УДК 510.56; 519.178; 519.179

С.Н. Баранов, Б. Буавер, С.В. Соловьев, Л. Феро

НЕКОТОРЫЕ ПРИЛОЖЕНИЯ λ-ИСЧИСЛЕНИЯ С ТИПАМИ К АТРИБУТНЫМ ВЫЧИСЛЕНИЯМ В СИСТЕМАХ КАТЕГОРНЫХ ПРЕОБРАЗОВАНИЙ ГРАФОВ

Баранов С.Н., Буавер Б., Соловьев С.В., Феро Л. **Некоторые приложения λ-исчисления с типами к атрибутным вычислениям в системах категорных преобразований графов.**

Аннотация. Рассматриваются преобразования моделей на основе преобразований атрибутных графов. Исследуется подход одинарного универсального квадрата (single pushout) для применения правил преобразования в категории атрибутных графов. Преобразования атрибутов задаются при помощи λ-термов в λ-исчислении с индуктивными типами. Предлагаются решения, позволяющие работать с конструкцией одинарного универсального квадрата для преобразования структуры графа и атрибутных вычислительных функций. Использование индуктивных типов повышает выразительность и эффективность атрибутных вычислений по сравнению с известным подходом на основе Σ-алгебр. Приводится ряд примеров, демонстрирующих особенности предлагаемого подхода.

Ключевые слова: теория категорий, универсальный квадрат, λ-исчисление, переписывание атрибутных графов.

Baranov S., Boisvert B., Soloviev S., Féraud L. **Applications of typed λ-terms to categorical attributed graph transformations.**

Abstract. This paper deals with model transformation based on attributed graph rewriting. Our contribution investigates a single pushout approach for applying the rewrite rules. The computation of graph attributes is obtained through the use of λ-terms in a typed λ-calculus with inductive types. In this paper we present solutions to cope with single pushout construction for the graph structure and the computations functions. As this rewrite system uses inductive types, the expressive power of attribute computations increases and appears to be more efficient than the one based on Σ-algebras. Some examples showing the interest of our computation approach are described in this paper.

Keywords: category theory, single pushout, λ-calculus, attributed graph rewriting.

1. Введение. В настоящее время для технологии программирования на основе моделей требуется строгое теоретическое обоснование. Модели в таких технологиях, как правило, описываются с помощью графических языков (UML, SDL, ...) и состоят из структурной части, которая может быть представлена в виде графа, и атрибутов, представляющих собой информацию, присоединенную к вершинам или ребрам такого графа. Таким образом, модели мо-

гут быть формализованы как атрибутные графы, а преобразования моделей – как преобразования атрибутных графов. Преобразование атрибутного графа состоит из переписывания его структурной части и вычисления его атрибутов, и для выражения этих двух видов преобразования нужен формальный каркас.

Для преобразования структурной части графа широко используются системы переписывания графов на базе теории категорий. Проблему для систем переписывания атрибутных графов представляет реализация атрибутных вычислений. Большинство известных систем на базе теории категорий принимают стандартный алгебраический подход, в котором атрибуты используют алгебраические типы данных, представляемые Σ- алгебрами [5, 8]. Однако реализация вычислений над алгебраическими типами данных встречает ряд трудностей, поэтому из соображений эффективности и простоты в использовании такие системы, как правило, не реализуют атрибутные вычисления целиком, а опираются на программы, написанные на каком-либо языке, зависящим от платформы.

В более ранней работе [9,10,12] предлагалось использовать индуктивные типы и λ-термы в сочетании с некоторой модификацией подхода с двойным универсальным квадратом [11] , для которого применяется английская аббревиатура DPoPb ("double pushout-pullback" approach). Нашей целью было использовать хорошо разработанный подход с двойным универсальным квадратом для реализации переписывания структурной части атрибутных графов, а также применить выразительную силу λ-термов и индуктивных типов для описания атрибутов и упрощения их вычисления. Однако конструкция двойного универсального квадрата накладывает определенные ограничения на функции вычисления, в основном из-за использования всюду определенных отображений и необходимости разбивать вычисления на два этапа. Именно поэтому мы представляем здесь новый подход на основе одинарного универсального квадрата.

В первом разделе данной работы дается введение в основные подходы к переписыванию графов на базе теории категорий и в частности подход с одинарным универсальным квадратом, на котором основан наш подход. Во втором разделе дается определение категории атрибутных графов и объясняется, как применять правило переписывания через построение слабого универсально-

го квадрата. В заключение приводятся два примера, в которых описываемый подход сравнивается с подходами на базе двойного универсального квадрата.

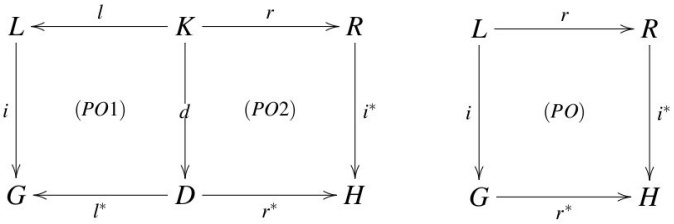
2. Категорный подход к преобразованиям графов. В системах преобразования графов на основе теории категорий обычно определяется категория, объектами которой являются графы, а морфизмами являются гомоморфизмы графов. В правиле преобразования участвуют, по меньшей мере, два графа, которые называются его левой (обычно обозначаемой L) и правой (обычно обозначаемой R) частями. Левая часть описывает, какой подграф должен содержаться в графе G, чтобы данное преобразование могло быть к нему применено, а правая часть описывает, как будет выглядеть этот подграф после данного преобразования. Морфизмы между левыми и правыми частями описывают, какие части графа будут удалены, преобразованы или добавлены. Чтобы применить правило к некоторому подграфу графа G необходимо сначала выявить левую часть правила как подграф графа G. Процесс такого выявления представлен включением $L \xrightarrow{i} G$. Ср. рис.1(a) и 1(b).

Существует два основных подхода к преобразованию графов, основанных на теории категорий: на основе двойного универсального квадрата (сокращенное обозначение - DPo, введенное Х. Эригом и его коллегами [3], [11]), и на основе одинарного универсального квадрата (сокращенное обозначение SPo, введенное Лёве [6], [11]). Главное различие этих подходов в том, что морфизмы в DPo это всюду определенные отображения, а морфизмы в SPo – частичные отображения. Это различие влечет за собой разницу в форме правил.

В подходе DPo правило преобразования задается тремя графами и двумя морфизмами: $L \xleftarrow{l} K \xrightarrow{r} R$. Морфизм l указывает, какие вершины или ребра должны быть удалены (те, которые не входят в образ K), и какие должны быть добавлены (те, которые не входят в образ r). Применение этого правила осуществляется путем построения дополнения до универсального квадрата в левой части диаграммы (с добавлением стрелок $K \xrightarrow{d} D$ и $D \xrightarrow{l^*} G$, а затем через построение универсального квадрата в правой части (стрелки $R \xrightarrow{i^*} H$ и $D \xrightarrow{r^*} H$), см. рис. 1(a).

В подходе SPo правило преобразования определяется одним частичным морфизмом $L \xrightarrow{r} R$. Вершины и ребра, не включенные

в область определения r, будут удалены, те, которые находятся в области определения r, будут преобразованы, а те, которые не находятся в образе r, будут добавлены. Применение правила осуществляется путем построения одинарного универсального квадрата (добавлением стрелок $G \xrightarrow{r^*} H$ и $R \xrightarrow{i^*} H$). См. рис. 1(b).



