Применение теории LP-структур к формализации методов рефакторинга кода

А. А. Ногих, email: a.nogikh@yandex.ru [1]

[1] Воронежский государственный университет

Аннотация. *Рассматриваются вопросы автоматизации двух методов рефакторинга объектно-ориентированных программ – метод поднятия общих атрибутов и метод совмещения атрибутов. Отражаются и анализируются последние результаты в области применения теории LP-структур к данной задаче. Предлагаются дальнейшие направления исследований LP-структур на решетках типов.*

Ключевые слова: *объектно-ориентированное программирование, рефакторинг, интеллектуальная система, LP-структура, инструментальные средства разработки.*

Введение

В течение стадии сопровождения программные системы модифицируются с целью исправления ошибок, адаптируются к новым требованиям, вынужденно уходят от изначально спроектированной структуры. Этот процесс неизбежно сопровождается ухудшением качества кода, что делает дальнейшие изменения системы все более сложными.

Остановить деградацию кода помогает рефакторинг – процесс такого изменения программной системы, при котором ее внешнее поведение сохраняется, а внутренняя структура улучшается. Ознакомиться с известными методами рефакторинга можно в фундаментальном обзоре [1].

Очевидно, что рефакторинг больших компьютерных программ представляет серьезную проблему. Для ее решения и автоматизации привлекаются различные математические подходы.

В статье [2] показано, что математической основой рефакторинга может служить теория LP-структур, предоставляющая эффективные алгебраические модели для различных систем в информатике. Этот подход позволяет проводить автоматическую оптимизацию типов объектно-ориентированной системы, включающую устранение дублирования кода путем «подъема» общих атрибутов по иерархии

наследования («Pull Up Field» в терминологии [1]) и удаление избыточных атрибутов.

Путем формального применения двойственности операций в LP-структурах в статье [3] был получен новый метод рефакторинга – совмещение атрибутов и описан соответствующий ему класс LP-структур. Данный метод применим только для рефакторинга кода на языках программирования, поддерживающих множественное наследование. Однако, его существование можно оправдать наличием как минимум языка C++, уже десятилетия остающегося в числе высоко востребованных.

В течение последних лет велась исследовательская работа по поиску способов адаптации изначально теоретической и очень абстрактной модели к многообразию возможных на практике программных систем и подходов к их рефакторингу. В данной работе рассматриваются и систематизируются последние достижения в рамках это области, а также описываются направления для дальнейшей деятельности.

1. Базовые понятия теории LP-структур

В статьях [2] и [3] были введены и исследованы LP-структуры на решетках типов. Ниже представлены основные понятия и результаты, необходимые для использования в настоящей работе.

Определение 1. Под LP-структурой подразумевается алгебраическая система, представляющая собой математическую решетку $\mathbb{F}(\leq, \wedge, \vee)$ с дополнительно заданным на ней бинарным отношением $\leftarrow$, обладающим следующими свойствами:

– включение «тавтологий»: $\leq \subseteq \leftarrow$;

– транзитивность: из $a \leftarrow b$ и $b \leftarrow c$ следует $a \leftarrow c$ ($a, b, c \in \mathbb{F}$);

– дистрибутивность (см. ниже).

Выделяется два вида дистрибутивности.

– $\wedge$ -дистрибутивность, при которой из $b \xleftarrow{R} a_1$ и $b \xleftarrow{R} a_2$ следует $b \xleftarrow{R} a_1 \wedge a_2$;

– $\vee$ -дистрибутивность: из $b_1 \leftarrow a$ и $b_2 \leftarrow a$ следует $b_1 \vee b_2 \leftarrow a$.

Между парами типов (классов) в объектно-ориентированном программировании существуют как минимум два вида связей – наследование и агрегация. Множество типов и связи по наследованию моделирует решетка $\mathbb{F}$. Ее элементы соответствуют типам данных: если

тип b является наследником a, то $b \le a$. Элемент $a \wedge b$ соответствует наибольшему общему потомку a и b, в то время как $a \vee b$ обозначает наименьшего общего предка a и b. Вводятся (при их отсутствии) фиктивные общий предок I и общий потомок O. Определенная таким образом решетка называется решеткой типов.

Дополнительно на решетке $\mathbb{F}$ задается бинарное отношение R, соответствующее агрегации: если тип b агрегирует объект типа a, то $(b,a) \in R$. Отметим, что $(\mathbb{F}, R)$ еще не является LP-структурой, поскольку при таком построении нельзя гарантировать, что R будет удовлетворять всем условиям определения 1.

Оба введенных отношения ($\le$ и R) в предметной области имеют общую семантику – один тип получает возможности другого типа доступом к его атрибутам. Семантически ясно, что отношение «обладания набором возможностей» обязано быть рефлексивным и транзитивным. Будем обозначать как $\xleftarrow{R}$ замыкание отношения R относительно 3-х свойств в определении 1.

