.

Заметим, что по теореме 2 отношение $\check{R}$ эквивалентно R.

Теорема 3. Логическое замыкание отношения R совпадает с транзитивным замыканием $\check{R}^*$ соответствующего отношения $\check{R}$.

Изучен вопрос о существовании и построении логической редукции LP-структур на решетках типов. Справедлива следующая теорема.

Теорема 4. Пусть для отношения R построено соответствующее отношение $\check{R}$. Тогда, если для $\check{R}$ существует транзитивная редукция R^0, то отношение $\underline{R^0}$, полученное исключением из R^0 всех пар вида $b \le a$, представляет собой логическую редукцию исходного отношения R.

3 Вопросы реализации

Кратко обсудим некоторые вопросы практической реализации рассмотренных выше LP-структур и, соответственно, вычислительной сложности построения их логического замыкания и редукции. Настоящая работа в основном посвящена теоретическим результатам обоснования решения этих задач. Создание готовых к реализации оптимальных алгоритмов может служить предметом отдельной статьи. Однако краткое обсуждение алгоритмических вопросов поможет составить общее представление о практической применимости теории LP-структур на решетках типов.

Заметим, что разработка алгоритмов на абстрактных решетках и получение оценок их сложности в общем случае не приносят оптимистичных результатов. Например, популярный вид решетки булеан при количестве атомов n состоит из 2^n различных элементов, со всеми вытекающими «алгоритмическими последствиями».

Для общих решеток разработаны методы их представления деревьями и битовыми векторами [12]. В частности, булеан может быть представлен множеством битовых векторов размерности n. Каждому атому решетки соответствует вектор с одной единицей в соответствующей позиции и остальными нулями. Вычисление решеточных операций сводится к вычислению побитовых логических операций сложения и умножения над компонентами векторов. Нельзя сказать, что операция над двумя векторами будет выполняться за константное время, поскольку разрядность компьютера (например, 32 или 64) может оказаться недостаточной для представления битового вектора единственным словом памяти. Тем не менее, сложность операции n / 32 или n / 64 приемлема при общем количестве элементов решетки 2^n.

Однако решетка типов обычно не является дистрибутивной, и число ее элементов относительно невелико. По этой причине *в данном случае* допустимо выбрать в качестве основы анализа сложности величину N – общее число элементов решетки, а также хранить саму решетку и бинарные отношения на ней в виде матриц смежности размера $N \times N$. Таким образом, в рамках настоящей работы можно абстрагироваться от низкоуровневых проблем представления решеток. В рамках такого допущения оказывается справедливой следующая теорема.

Теорема 5. Задачи нахождения логического замыкания и логической редукции LP-структуры на решетке типов решаются за полиномиальное время, не превышающее $O(N^6)$, где N – общее число элементов решетки.

Заключение

Представленные результаты создают теоретическую основу для автоматизированных исследований иерархий типов в объектно-ориентированных информационных системах, включая эквивалентные преобразования, верификацию и оптимизацию. Они могут быть использованы для практического проектирования или модернизации иерархий типов.

Список источников

[1] Подловченко Р.И. Иерархия моделей программ // Программирование. 1981. № 2. – С. 3–14.

[2] Замулин А.В. Алгебраическая семантика императивного языка программирования // Программирование. 2003. № 6. – С. 51–64.

[3] Davis R., King J. An overview of production systems // Machine Intelligence. – Chichester : Ellis Horwood Limited, 1977. Vol. 8. – P. 300–332.

[4] Махортов С.Д., Подвальный С.Л. Алгебраический подход к исследованию и оптимизации баз знаний продукционного типа // Информационные технологии. 2008. № 8. – С. 55–60.

[5] Махортов С.Д. LP-структуры на решетках типов и некоторые задачи рефакторинга // Программирование. 2009. Т. 35. № 4. – С. 5–14.

[6] Фаулер М. Рефакторинг: улучшение существующего кода : пер. с англ. – СПб. : Символ-Плюс, 2004.

[7] Mens T., Tourw'e T. A Survey of Software Refactoring // IEEE Trans. on Software Engineering. 2004. Feb. Vol. 30(2). – P. 126–139.

[8] Махортов С.Д. LP-структуры для обоснования и автоматизации рефакторинга // Программная инженерия. 2010. № 2. – С. 13–25.

[9] Godin R., Valtchev P. Formal Concept Analysis-Based Class Hierarchy Design in Object-Oriented Software Development // Formal Concept Analysis/ Eds. B. Ganter, G. Stumme, R. Wille // LNCS. V. 3626. – Berlin-Heidelberg: Springer-Verlag, 2005. - P. 304–323.

[10] Биркгоф Г. Теория решеток : Пер. с англ. – М.: Наука, 1984.

[11] Aho A.V., Garey M.R., Ulman J.D. The transitive reduction of a directed graph // SIAM J. Computing. 1972 1 : 2. – P. 131-137.

[12] Halib M., Nourine L. Bit-vector encoding for partially ordered sets // LCNS. V. 831. – Berlin-Heidelberg: Springer-Verlag, 1994. – P. 1–12.

Сведения об авторах

Махортов Сергей Дмитриевич, 1958 г. рождения. Окончил Воронежский государственный университет (1980), д. ф.-м. н. (2010, МГУ). Заведующий кафедрой Математического Обеспечения ЭВМ Воронежского государственного университета. Член редколлегий журналов «Программная инженерия», «Вестник ВГУ. Серия системный анализ и информационные технологии», включенных в перечень ВАК, член диссертационного совета Д 212.038.24, председатель Воронежского регионального отделения Российской ассоциации искусственного интеллекта, рецензент IEEE "Transactions on Knowledge and Data Engineering".

Sergey D. Makhortov (b. 1958) graduated from the Voronezh State University (1980), D. of Phys.-Math. Sc. (2010, MSU). Head of Department of Applied and System Software at Voronezh State University. Reviewer for IEEE Transactions on Knowledge and Data Engineering, member of editorial boards of the journals "Software Engineering" and "Proceedings of Voronezh State University. Ser. Systems analysis and information technologies", head of Voronezh Regional Division of the Russian Association for Artificial Intelligence, member of the Dissertation Council D 212.038.24.

Шурлин Максим Дмитриевич, 1987 г. рождения. Окончил Воронежский государственный университет (2009). Преподаватель кафедры Математического обеспечения ЭВМ Воронежского государственного университета, ведущий разработчик компании Murano Software.

Maxim D. Shurlin (b. 1987) graduated from the Voronezh State University (2009). Holding a part-time position of lecturer at Applied and System Software department at Voronezh State University. Senior .NET developer at Murano Software, Inc.