(a) Подход DPo (b) Подход SPo

Рис. 1: Классические категорные подходы к переписыванию графов

Поскольку в категориях графов не все дополнения до универсального квадрата с необходимостью существуют, то с подходом DPo связаны определенные "условия применения". Как следствие, в подходе DPo запрещены правила, которые создают висячие ребра, тогда как в подходе SPo висячие ребра всегда удаляются при применении правила.

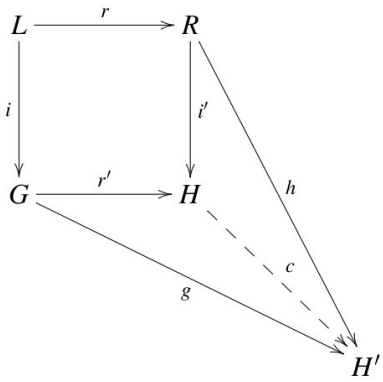
Оба подхода встречаются со многими трудностями на уровне вычисления атрибутов. Наш опыт с подходом DPoPb [9,10,12] и использованием λ-термов для атрибутов оказался перспективным, но построение двойного универсального квадрата повлекло ряд ограничений, связанных с использованием всюду определенных отображений и вынужденным разбиением вычислений на два этапа. Новый подход, представляемый в данной работе, является более естественным, без условий применения и каких-либо ограничений на уровне вычисления атрибутов. Процесс переписывания включает преобразование структуры графа и вычисление его атрибутов. В данной работе мы сосредоточимся на вычислениях атрибутов и лишь слегка коснемся преобразований структуры. Мы надеемся, что сочетание применения подхода SPo и λ-термов позволит нам преодолеть известные трудности вычисления атрибутов.

Для создания системы преобразования графов, основанной на

теории категорий, необходимо определить категорию (объекты и морфизмы), а затем объяснить, как применять правила (в нашем случае через вычисление универсального квадрата).

Универсальный квадрат для пары морфизмов $L \xrightarrow{r} R$, $L \xrightarrow{i} G$ задается парой морфизмов ($G \xrightarrow{r'} H$, $R \xrightarrow{i'} H$) таких, что:

- $i' \circ r = r' \circ i$,
- для любой другой пары морфизмов ($R \xrightarrow{h} H'$, $G \xrightarrow{g} H'$) такой, что $h \circ r = g \circ i$, существует единственный морфизм c такой, что коммутирует следующая диаграмма:



Как следствие, существование универсального квадрата влечет единственность объекта H с точностью до изоморфизма (см. [4,6]). Если в определении универсального квадрата присутствуют эти два свойства, но без единственности c, то такая конструкция называется слабым универсальным квадратом. В этом случае объекты H существуют, но не обязательно определены однозначно с точностью до изоморфизма.

3. Типовое λ-исчисление с индуктивными типами. λ-исчисление в данной работе является простым λ-исчислением с индуктивными типами, сюръективным оператором образования пары и терминальным объектом. Подробности есть в работе [2]; приведем только основные определения, касающиеся этой системы, которая в дальнейшем будет обозначаться как T.

Типы могут быть либо атомарными, либо определенными через конструктор типов. Атомарными типами являются:

- константный тип $\top$;
- конечное или бесконечное множество $\mathcal{S} = \{\alpha, \beta, \ldots\}$, элементами которого являются *переменные для типов*.

Конструкторами типов являются:

- оператор $\to$ для образования функциональных типов, который строит тип функций $A \to B$ для любых типов A и B;
- оператор $\times$ для образования произведения типов, который строит тип пар (произведение) $A \times B$ для любых типов A и B;
- оператор Ind, определяемый следующим образом: пусть $\mathcal{C}$ является бесконечным множеством *операторов интродукции* (конструкторов *элементов* индуктивных типов), с помощью операторов из $\mathcal{C} \cap \mathcal{S} = \emptyset$; тогда индуктивный тип с n конструкторами $c_1, \ldots, c_n \in \mathcal{C}$, каждый из которых имеет местность k_i (где $1 \le i \le n$), имеет вид: $\mathrm{Ind}(\alpha)\{c_1 : A_1^1 \to \ldots \to A_1^{k_1} \to \alpha | \ldots | c_n : A_n^1 \to \ldots \to A_n^{k_n} \to \alpha\}$, Здесь каждое $A \equiv A_i^1 \to \ldots \to A_i^{k_i} \to \alpha$ является схемой индукции; т.е., каждое A_i^j это:
  - либо тип, не содержащий α;
  - либо тип, имеющий вид: $A_i^j \equiv C_1 \to \ldots \to C_m \to \alpha$, где α не встречается ни в каком $C_{\ell \in 1..m}$ (такие A_i^j называются *строго положительными операторами*).

  Здесь $Ind(\alpha)$ *связывает* переменную α.

Примеры. Определение типов $Bool$, Nat и T_ω, типа ω-деревьев:

- $Bool = \mathrm{Ind}(\alpha)\{T : \alpha \,|\, F : \alpha\}$;
- $Nat = \mathrm{Ind}(\alpha)\{0 : \alpha \,|\, \mathrm{S} : \alpha \to \alpha\}$;
- $T_\omega = \mathrm{Ind}(\alpha)\{0_\omega : \alpha \,|\, \mathrm{S}_\omega : \alpha \to \alpha \,|\, \mathrm{Lim} : (Nat \to \alpha) \to \alpha\}$.

Пусть $\mathcal{V}$ является бесконечным множеством переменных $\mathcal{V}$ (причем $\mathcal{V} \cap \mathcal{S} \cap \mathcal{C} = \emptyset$). Множество из λ-*термов* генерируется по следующим правилам грамматики:

$$M \ ::= \ c \,|\, Rec^{B \to D} \,|\, x \,|\, (\lambda x : B \cdot M) \,|\, (M \ M) \,|\, < M \ M >,$$

где $x \in \mathcal{V}$, $c \in \mathcal{C}$, B и D являются произвольными типами, а $Rec^{B \to D}$ является стандартным оператором рекурсии (подробности имеются в работах [7], [2]).

Все термы и типы рассматриваются с точностью до α-конверсии, *т.е.*, с точностью до переименований связанных переменных. Контекстом Γ является множество переменных для термов, имеющих типы: $x_1 : A_1, ..., x_n : A_n$ ($x_1, ..., x_n$ все различны). Γ, Δ обозначает объединение контекстов Γ, Δ (предполагается, что Γ, Δ не имеют общих переменных для термов). Выражение $\Gamma \vdash M : A$ называется суждением о типе (или секвенцией). Его смысл состоит в том, что "терм M имеет тип A в контексте Γ".

Далее приведены аксиомы и правила вывода для суждений (т.е. правила вывода правильно построенных термов с типами в контексте). Ниже A, B, D обозначают произвольные типы, Γ обозначает произвольный контекст.

Аксиомы:

- $\Gamma, x : A \vdash x : A, \quad \Gamma \vdash 0 : \top$;
- Для каждого индуктивного типа $C = \mathrm{Ind}(\alpha)\{c_1 : A_1 \,|\, \ldots \,|\, c_n : A_n\}$ and $1 \le i \le n$
$$\Gamma \vdash c_i : A_i[B/\alpha]$$
(например, если $C = Nat$, то $\Gamma \vdash 0 : Nat$ и $\Gamma \vdash S : Nat \to Nat$);
- Для C такого же, как и выше, и любого типа D аксиома[1]:
$$\Gamma \vdash Rec^{C \to D} : \Upsilon_C(A_1, D) \to ... \to \Upsilon_C(A_n, D) \to C \to D.$$

Правила типизации термов.