Понятие дистрибутивности отношения $\xleftarrow{R}$ задает семантику монотонного логического вывода. При наличии $\vee$-дистрибутивности из $b_1 \xleftarrow{R} a$ и $b_2 \xleftarrow{R} a$ следует $b_1 \vee b_2 \xleftarrow{R} a$. Это означает, что если тип b_1 обладает возможностями a и b_2 обладает возможностями a, то и наименьший общий предок b_1 и b_2 также обладает возможностями типа a. Если поместить атрибут a в $b_1 \vee b_2$, то b_1 и b_2 получат его в порядке наследования, что будет соответствовать рефакторингу методом подъема общих атрибутов.

При наличии $\wedge$-дистрибутивности из $b \xleftarrow{R} a_1$ и $b \xleftarrow{R} a_2$ следует $b \xleftarrow{R} a_1 \wedge a_2$. Это означает, что если тип b обладает возможностями типов a_1 и a_2, то он также обладает возможностями типа $a_1 \wedge a_2$. Если поместить атрибут типа $a_1 \wedge a_2$ в b, то тип b получит возможности типов a_1 и a_2 одновременно, независимо от наличия атрибутов a_1 и a_2 в отдельности, что будет соответствовать рефакторингу методом совмещения атрибутов.

Рефакторинг методом совмещения атрибутов схематично изображен на рисунке в виде UML-диаграмм классов. «SomeClass», содержащий атрибут типа «A» и атрибут типа «B», трансформируется в «SomeClass'», содержащий один атрибут типа «C», являющийся

наследником «A» и «B» (в терминах решетки типов это можно записать как $A \wedge B = C$).

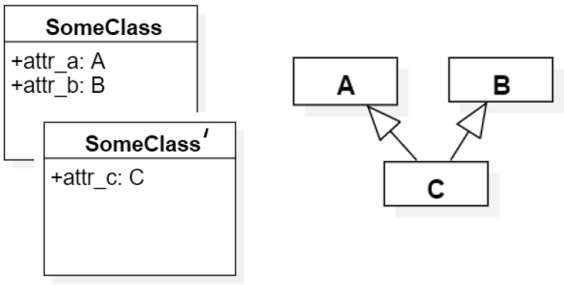

Рисунок. Схематичное изображение рефакторинга методом совмещения атрибутов

Рефакторинг выполняется для улучшения внутренней структуры кода, и недопустимы ситуации с противоположным результатом. В частности, безусловное выполнение дистрибутивности нецелесообразно, так как может привести к совмещению таких атрибутов, которые должны быть раздельными по объективным причинам. Для исключения подобных случаев дополнительно устанавливается ряд ограничений на допустимые преобразования иерархии типов.

В работе [3] была представлена формализация процесса рефакторинга методом совмещения атрибутов на основе применения LP-структур, обладающих $\vee$ -дистрибутивностью.

Ниже перечисляются некоторые из определений, позволивших учесть естественные ограничения в процессе рефакторинга.

Определение 2. Пусть R – отношение на решетке $\mathbb{F}$. Две пары вида $(b_1, a), (b_2, a) \in R$ называются $\vee$ -совместимыми, если существуют такие $c_1, c_2 \in \mathbb{F}$, что $b_1 \vee b_2 \le c_1 \vee c_2$, причем $(c_1 \vee c_2) \wedge a = O$ и пары $(c_1, a), (c_2, a) \in R$ не транзитивны. Назовем (c_1, c_2, a) $\vee$ -дистрибутивной тройкой.

Для описания и предупреждения возможных конфликтов при поднятии атрибутов в разных тройках, было введено понятие неконфликтности $\vee$ -дистрибутивной тройки. Тройка неконфликтна, если для нее не существует ни одной другой $\vee$ -дистрибутивной тройки, такой, что при поднятии атрибута в обеих из них возникнет конфликтная

ситуация. Строгое математическое определение можно найти в оригинальной статье.

Следующие три определения завершают формальное построение LP-структуры на решетке типов.

Определение 3. Назовем отношение R логическим, если оно содержит $\leq$, транзитивно и для любых неконфликтно $\vee$-совместимых пар $(b_1, a), (b_2, a) \in R$ справедливо $(b_1 \vee b_2, a) \in R$.

Определение 4. Логическое замыкание отношения R – наименьшее логическое отношение, содержащее R и его множество неконфликтно $\vee$-совместимых пар.

Определение 5. Два отношения R_1 и R_2, определенные на одной решетке, назовем эквивалентными ($R_1 \sim R_2$), если их логические замыкания совпадают. Логической редукцией отношения R названо эквивалентное ему минимальное отношение R_0.

В работе [2] была доказана теорема о существовании логической редукции LP-структуры на решетке типов и обоснован способ ее эффективного построения.

