КОМПЬЮТЕРНЫЕ НАУКИ 2009. Вып. 2

УДК 004.07(063)

© *В. Э. Вольфенгаген, Л. Ю. Исмаилова, С. В. Косиков, А. Д. Лаптев, В. Н. Назаров, В. В. Рословцев, И. С. Сафаров, А. Л. Степанов*

КОМБИНАТОРЫ: ОБЪЕКТЫ, ПОМОГАЮЩИЕ ПОНЯТЬ СТРОЕНИЕ КОМПЬЮТИНГА. АТОМАРНО-МОЛЕКУЛЯРНЫЙ МАТЕРИАЛ СРЕДЫ КОМПЬЮТИНГА

В повседневном компьютинге над сущностями выполняются операции, на внутреннюю структуру которых обращается мало внимания. Вместе с тем многие обычные операции состоят из более примитивных конструкций, соединенных посредством способа комбинирования. Взаимодействие конструкций осуществляется в среде «аппликативного взаимодействия», а изучение свойств этой среды позволяет понять природу вычислений.

В настоящей работе основное внимание уделено выяснению технологических особенностей вычислений с объектами. Их взаимодействие рассматривается в аппликативной среде, что позволяет выяснять внутреннюю структуру обычных операций, знание которой позволяет понять их свойства. Обсуждается выбор исходных константных сущностей, считающихся исходными и называемыми комбинаторами. Эти исходные сущности используются как основные «строительные блоки», вступающие в аппликативной среде во взаимодействие друг с другом. В результате взаимодействия возникают конструкции, дающие представительные наборы обычных операторов и погруженные вычислительные системы.

Ключевые слова: комбинаторная логика, компьютинг, аппликативная среда, погруженные вычислительные системы.

Введение

Обилие предоставляемых информационных сервисов заставляет по-новому взглянуть на то, какие же функции они в действительности выполняют. Представление о внутренней структуре сервисов обычно отсутствует, а их вычислительное поведение остается до конца не ясным. Попытки сузить круг явлений и специализировать применяемые средства до специальных разделов программирования переводят обсуждение в плоскость реализации, нагромождая второстепенные детали и уводя в сторону от важных механизмов, которые лежат в основе всякого вычисления [1]. Похоже, что трудности создания моделей вычислений [2–6] во многом остались не преодоленными, а взамен возникли иные направления исследований [7], на время отставившие старые вопросы и предлагающие заняться пока еще не вызвавшими разочарования моделями, границы применения которых на время остаются не выясненными.

Вопросы же компьютинга отдаются на волю компаний-разработчиков, которые ориентируются на потребности рынка сбыта, а не на поиск действительно перспективных концепций [8]. Связи же технологического и естественного компьютинга только лишь намечаются, а о сложившейся понятийной основе пока говорить не приходится, хотя имеются сильные аргументы, что на их роль претендует аппликативный компьютинг и комбинаторная логика [9,10].

В лучшем случае превалирует вполне прагматическая — и технологическая, — точка зрения, принимающая в качестве основного вызова развитию компьютинга сеть Web, которая рассматривается как однородный и в высокой степени связный набор вычислительных ресурсов, что может рассматриваться как некоторый метакомпьютер [11]. На пути характеризации ассоциированной этому скорее гипотетическому, чем даже виртуальному вычислителю среды вычислений возникает ряд принципиальных трудностей, которые преодолеть не удалось [12]. Нет ясности, какого рода языками программирования нужно оснастить этот метакомпьютер, поскольку не удается получить единую модель вычислений, включающую наблюдаемые эффекты Web, а предложения ограничиваются весьма специальными случаями [13]. Исследователи не

определились в выборе между фон-Неймановской и не фон-Неймановской архитектурой, сильной структуризацией баз данных и слабыми структуризациями гипертекстовых представлений информации, семантическими сетями и онтологиями, семантикой вычислений с состояниями или без них, установлением или стиранием границ между программами и данными. Если на все это дополнительно наложить постоянные сдвиги границ между отдельными дисциплинами компьютинга, то принципиальные продвижения и прогресс могут показаться скорее исключением, чем правилом, что в области информационных технологий создает благоприятную основу для лавинообразного нарастания потока инноваций [14]. Этими альтернативами дело далеко не исчерпывается, но они помогают входящему в проблематику сориентироваться в состоянии исследований (см. [15–17]).

В настоящей работе предпринимается попытка выяснения технологических основ «чистого» компьютинга, в особенности тех из них, которые касаются вычислений с объектами. Их взаимодействие рассматривается в аппликативной среде, что позволяет выяснять внутреннюю структуру обычных операций, знание которой позволяет понять их свойства. Обсуждается выбор исходных константных сущностей, считающихся исходными и называемыми *комбинаторами*. Эти исходные сущности используются как основные «строительные блоки», вступающие в аппликативной среде во взаимодействие друг с другом. В результате взаимодействия возникают конструкции, дающие представительные наборы обычных операторов и погруженные вычислительные системы.