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash < M\ N > : A \times B}(pair),$$

$$\frac{\Gamma \vdash M : A \times B}{\Gamma \vdash p_1 M : A}(p_1), \quad \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash p_2 M : B}(p_2),$$

[1] $\Upsilon_C(A, D)$ являются определенными вспомогательными типами, которые используются для определения рекурсии из C в D. Они соответствуют типам функций, которые появляются в стандартных рекурсивных уравнениях над C. Например, если $C = D = Nat$ и $A = Nat \to Nat$ (тип наследника S), то $\Upsilon_{Nat}(A, Nat) = Nat \to Nat \to Nat$.

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash (\lambda x : A \cdot M) : A \to B}(\lambda),$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash (M\ N) : B}(app).$$

(i) Константа $Rec^{C \to D}$ называется *рекурсором* из C в D. Заметим, что в случае ее применения (используя правило *app*) к термам

$$M_1 : \Upsilon_C(A_1, D), ..., M_n : \Upsilon_C(A_n, D)$$

мы определяем функцию

$$Rec^{C \to D} M_1 ... M_n : C \to D.$$

Часто включается следующее производное правило:

$$\frac{\Gamma \vdash M_i : \Upsilon_C(A_i, D)\ \ (1 \leq i \leq n)}{\Gamma \vdash (Rec^{C \to D}\ M_1\ \ldots\ M_n) : C \to D}(elim).$$

(ii) Обычно следующие **структурные правила** включаются в T (они допустимы по отношению к другим правилам):

$$\frac{\Gamma \vdash M : B}{\Gamma, x : A \vdash M : B}(wkn), \quad \frac{\Gamma, x : A, x' : A \vdash M : B}{\Gamma, x : A \vdash [x/x']M : B}(contr),$$

$$\frac{\Gamma \vdash N : A \quad \Gamma, x : A \vdash M : B}{\Gamma \vdash [N/x]M : B}(subst).$$

Здесь $[N/x]$ обозначает подстановку с переименованием связанных переменных для избежания коллизии переменных.

Нормализация и равенство. Термы системы T рассматриваются с точностью до равенства, порождаемого отношением конверсии. Уже упоминалась α-конверсия (переименование связанных переменных). Другими конверсиями являются[2] : (i) β-конверсия $(\lambda x : A.M)N = [N/x]M$; (ii) η-конверсия $\lambda x : A.(Mx) = M$ (где переменная x не должна быть свободной в M); (iii) π-конверсии для пары и терминального объекта $p_1 < M, N >= M, p_2 < M, N >= N, < p_1 M, p_2 M >= M$, а также для всякого $M : \top, M = 0$; (iv) и ι-конверсия для рекурсии. Эта ι-конверсия соответствует одному шагу в рекурсивном вычислении.

[2]Мы опускаем контексты и типы термов.

Она применяется к термам, имеющим вид $(Rec^{C\to D}M_1...M_n)(c_iN)$, т.е. когда рекурсивно определяемая функция применяется к терму, который начинается каким-либо оператором интродукции. Например, для $(Rec^{Nat\to Nat}ag)0 \to_\iota a$ и $(Rec^{Nat\to Nat}ag)(Sn) \to_\iota (Rec^{Nat\to Nat}ag)((gn)(Sn))$ (здесь $a : Nat$ является "начальным значением" и $g : Nat \to Nat \to Nat$ определяет шаг индукции. Точное общее определение можно найти в [2], стр.884.

T конфлюэнтно и сильно нормализуемо по отношению к $\beta\eta\pi\iota$-редукциям (направленным конверсиям). Подробное описание и теоремы нормализации для T есть в [2]. Таким образом, отношение эквивалентности для термов, основанное на конверсии, (часто называемое $\beta\eta\pi\iota$-равенством) является разрешимым.

4. Категория атрибутных графов. Рассматриваемая далее категория графов будет обозначаться через Gr^T.

Объекты. Объектами категории Gr^T являются ориентированные атрибутные графы. Будем предполагать, что их вершины и ребра являются множествами натуральных чисел, обозначаемыми как $V(G)$ и $E(G)$ для графа G; причем $V(G) \cap E(G) = \emptyset$, и что на вершинах и ребрах определен стандартный (лексикографический) порядок, обозначаемый как $<$. Это предположение позволяет избежать неоднозначности в определении морфизмов и в любом случае является обычным, когда рассматриваются их програмные реализации. Атрибутами графов будут служить λ-термы системы T, описанной выше.

Мы предполагаем также, что у любой вершины или ребра имеется только один атрибут. Операция пары позволяет представлять различные данные в виде этого единственного атрибута. Каждый атрибут можно рассматривать как кортеж, содержащий всю информацию, находящуюся в данной вершине или ребре. Кортеж из n элементов $< t_1, ..., t_n >$ рассматривается как сокращенная форма записи терма $< ... < t_1, t_2 >, ..., t_n >$. Если $A_1, ..., A_n$ – типы для $t_1, ..., t_n$ соответственно, то тип всего такого кортежа будет записываться как $A_1 \times ... \times A_n$, а не как $(...(A_1 \times A_2) \times ...A_n)$. Тривиальный атрибут $0 : \top$ ($\top$ – терминальный объект) представляет отсутствие атрибута. Таким образом, имеется взаимно однозначное соответствие между множеством вершин и ребер и множеством атрибутов, что позволяет упростить некоторые доказательства.

Если G является атрибутным графом, $V(G)$ – множеством его вершин, $E(G)$ – множеством его ребер, то $att(v)$, где $v \in V(G) \cup$

$E(G)$ является соответствующим атрибутом v (λ-термом).

Трехуровневые морфизмы. Пусть G, H – два атрибутных графа. Будем считать, что все рассматриваемые далее λ-термы обладают типом в одном и том же контексте Г. Этот контекст может быть зафиксирован для всей категории или, по крайней мере, должен быть достаточен для всех рассматриваемых графов и термов. Термы не обязательно должны быть замкнутыми. Равенство термов понимается в обычном смысле как равенство по отношению к α, β, η и также ι конверсии для рекурсии[3]. Морфизм $f : G \to H$ определяется с помощью следующей трехуровневой конструкции:

1. "Структурная часть"f_{str} является частичным гомоморфизмом графов (без атрибутов) из G в H (см. рис. 2(a)).

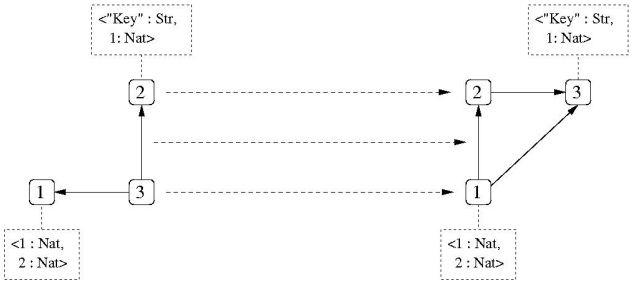
2. "Отношение атрибутной зависимости"f_{adr} является отношением между множествами $V(G) \cup E(G)$ и $V(H) \cup E(H)$, порождаемым вычислительными функциями. Для каждой вершины $v \in V(H) \cup E(H)$ ее прообразом (т.е., множеством всех ее антецедентов) является $[v]_{f_{adr}} \subseteq V(G) \cup E(G)$, которое представляет все те атрибуты графа G, которые можно использовать для вычисления атрибута v. (См. рис. 2(b).)