Симметричные построения, опирающиеся на двойственность операций над LP-структурами, позволили в работе [3] формализовать рефакторинг методом совмещения атрибутов. Набор определений и теорем двойственен описанным выше: $\wedge$-совместимые тройки, неконфликтно $\wedge$-совместимые пары и т.д.

2. Дополнения к модели

Несмотря на то, что семантика элементов LP-структуры на решетке типов четко сформулирована в работах [2] и [3], преобразование в нее структуры классов реальной программной системы не является тривиальной задачей. Обе двойственные формализации рефакторинга приводят к схожим трудностями при попытке их практического применения. В данном разделе описаны дополнения, предложенные к ним.

Существенным препятствием на пути к практическому применению является используемое в оригинальной модели представление агрегации как обладание одного типа возможностями другого. Такой подход допустим, но существует и множество ситуаций, когда тип обладает возможностями некоторого другого типа одновременно в различных «контекстах», не связанных друг с другом.

В качестве решения данной проблемы в работах [4] и [5] было предложено использовать расширенную решетку типов. Она отличается тем, что включает в себя не только типы, но и элементы,

представляющие атрибуты (более точно, «контексты» использования атрибутов).

Расширенная решетка типов строится следующим образом:

– Для каждого типа $t \in \mathbb{F}$ множество $A_t \subseteq A$ атрибутов, имеющих тип t, необходимо разбить на непересекающиеся подмножества $\{A_t^{(i)}\}_{i=1}^{N}$. Логика разбиения такова: атрибуты, входящие в одно подмножество, принадлежат к одному «контексту» использования, т.е. их допустимо объединить в общем типе-предке или заменить атрибутом-потомком.

– Для каждого $A_t^{(i)}$ из $\{A_t^{(i)}\}_{i=1}^{N}$ в решетку типов вносится элемент $t_{agg}^{(i)}$. При рефакторинге методом подъема общих атрибутов, как потомок типа t ($t_{agg}^{(i)} \le t$). При рефакторинге методом совмещения атрибутов, как предок типа t ($t \le t_{agg}^{(i)}$).

– Обозначим множество таких внесенных элементов $\tilde{T}$ и введем вспомогательное отображение $f_{agg} : A \to \tilde{T}$. Если для $A_t^{(i)}$ в решетку типов был внесен $t_{agg}^{(i)}$, то $\forall a \in A_t^{(i)} : f_{agg}(a) = t_{agg}^{(i)}$.

– Бинарное отношение R, соответствующее агрегации, выглядит следующим образом: $\forall a \in A : \left(f_{agg}(a), f_{type}(a) \right) \in R$, где $f_{type}(a)$ – тип атрибута a.

В работах [4] и [5] было показано, что данное построение обладает следующим важным свойством.

Логическая редукция R_0 отношения R не будет содержать таких пар $(t, a) \in R_0$, что $t \in \tilde{T}$. Это утверждение соответствует тому, что в результате рефакторинга искусственно добавленные элементы $\tilde{T}$ не будут агрегировать никакие другие типы.

Исходная модель ограничивает совмещение (или перемещение) атрибутов только во избежание ряда учтенных при ее формировании потенциально нежелательных ситуаций. Для обоих рассматриваемых в данной работе методов рефакторинга предложены дополнительные механизмы контроля. Это существенно повышает гибкость автоматизированного рефакторинга.

В обоих случаях данный параметр принял вид предиката $P : \mathbb{F} \times \mathbb{F} \to \{0,1\}$. Для моделируемой иерархии типов равенство

$P(t,a)=0$ означает запрет типу t после рефакторинга агрегировать экземпляр типа a.

Следующее утверждение было доказано в [5] для LP-структур на решетках типов, обладающих $\wedge$-дистрибутивностью, а в [4] для LP-структур на решетках типов, обладающих $\vee$-дистрибутивностью.

Пусть $\tilde{R}$ – логическое отношение с предикатом P, а R_0 – его логическая редукция. Тогда R_0 не содержит пар (t,a), для которых $P(t,a)=0$.

Предикат $P(t,a)$ гарантированно подействует только на непосредственное владение атрибутом. Чтобы запретить типу t получить атрибут типа a как напрямую, так и в порядке наследования, необходимо назначить $P(t',a)=0$ для всех $t': t \le t'$.

3. Процедура автоматизированного рефакторинга

Ниже описывается обобщенный процесс проведения рефакторинга объектно-ориентированной программы на основе рассматриваемой модели.

Шаг 1. Построение и настройка модели.

В первую очередь данный этап подразумевает формирование решетки типов $\mathbb{F}$ и на ней бинарного отношения R.

При обработке программных систем, где иерархия типов не может быть целиком представлена математической решеткой, следует выбрать подмножество значимых для осуществления рефакторинга типов.