3. "Вычислительная часть"для каждого элемента $v \in V(H) \cup E(H)$ представляется λ-термом $f_{cmp}(v)$. Эти λ-термы будут называться вычислительными функциями. Их аргументами являются атрибуты графа G , определяемые через f_{adr}. Более строго, пусть $v \in V(H) \cup E(H)$ и $att(v) = t : A$ является соответствующим атрибутом. Пусть $[v]_{f_{adr}} = \{u_1, ..., u_k\}, u_1 < ... < u_k$ (используется упорядочение элементов графа) и $att(u_1) = t_1 : A_1, ..., att(u_k) = t_k : A_k$ являются атрибутами его антецедентов. Тогда терм $s_v = f_{cmp}(v)$ имеет тип $A_1 \to ... \to A_k \to A$ и должен иметь следующее свойство: $s_v(t_1, ..., t_k) = t$ (разумеется, многие аргументы могут быть "пустыми"). В частности, если $[y]_{f_{adr}} = \emptyset$, то $s_v = t$. (см.Рис.2(c).)

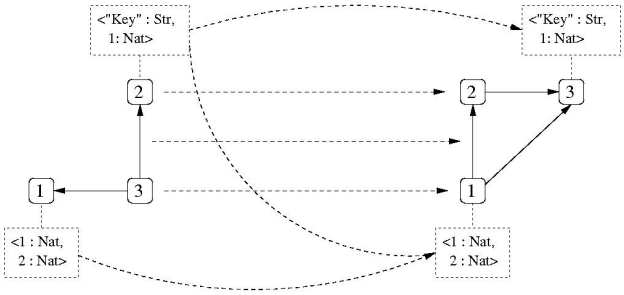
Равенство морфизмов. Два морфизма $f, g : G \to H$ равны, если:

1. $f_{str} = g_{str}$;

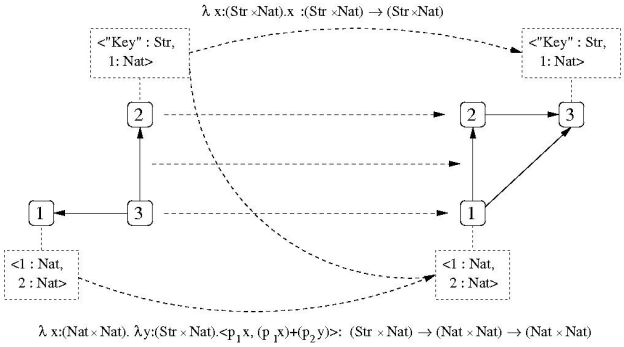
[3]В принципе, могут быть рассмотрены и другие формы равенства.



(a) Структурная часть



(b) Структурная часть + атрибутное отношение зависимости



(c) Структурная часть + атрибутное отношение зависимости + вычислительная часть

Рис. 2: Трехуровневый морфизм атрибутных графов

2. отношения $f_{adr} = g_{adr}$;

3. и для каждого элемента $v \in V(H) \cup E(H)$ его вычислительные функции $f_{cmp}(v)$ и $g_{cmp}(v)$ равны на своих аргументах в G. Более точно, пусть $[v]_{f_{adr}} = [v]_{g_{adr}} = \{u_1, ..., u_k\}$ являются аргументами $f_{cmp}(v)$ и $g_{cmp}(v)$. Тогда требуется, чтобы $f_{cmp}(v)(att(u_1), \ldots, att(u_n)) = g_{cmp}(v)(att(u_1), \ldots, att(u_n))$ как λ-термы.

Замечания:

- Можно отметить, что из равенства морфизмов не следует равенство вычислительных функций, так что два равных морфизма могут иметь различные вычислительные функции.

- Если атрибутами G являются переменные, то равенство функций на их значениях равносильно равенству самих вычислительных функций. (Такой выбор переменных в качестве атрибутов является естественным, если рассматривать схемы правил вместо конкретных правил).

- Если заданы два морфизма $f, g : G \to H$, то из равенства первых двух уровней $f_{str} = g_{str}$ и $f_{adr} = g_{adr}$ следует равенство $f = g$, поскольку значения атрибутов H одни и те же.

- Принимая во внимание способ применения правил, видим, что он вполне согласуется с интуицией. Морфизм $r : L \to R$ используется для формулирования правила (или схемы правил), и естественно принять, что известны соответствующие значения (атрибуты R).

Композиция.

1. На уровне структуры берется композиция $g_{str} \circ f_{str}$;

2. На уровне отношений атрибутной зависимости берется композиция отношений $(g_{adr} \circ f_{adr})$. Можно заметить, что:

$$[w]_{(g \circ f)_{adr}} = [w]_{g_{adr} \circ f_{adr}} = \cup_{v \in [w]_{g_{adr}}} [v]_{f_{adr}}.$$

3. На уровне вычислительных функций (λ-термов) композиция определяется с использованием композиции λ-термов. Более точно, пусть:

- $t = g_{cmp}(w)$;
- $[w]_{g_{adr}} = \{v_1, ..., v_k\}, v_1 < ... < v_k$;
- $t_1 = f_{cmp}(v_1), ..., t_k = f_{cmp}(v_k)$;
- $[v_1]_{f_{adr}} = \{u_{11}, ..., u_{1n_1}\}, ..., [v_k]_{f_{adr}} = \{u_{k1}, ..., u_{kn_k}\}$.

Пересечения антецедентов могут быть непустыми, поэтому пусть $u_1 < ... < u_p$ – различные элементы (вершины или ребра) объединения $[v_1]_{f_{adr}} \cup ... \cup [v_k]_{f_{adr}}$. Пусть $A_1, ..., A_p$, соответственно, – типы атрибутов $att(u_1), ..., att(u_p)$ и $x_1 : A_1, ..., x_p : A_p$ – термальные переменные, не входящие в контекст Γ. Поскольку каждый из элементов u_{ij} соответствует в точности одному из $u_1, ..., u_p$, то $u_{ij} = u_m$ для некоторого $m, 1 \leq m \leq p$ и для всех термальных переменных можно положить $x_{ij} = x_m$. Теперь определяем:

$$(g \circ f)_{cmp}(w) =_{df}$$

$$\lambda x_1 : A_1 ... \lambda x_p : A_p.(t(t_1 x_{11} ... x_{1n_1}) ... (t_k x_{k1} ... x_{kn_k})).$$

Тождественные морфизмы. Для атрибутного графа G его тождественный морфизм Id_G определяется так:

1. для структурной части берется тождественный гомоморфизм графа;
2. для отношения атрибутной зависимости берется отношение тождества;
3. для вычислительных функций для каждого элемента $v \in V(G) \cup E(G)$ пусть A – тип $att(v)$; тогда $(Id_G)_{cmp}(v) = \lambda x : A.x : A \to A$.

Теорема 1. Атрибутные графы и описанные выше морфизмы графов образуют категорию.

Доказательство: Композиция ассоциативна в силу ассоциативности композиции гомоморфизмов графов и ассоциативности композиции отношений. Для λ-термов композиция также ассоциативна, из-за единственности нормальной формы и того факта, что все λ-термы простых типов являются сильно нормализуемыми. Таким образом, любая стратегия их вычисления завершится на таком же λ-терме простого типа.

5. Построение слабого универсального квадрата в категории Gr^T. Как было отмечено, чтобы применить правило $L \xrightarrow{r} R$ к графу G, необходимо найти вложение $R \xrightarrow{i} G$ и затем вычислить универсальный квадрат для r и i. В категории Gr^T роль вложений играют инъективные морфизмы атрибутных графов.

Инъективный морфизм атрибутных графов. Пусть $f : G \to H$ является морфизмом атрибутных графов. Морфизм f инъективен, если (с точностью до равенства морфизмов атрибутных графов):

1. f_{str} – инъективный частичный гомоморфизм графов (т.е. $\forall v_1, v_2 \in V(G) \cup E(G).(f_{str}(v_1) = f_{str}(v_2) \Rightarrow v_1 = v_2)$);
2. $f_{adr} = f_{str}$;
3. для всех $v' \in V(H) \cup E(H)$:
   - если $[v']_{f_{adr}}$ пустое множество, то $f_{cmp}(v') = att(v')$,
   - если множество $[v']_{f_{adr}}$ не пусто (таким образом одноэлементное множество $[v']_{f_{adr}}$ $\{v\}$, поскольку $f_{adr} = f_{str}$, а гомоморфизм f_{str} инъективен), то $f_{cmp}(v') = \lambda x : A.x : A \to A$ где A – тип $att(v)$; заметим, что при этом $att(v) = att(v')$.

Каноническая ретракция[4] для всюду определенного инъективного морфизма атрибутных графов. Пусть $f : G \to H$ – всюду определенный инъективный морфизм атрибутных графов. Ретракцией для f (или его левым обратным) является морфизм атрибутных графов $\overline{f} : H \to G$, такой что $\overline{f} \circ f = Id_G$.

Из этого определения не следует, что $f \circ \overline{f} = Id_H$, и мофизм $\overline{f}$ в общем случае не является единственным. Именно поэтому для ретракции f предлагается каноническая конструкция. Эта конструкция определяется следующим образом:

1. для каждого $v' \in V(H) \cup E(H)$, если множество $[v']_{f_{str}}$ пусто, то v' не имеет образа при $\overline{f}_{str}$; если же $[v']_{f_{str}}$ не пусто (таким образом $[v']_{f_{str}}$ одноэлементное множество $\{v\}$ как выше), то $\overline{f}_{str}(v') = v$;
2. $\overline{f}_{adr} = \overline{f}_{str}$;

[4] Мы пользуемся здесь общепринятой математической терминологией.

3. для каждого $v \in V(G) \cup E(G)$:

   - если множество $[v]_{\overline{f}_{adr}}$ пусто, то $\overline{f}_{cmp}(v) = att(v)$;
   - если множество $[v]_{\overline{f}_{adr}}$ не пусто (и $[v]_{\overline{f}_{adr})}$ одноэлементное множество $\{v'\}$), то $\overline{f}_{cmp}(v) = \lambda x : A.x : A \to A$, где A – тип атрибута $att(v')$.

Поскольку f является всюду определенным морфизмом, то легко видеть, что $\overline{f} \circ f = Id_G$.

Построение слабого универсального квадрата. Наш способ построения (слабого) универсального квадрата в случае применения какого-либо правила восходит работе Лёве и др. [11], но в нашем подходе есть отличия, обусловленные нашим определением атрибутных графов и их морфизмов.

"Отправной точкой"служит пара морфизмов ($L \xrightarrow{r} R$, $L \xrightarrow{i} G$), где i – инъективеный и всюду определенный морфизм атрибутных графов, как это определено выше. Мы хотим вычислить слабый универсальный квадрат ($R \xrightarrow{i'} H$, $G \xrightarrow{r'} H$) для данной пары морфизмов.

Определение универсального квадрата в работе Лёве [11] использует коуравнители. Мы будем иметь в виду эту конструкцию, но дадим в более ограниченном случае прямое определение без рассмотрения коуравнителей в категории Gr^T.

Первым шагом в определении универсального квадрата с использованием коуравнителей в категории графов должно быть взятие копроизведения $G + R$ графов G и R (используемое здесь копроизведение – это просто несвязное объединение). Следующим шагом должна быть его факторизация по некоторому отношению эквивалентности (создание графа $(G+R)'$, содержащего классы эквивалентности в качестве элементов), и затем завершение построения морфизма применением композиции с некоторым морфизмом p из факторного объекта в объект H, представляющий вершину универсального квадрата.

Определим каждый из морфизмов r' и i' как композицию трех морфизмов (см. рис.3) чтобы получить

$$r' = G \xrightarrow{j'} (G+R) \xrightarrow{f'} (G+R)' \xrightarrow{p} H, i' = R \xrightarrow{j''} (G+R) \xrightarrow{f''} (G+R)' \xrightarrow{p} H.$$

Объекты и морфизмы в этих диаграммах определяются следующим образом в несколько шагов:

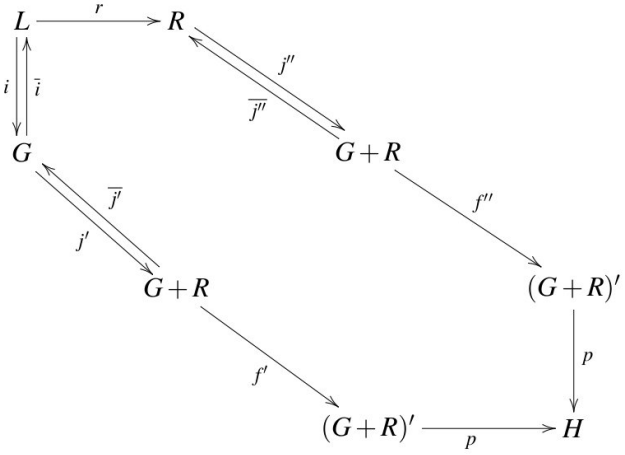


Рис. 3: Построение слабого универсального квадрата

- на уровне структуры $G+R$ является несвязным объединением графов G и R;
- на уровне атрибутов каждый элемент графов G и R в $G+R$ имеет тот же атрибут, что и в G и R;
- j' и j'' являются включениями, соответственно, G и R в $G+R$; таким образом они являются всюду определенными инъективными морфизмами атрибутных графов.

Для продолжения процесса сперва определим отношение эквивалентности $\sim_1$ на элементах графовой структуры $G+R$.

- Положим $a \sim_1 b$ для $a, b \in G+R$ если $\exists x \in L.(j'(i(x)) = a \wedge j''(r(x)) = b)$.
- Теперь отношение $\sim$ определяется как рефлексивное, симметричное и транзитивное замыкание $\sim_1$.
- Заметим, что элементы $G+R$, которые не являются образами элементов $G - i(dom(r))$, образуют классы эквивалентности, состоящие из единственного элемента (самого себя).

Элементы $(G+R)'$ определяются как классы эквивалентности элементов $G+R$. Легко проверить, что данное определение согласуется с отношением инцидентности и что отображение, отображающее каждый элемент $G+R$ в его класс эквивалентности является (всюду определенным) гомоморфизмом графов. Это отображение будет структурной частью морфизмов f' и f''.

Более того, каждый класс эквивалентности по отношению к $\sim$, содержащий образ какого-либо элемента R, может рассматриваться как "конус", состоящий из образа этого элемента R при отображении j'' и образов его антецедентов относительно r после применения к ним $j' \circ i$. В частности, каждый класс эквивалентности содержит в точности один образ какого-либо элемента R. Как следствие, композиция $f''_{str} \circ j''_{str}$ является инъективной.

Данные рассуждения позволяют определить атрибутную часть $(G+R)'$. Каждый класс эквивалентности, который содержит образ какого-либо элемента R, имеет тот же атрибут, какой данный элемент имеет в R. Другие классы эквивалентности (которые имеют вид $\{j'(y)\}, y \in G, y \neq i(x)$ для некоторого $x \in L$) сохраняют тот же атрибут, что и в G.

Определения для отношений зависимости атрибутов и вычислительных функций f' и f'' уже иные. Для f'' отношение f''_{adr} связывает элементы R с соответствующими классами эквивалентности (оно взаимно однозначно на R). Здесь нет никаких связей, относящихся к G. Вычислительные функции – это тождества.

Замечание. Композиция $f'' \circ j''$ инъективна, в частности, $(f'' \circ j'')_{str}$ является инъективным всюду определенным гомоморфизмом графов, $(f'' \circ j'')_{adr} = (f'' \circ j'')_{str}$ и вычислительные функции являются тождествами.