Настройка модели может включать, как минимум, контроль над допустимостью совмещения атрибутов (при использовании расширенной решетки типов) и контроль над допустимым размещением атрибутов (при использовании предиката $P(t,a)$).

Шаг 2. Получение новой иерархии типов.

Можно выделить по крайней мере два способа получения новой иерархии типов.

Полная оптимизация подразумевает как выполнение требуемого вида рефакторинга, так и устранение избыточности из иерархии типов. Данный способ осуществляется путем вычисления логической редукции отношения R.

Для осуществления только рефакторинга методом поднятия общих атрибутов, необходимо для каждой $\vee$-дистрибутивной тройки (c_1,c_2,a) поместить атрибут типа a в $c_1 \vee c_2$, а затем найти транзитивную редукцию полученного отношения.

Для осуществления только рефакторинга методом совмещения атрибутов необходимо для каждой $\wedge$ -дистрибутивной тройки (b, c_1, c_2) поместить атрибут типа $c_1 \wedge c_2$ в b и при этом удалить атрибуты, чей тип x соответствует $c_1 \wedge c_2 < x$, из типа b.

Шаг 3. Построение соответствия между прежней и новой иерархией типов.

Помимо получения размещения атрибутов в новой иерархии, необходимо дополнительно решить следующие практические задачи.

– Обновление дефиниций. Для каждого атрибута a новой иерархии необходимо получить множество атрибутов прежней иерархии типов, которые были совмещены в a. Это позволит учесть свойства имевшихся атрибутов при генерации кода для новых дефиниций.

– Обновление ссылок. В исходном коде программной системы необходимо заменить обращения к атрибутам исходной иерархии, которые были удалены или перемещены, на обращения к атрибутам новой иерархии типов.

В работе [4] было показано, как получить необходимы отображения из результатов шага два при проведении рефакторинга методом подъема общих атрибутов. Для метода совмещения атрибутов способы построения таких отображений были представлены и обоснованы в работе [5].

В публикации [6] приводится детальное описание возможного процесса рефакторинга методом подъема общих атрибутов.

4. Дальнейшие исследования

В качестве возможных направлений для дальнейших исследований по теме можно выделить следующие.

– При формировании множества неконфликтных $\vee$ -дистрибутивных (или $\wedge$ -дистрибутивных) троек берутся только те тройки, для которых не существует ни одной нейтрализующей. Для получения множества дистрибутивных троек, из которых ни одна не является нейтрализующей для другой, возможны и другие подходы.

– Выявление других нежелательных ситуаций, которые модель должна избегать при осуществлении преобразований иерархии типов.

– Исследование способов автоматизированного построения расширенной решетки типов на основе анализа исходного кода программы.

5. Заключение

В данной работе рассматривались подходы к автоматизации рефакторинга, основанные на применении теории LP-структур. В

рамках одной публикации отражены результаты как начальных работ по теме LP-структур на решетках типов, так и последующих научных трудов, опубликованных как в отечественных, так и в зарубежных журналах.

Представленная алгебраическая структура обладает гибкими возможностями формального описания и преобразования иерархий типов с учетом имеющихся знаний о программной системе. Это обстоятельство позволяет рассматривать такую модель в качестве основы для построения интеллектуальных инструментальных средств автоматизации рефакторинга.

Дальнейшие перспективы в рамках данной тематики связаны как с практической реализацией соответствующих инструментальных систем, так и с совершенствованием самой модели.

Работа выполнена при финансовой поддержке РФФИ в рамках научного проекта № 19-07-00037.

Список литературы

1. Фаулер, М. Рефакторинг: улучшение существующего кода / М. Фаулер. – Пер. с англ. – СПб: Символ-Плюс, 2003. – 432 с, ил.

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

3. Махортов, С. Д. LP-структуры для обоснования и автоматизации рефакторинга в объектно-ориентированном программировании / С. Д. Махортов // Программная инженерия. – 2010. – № 2. – С. 15-21.

4. Махортов, С.Д. Автоматизация рефакторинга на основе алгебраического представления знаний / С. Д. Махортов, А. А. Ногих // Материалы VII Международной конференции «Знания – Онтологии – Теории» (ЗОНТ-2019). – Новосибирск : Институт математики им. С.Л. Соболева СО РАН, 2019. – С. 257-263.

5. Makhortov, S. D. Application of LP Structures Theory to Intelligent Attribute Merger Refactoring / S. D. Makhortov, A. A. Nogikh // Proceedings of the 18th Russian Conference on Artificial Intelligence (RCAI 2020). Lecture Notes in Artificial Intelligence. – Cham : Springer, 2020. – V. 12412. – P. 437-447.

6. Махортов, С. Д. Применение LP-структур для автоматизации рефакторинга объектно-ориентированных программ / С. Д. Махортов, А. А. Ногих // Программная инженерия. – 2019. – № 5. – С. 195-203.