Теперь можно определить f' следующим образом:

- $\forall v \in img(j' \circ i)$:

$$f' = G + R \xrightarrow{\overline{j'}} G \xrightarrow{\bar{i}} L \xrightarrow{r} R \xrightarrow{j''} G + R \xrightarrow{f''} (G+R)';$$

- для элементов $G - i(L)$, f' определяется как для тождественных морфизмов.

Как обычно (см. [11]), H определяется как для конструкции коуравнителя. Пусть $L_0 = dom(r)$. В нашем случае H будет подграфом графа $(G+R)'$. Отношение инцидентности в $(G+R)'$ насле-

дуется из R и G. Элементами на H (на уровне структуры графа) являются:

1. все классы эквивалентности вида $\{x_1, ..., x_k, z\}$ $(x_1, ..., x_k \in j'(i(L_0)), z \in j''(r(L))$;
2. все классы эквивалентности вида $\{z\}$, $z \in j''(R - r(L))$;
3. все классы эквивалентности вида $\{x\}$, $x \in j'(G - i(L))$, не являющиеся висячими дугами [11].

Атрибуты классов эквивалентности первых двух типов наследуются из R, а для третьего типа – из G.

Морфизм p определяется следующим образом. Его структурная часть – это тождество на всех тех элементах $(G + R)'$, которые остаются в H. Также имеет место равенство $p_{adr} = p_{str}$, и все вычислительные функции являются тождественными.

Теперь i', r' и H определяются так, чтобы $i' \circ r = r' \circ i$. Пусть $h : R \to H'$ и $g : G \to H'$ – два других морфизма, таких что $h \circ r = g \circ i$. Поскольку отображение i' инъективно, то можно применить каноническую ретракцию $\overline{i'}$ и принять за c (см. рис. 4) $h \circ \overline{i'}$, а для элементов, не находящихся в области определения $\overline{i'}$, продолжить c, чтобы привести его в согласие с g. Коммутативность на уровне вычислительных функций следует из определения равенства морфизмов для атрибутных графов. Таким образом, данная диаграмма коммутативна, но в общем случае единственность c не гарантируется, поэтому здесь имеет место слабый универсальный квадрат.

В дальнейшем предложенная выше конструкция слабого универсального квадрата может рассматриваться как каноническая. Необходимо отметить, что она гарантирует инъективность второй вертикальной стрелки i'. Полученные результаты целесообразно сформулировать в виде теоремы:

Теорема 2. В категории Gr^T для любой пары морфизмов $L \xrightarrow{r} R, L \xrightarrow{i}$, где i является инъективным морфизмом, существует слабый универсальный квадрат, который задается парой $\xrightarrow{r'} R, L \xrightarrow{i'}$, причем в ней i' также является инъективным.

Каноническая конструкция слабого универсального квадрата позволяет корректно определить композицию правил, так, чтобы она соответствовала композиции морфизмов, задающих правила.

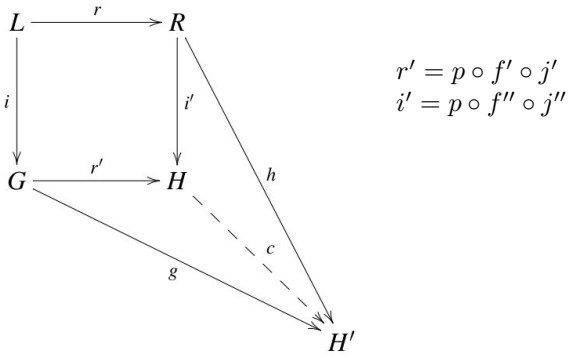


Рис. 4: Определение универсального квадрата

6. Примеры. Для иллюстрации описываемого подхода к преобразованию графов в данном разделе представлены два примера: первый сравнивает графовую грамматику для вычисления $n!$ в нашем подходе со стандартным определением, основанным на использовании Σ-алгебр. Упомянем также несколько примеров, которые не освещаются в данной работе, но которые легко развить, используя этот подход:

- клонирование графов [1];
- информационный обмен между атрибутами и структурой;
- вычисления с использованием атрибутов, являющихся функциями.

6.1 Вычисление $n!$. Данный пример показывает преимущества нашего подхода на вычислительном уровне. Рис.5 представляет пример вычисления факториала числа n с использованием двух различных систем переписывания графов. Первая основана на Σ-алгебрах и копирует пример, приведенный на сайте AGG, а другая основана на излагаемом здесь подходе с применением λ-термов.

Если применить классический подход на базе Σ-алгебр, то потребуются три правила в двух слоях (слои определяют приоритет применения правил [4] и обозначены римскими цифрами на рисунках) (см. рис.5(a)):

1. первое правило применяется $n-2$ раза и создает цепочку со всеми значениями между n и 2 (циклическое ребро используется для задания того, какой именно атрибут должен уменьшаться);

2. второе правило используется для завершения первого правила (оно состоит в удалении этого циклического ребра);

3. третье правило используется $n-2$ раза для перемножения всех чисел между n и 2.

Таким образом, для вычисления $n!$ должно быть применено $2n-3$ правил. Такое количество применений правил получается в силу того факта, что в системах переписывания графов каждое правило может смоделировать применение определенных операций Σ-алгебры, но поскольку в Σ-алгебрах трудно выразить рекурсию, то вычисление факториала требует промежуточных преобразований графов.

Если же использовать описываемый здесь подход, основанный на λ-термах, то нужно только одно правило для вычисления факториала числа n; поскольку существует оператор рекурсии, то легко записать λ-терм, вычисляющий $n!$ (см. рис. 5(b)). Разумеется, вычисление рекурсивных функций потребует многих шагов, но это входит в стандартную схему, основанную на индуктивных типах и оптимизированную для таких вычислений.

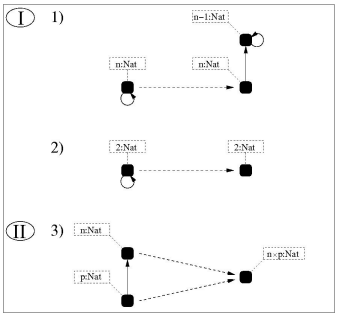
Такое сокращение числа правил, необходимых для вычисления атрибутов, имеет два преимущества:

- простота грамматики графа;

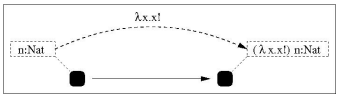
- применение правила затратно в терминах алгебраической сложности. Чтобы применить правило, нужно решить задачу об изоморфизме подграфов (которая NP-полна) для нахождения соответствия. Если же упростить левую часть правил, то при сокращении числа применений правил значительно сокращается время вычисления $n!$.

Если основываться на этом примере, то представляется, что вычисление атрибутов в нашем подходе более выразительно и, разумеется, более экономично, чем вычисления, основанные на Σ-

алгебрах, особенно при рассмотрении задач, где нужна рекурсия.[5]



(a) Вычисление $n!$ на основе Σ-алгебр



(b) Вычисление $n!$ на основе λ-термов

Рис. 5: Грамматики графов для вычисления $n!$ в двух различных системах переписывания атрибутных графов

6.2 Как работать с бесконечностью в функциональных атрибутах. Другим преимуществом применения λ-термов в качестве атрибутов является возможность иметь сложные структуры данных, которые могут представлять бесконечность. Смоделировать бесконечные атрибуты можно через определение функции с бесконечным типом данных в качестве ее области определения. Например, тип T_ω ω-деревьев (см. рис. 6(a)) представляет деревья, в вершинах которых допускается бесконечное ветвление. Применяя операторы рекурсии к индуктивным типам, можно определить преобразования на таких бесконечных древовидных структурах.

Можно привести форму рекурсивных уравнений для отображений из Nat в любой тип A и из T_ω в B:

- $\psi(0) = a; \psi(S(x)) = g(x, \psi(x))$ (для Nat);
- and $\phi(0_\omega) = t; \phi(S_\omega(x) = g(x, \phi(x)); \phi(Lim(f)) = h(f, \phi \circ f)$ ($\circ$ означает композицию).

В синтаксисе λ-исчисления решение этих уравнений выглядит так:

- $Rec^{Nat \to A}(a)(g)$,

[5] Для упрощения рисунка "$x!$" написано вместо λ-терма, вычисляющего $x!$

- and $Rec^{T_\omega \to B}(t)(g)(h)$.

Теперь преобразование рассматриваемых деревьев может быть записано следующим образом:

- Пусть d определено через $d(0) = 0, d(S(x)) = S(S(d(x)))$ и $d = Rec^{Nat \to Nat}(0)(\lambda x.\lambda y.S(S(y)))$, т.е. $d(x) = 2x$ в арифметической нотации.

- Пусть ϕ определено через
  $\phi(0_\omega) = 0_\omega, \phi(S_\omega(x)) = S_\omega(\phi(x)), \phi(Lim(f)) = Lim(f \circ d)$, выражая это с помощью Rec:

  $$\phi(f) = Rec^{T_\omega \to T_\omega}(0_\omega)(\lambda x^{T_\omega}.S_\omega)(\lambda u.\lambda v.(u \circ d)).$$

  Это преобразование выбирает (один раз) ветви с парными номерами на первом (бесконечном) разветвлении.

Чуть более сложное преобразование выбирает ветви с парными номерами на каждом бесконечном разветвлении. Есть только одно изменение. Определим ϕ' через $\phi'(0_\omega) = 0_\omega$, $\phi'(S_\omega(x)) = S_\omega(\phi'(x))$, $\phi'(Lim(f)) = Lim((\phi' \circ f) \circ d)$,выражая это с помощью Rec:

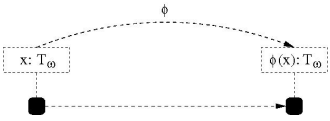
$$\phi'(f) = Rec^{T_\omega \to T_\omega}(0_\omega)(\lambda x^{T_\omega}.S_\omega)(\lambda u.\lambda v.(v \circ d)).$$

Рис. 6(b) представляет правило, которое выбирает ветви с парными номерами на первом бесконечном ветвлении (используя определенное выше ϕ). Рис.6(c) представляет пример ω-дерева определенного через терм $Lim(Rec^{Nat \to T_\omega}(0_\omega)(\lambda x^{Nat}\lambda y^{T_\omega}.S_\omega(y)))$, а рис. 6(d) представляет результат применения этого правила к этому дереву. Это очень простой пример, но возможны и более сложные, в которых выбор ветвей осуществляется рекурсивно в каждом ветвлении и (или) листьями являются элементы какого-нибудь сложного типа.

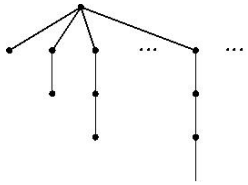
7. Заключение. Цель данной работы – представить новую систему переписывания атрибутных графов, основанную на подходе SPo, главной особенностью которой является применение λ-исчисления для выражения атрибутных вычислений. На структурных частях этот подход демонстрирует те же характеристики, что и классический подход DPo, но на вычислении атрибутов, как показано на примерах, можно упростить грамматики, расширить выразительность правил и получить определенный выигрыш в экономичности вычислений.

$$T_\omega = Ind\alpha\{0_\omega : \alpha,$$
$$S_\omega : \alpha \to \alpha,$$
$$Lim : (Nat \to \alpha) \to \alpha\}$$

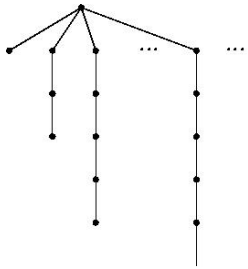
(a) Определение индуктивного типа ω-деревья



(b) Правило преобразования



(c) Пример ω-дерева (терм см.в тексте). Длина n-ой ветви равна n



(d) Результат применения правила 6(b) к графу с ω-деревом на рис.6(c) в качестве атрибута: выбраны только ветви с парными номерами

Рис. 6: Вычисление на атрибутах, представляющих бесконечные деревья

Благодаря выразительной силе индуктивных типов, становится возможным провести перенос некоторых механизмов переписывания из структурной части в вычислительную и наоборот (атрибуты могут представлять определенные типы графов, например, деревья).

Теоретически, при работе с атрибутами подход SPo делаете необходимым определение и построение слабого универсального квадрата. Решение приведено в данной работе. Следующим шагом

исследования будет изучение обычных свойств систем переписывания, таких как слияние, завершение, анализ критических пар и т.д. Отметим, что в применении к вычислению атрибутов эти свойства оказываются хорошо известными свойствами λ-исчисления. Дополнительно, мы намереваемся исследовать другой способ описания преобразований атрибутов, основанный на исчислении, использующим правила вывода.

Возможные области приложений данного подхода включают все обычные приложения преобразований графов, т.е., верификацию и преобразование моделей в программировании, но более "слабая"связь между вычислительной и структурной частями позволит также достичь значительно более конкретных целей.

Литература

1. Boisvert, B., Féraud, L., and Soloviev, S. (2011b). Typed lambda-terms in categorical graph rewriting. In *The International Conference Polynomial Computer Algebra, April 18-22, Saint-Petersburg, Russia, Euler International Mathematical Institute.*

2. Chemouil, D. (2005). Isomorphisms of simple inductive types through extensional rewriting. *Math. Structures in Computer Science*, 15(5), pages 875–917.

3. Ehrig, H. (1978). Introduction to the algebraic theory of graph grammars (a survey). In *Graph-Grammars and Their Application to Computer Science and Biology*, pages 1–69.

4. Ehrig, H., Ehrig, K., Prange, U., and Taentzer, G. (2006a). *Fundamentals of Algebraic Graph Transformation (Monographs in Theoretical Computer Science. An EATCS Series).* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

5. Ehrig, H., Padberg, J., Prange, U., and Habel, A. (2006b). Adhesive high-level replacement systems: A new categorical framework for graph transformation. *Fundam. Inf.*, 74(1):1–29.

6. Löwe, M., editor (1993). *Algebraic approach to single pushout graph transformation, TCS*, volume 109.

7. Luo, Z. (1994). *Computation and Reasoning: A Type Theory for Computer Science.* International Series of Monographs on Computer Science. Oxford University Press, USA.

8. Orejas, F. (2011). Symbolic graphs for attributed graph constraints. *J. Symb. Comput.*, 46:294–315.

9. Rebout, M., Féraud, L., Marie-Magdeleine, L., and Soloviev, S. (2011). Computations in Graph Rewriting: Inductive types and Pullbacks in DPO Approach. In Szmuc, T., Szpyrka, M., and Zendulka, J., editors, *Advances in Software Engineering Techniques, CEE-SET 2009, Krakow, Poland, October 2009*, volume 7054 of *LNCS*, pages 150–163. Springer-Verlag.

10. Rebout, M., Féraud, L., and Soloviev, S. (2008). A Unified Categorical Approach for Attributed Graph Rewriting. In Hirsch, E. and Razborov, A., editors, *International Computer Science Symposium in Russia (CSR 2008), Moscou 07/06/2008-12/06/2008*, volume 5010 of *LNCS*, pages 398–410. Springer-Verlag.

11. Rozenberg, G., editor (1997). *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations.* World Scientific.

12. Tran, H. N., Percebois, C., Abou Dib, A., Féraud, L., and Soloviev, S. (2010). Attribute Computations in the DPoPb Graph Transformation Engine (regular paper). In *GRABATS 2010, University of Twente, Enschede, The Netherlands, 28/09/2010-28/09/2010*, page (electronic medium), http://www.utwente.nl/en. University of Twente.

Баранов Сергей Николаевич — д.ф.-м.н., проф.; гл.н.с. лаборатории технологий и систем программирования СПИИРАН. Область научных интересов: технология программирования, промышленная разработка программных изделий, формальные методы верификации требований, формальные языки. Число научных публикаций — 90+. SNBaranov@gmail.com; СПИИРАН, 14-я линия В.О., д. 39, г. Санкт-Петербург, 199178, РФ; р.т./факс +7(812)328-0887.

Sergey N. Baranov — Doc.Nat.Sci, Professor; Chief Research Associate, lab of software engineering and systems, SPIIRAS. Research area: software engineering, industrial development of software products, formal methods for requirements verification, formal languages. Number of publications — 90+.

SNBaranov@gmail.com; SPIIRAS, 14-th line V.O., 39, St. Petersburg, 199178, Russia; office phone/fax +7(812)328-0887.

Буавер Бертран — аспирант университета Тулузы, IRIT. Область научных интересов: преобразования графов, методы теории категорий. Число научных публикаций — 4. Bertrand.Boisvert@irit.fr; IRIT, route de Narbonne, 118, Toulouse, 31400 France; р.т. +33 561556255, факс +33 561556258. Научные руководители — С.В.Соловьев, Л.Феро.

Boisvert Bertrand — PhD student at the University of Toulouse, IRIT. Research area: graph transformations, category theory methods. Number of publications — 4. Bertrand.Boisvert@irit.fr; IRIT, route de Narbonne, 118, Toulouse, 31400 France;

phone +33 561556255, fax +33 561556258. Scientific advisors — S.V. Soloviev, L.Féraud.

Соловьев Сергей Владимирович — к.ф.-м.н., профессор IRIT и университета г. Тулузы-3, Франция; советник директора СПИИРАН. Область научных интересов: теория доказательств и ее применения в математике и информатике. Число научных публикаций – 70+. Sergei.Soloviev@irit.fr; IRIT, route de Narbonne, 118, Toulouse, 31400 France; р.т. +33 561556255, факс +33 561556258.

Sergey V. Soloviev — PhD, Professor at IRIT and the University of Toulouse-3, advisor to Director of SPIIRAS. Research area: proof theory and its applications in mathematics and computer science. Number of publications — 70+.

Sergei.Soloviev@irit.fr; IRIT, route de Narbonne, 118, Toulouse, 31400 France; phone +33 561556255, fax +33 561556258.

Луи Феро — д.ф.-м.н., профессор IRIT и университета г. Тулузы-3, Франция. Область научных интересов: теоретическая информатика, переписывание графов, формальные языки, синтаксический анализ, компиляция. Число научных публикаций – 45. feraud@irit.fr; IRIT, route de Narbonne, 118, Toulouse, 31400 France; р.т. +33 5 61556946, факс +33 561556258.

Louis Féraud — Doc.Sci, Professor at IRIT and the University of Toulouse-3. Research area: theoretical computer science, graph rewriting, modeling, formal languages, syntax analysis, compilation. Number of publications — 45.

feraud@irit.fr; IRIT, route de Narbonne, 118, Toulouse, 31400 France; phone +33 5 61556946, fax +33 561556258.

Поддержка исследований. Работа выполнена при частичной финансовой поддержке проекта Climt (грант ANR-11-BS-02-016-02) и сети франко-российских научных обменов (GDRI CNRS).

Рекомендовано ЛПИ СПИИРАН, зав. лаб. Юсупов Р.М., член-корр. РАН.

Статья поступила в редакцию 23.10.2012.

РЕФЕРАТ

Баранов С.Н., Буавер Б., Соловьев С.В., Феро Л. **Некоторые приложения λ-исчисления с типами к атрибутным вычислениям в системах категорных преобразований графов.**

Статья посвящена проблемам в системах переписывания графов с атрибутами в виде λ-термов. Предлагаемый в статье новый алгоритм для преобразования таких графов основан на построении одинарного универсального квадрата с применением рекурсии для вычисления соответствующих λ-термов.

Целью работы является описать формальный каркас, который способен выражать оба вида преобразований, необходимых для переписывания графов: переписывание структурной части графа и сопровождающие его вычисления на его атрибутах. Данный каркас может быть полезен для представления и исследования различных моделей сложных систем, разрабатываемых с применением современных графических языков, таких как SDL и UML.

Классические системы переписывания графов, основанные на теории категорий, при применении к атрибутным графам сталкиваются с проблемой сложности при вычислении атрибутов. Известным частичным решением является подход на основе двойного универсального квадрата (DPo). Однако необходимость его построения накладывает определенные ограничения на вычислительные функции, в основном из-за использования всюду определенных отображений и необходимости разбивать процесс вычисления атрибутов на два этапа. В данной работе исследуется альтернативных подход, основанный на одинарном универсальном квадрате (SPo) в сочетании с рекурсивным вычислением λ-термов, представляющих атрибуты рассматриваемого графа.

В работе дается обзор существующих подходов к переписыванию атрибутных графов на основе теории категорий и обсуждаются связанные с этим новые проблемы. Формальным образом определяется категория атрибутных графов с λ-термами в качестве атрибутов и вводятся соответствующие аксиомы и правила преобразования типов при переписывании графов. Центральным пунктом работы является доказательство существования слабого универсального квадрата и алгоритм его вычисления для заданного атрибутного графа. Применение данного подхода иллюстрируется двумя примерами: вычислением $n!$ и решением проблем с бесконечностью в атрибутах графа через использование ω-деревьев. В обоих примерах вычисление атрибутов сравнивается с аналогичными вычислениями в классическом подходе DPo; показано, что предлагаемый подход дает существенное сокращение числа шагов.

SUMMARY

Baranov S., Boisvert B., Soloviev S., Féraud L. **Applications of typed λ-terms to categorical attributed graph transformations.**[6]

This paper relates to problems in graph rewriting systems for graphs attributed with typed λ-terms. The proposed new algorithm for graph transformations is based on single pushout with recursion for computation of the corresponding λ-terms.

The paper's aim is to describe a formal framework that can express both types of transformation necessary for graph rewriting: a rewrite of the structural part of a graph and accompanying computations on its attributes. This framework may be useful for representing and studying various models of complex systems developed in modern graphical languages, such as SDL and UML.

A classical graph rewriting systems based on category theory has a known challenge of attribute computations when applied to attributed graphs. A known partial solution is based on the double pushout approach (DPo).

However, the construction of the double pushout imposed restricting constraints on computation functions mostly due to the usage of total maps and the obligation to split attribute computations into two parts. Here we investigate an alternative approach based on single pushout (SPo) combined with computation of typed λ-terms which represent graph attributes. We demonstrate that this facilitates the overall computation and substantially reduces the number of steps in the respective algorithms.

The paper provides an overview of existing approaches to graph rewriting based on category theory and discusses the related challenges. A category of attributed graphs with typed λ-terms as attributes is formally defined and respective axioms and typing rules for graph rewriting are introduced. The central point of the paper is a proof of existence and the algorithm for computation of a weak pushout for a given attributed graph. Application of this approach is illustrated with two examples: computation of $n!$ and managing the infinity in graph attributes using ω-trees. In both examples computation of attributes is benchmarked against classical DPo approach and the substantial reduction of the number of steps is demonstrated.

[6]The research has been partially supported by the Climt project, ANR-11-BS-02-016-02 and the Network of French-Russian Scientific Exchanges (GDRI CNRS).