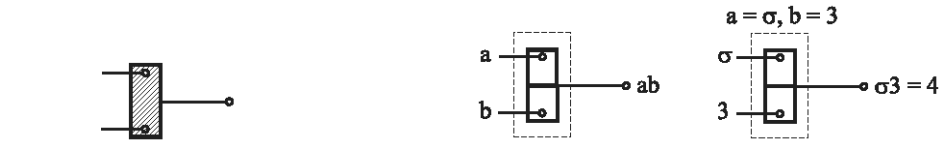
Изложение построено в виде четырех основных разделов. В разделе 1 обсуждается представление об аппликативной среде, обеспечивающей взаимодействие объектов друг с другом. Вопрос, как из исходных объектов, пользуясь S- и K-экспансией в аппликативной среде, синтезировать объект с заданной комбинаторной характеристикой, вынесен в раздел 2. Для конкретности для операции композиции строится ее представление посредством виртуального объекта, что подчеркивает неатомарность «обычных» операций. Изучение их внутренней структуры проявляет внутреннее единство интуитивно применяемых в информационных и вычислительных технологиях операций, несмотря на многообразие применяемых форм этих операций. В разделе 3 устанавливается, что аппликативная среда позволяет поддерживать виртуальные объекты, дающие представление условных конструкций. Для примера в деталях рассмотрена известная конструкция *if e then a else b*. В разделе 4 рассмотрены бесконечные диаграммы представления объектов, в основе которых лежит идея использования неподвижной точки отображений. Бесконечные структуры объектов имеют первостепенное значение для рассмотрения периодичности или цикличности вычислений с объектами, в особенности эффектов рекурсивной модификации аппликативной среды.

§ 1. Аппликативная среда: взаимодействие объектов

Все мы привыкли использовать в повседневных вычислениях операции и операторы. С интуитивной точки зрения они применяются как *элементарные*, но изредка все же возникает вопрос, а откуда они взялись. И здесь не всегда спасает точка зрения, что операции — это наипростейшие конструкции. Нельзя ли средствами компьютинга посмотреть, какова их внутренняя структура?

Попробуем дать ответ на этот вопрос, пользуясь объяснительной системой такого компьютинга, единственной примитивной операцией которого является применение — аппликирование, — одного объекта к другому. Имеются две выделенные *точки входа*, через которые поступают участвующие во взаимодействии объекты, а также *точка выхода*, с которой снимается результат. Основной возникающий вопрос заключается в том, что происходит внутри «черного ящика», показанного на рис. 1a.

Объекты вступают во *взаимодействие* друг с другом, а происходящее в ходе осуществления их аппликации с интуитивной точки зрения показано на рис. 1b. Единственное, что можно усмотреть из этого рисунка — это сам факт применения, или, по иной терминологии, *аппликирования* объекта a к объекту b. В тех приложениях, которые к настоящему времени достаточно осмыслены, считается, что a играет роль *функции*, а b — *аргумента*, а взаимодействие



(a) Представление аппликации (b) Выполнение аппликации

Рис. 1. Аппликация: представление и выполнение

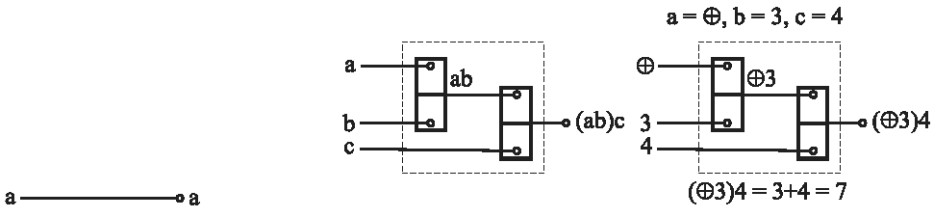
объектов представляется вычислением. Рассматривая такое аппликирование как *операцию*, а еще лучше — как *метаоперацию*, — можно ставить вопрос о *результате* этой операции. Следуя всем известным традициям оперирования функциями, примем, что $a(b)$ считается значением применения объекта a к объекту b. Из соображений симметрии объектов a и b записями результата их взаимодействия $a(b)$, (ab) и ab на практике пользуются взаимозаменяемо.

В примере выполнения аппликации на рис. 1b: a — операция σ прибавления единицы, или 'следования за' $\sigma\ \ z = z + 1$, а b — операнд, равный 3. Тогда в итоге осуществления аппликирования получаем результат $\sigma\ 3 = 4$.

Имеются два *исходных* объекта, и это *входные* объекты, но 'самый верхний' объект влияет на расположенный ниже объект, а на *выходе* формируется *значение* — результат применения первого объекта ко второму. Это своего рода «черный ящик», только с тем дополнением, что он состоит ровно их двух секций, одна из них играет роль объекта-функции, а вторая — объекта-аргумента.

С помощью такого прямоугольника можно представлять *примитивную операцию взаимодействия* двух объектов, которую не подвергаем дальнейшему анализу. В противном случае эту *представляющую диаграмму* можно компоновать из других диаграмм.

Тривиальным примером компоновки может быть комбинирование *никаких* объектов вообще. Результатом будет линия, заканчивающаяся кружком на рис. 2a.



(a) Тождество (b) Каскадирование аппликаций

Рис. 2. Комбинирование объектов

По таком каналу объект передается со входа на выход безо всякого преобразования, поэтому такая диаграмма называется тождественной функцией.

Нетривиальная компоновка показана на рис. 2b. Эта комбинация может рассматриваться как произведение аппликаций, когда выход первого двухсекционного прямоугольника непосредственно направляется на вход второго двухсекционного прямоугольника, давая очевидный результат. В примере: a — операция $\oplus$ сложения в префиксной форме записи $\oplus x\, y = x + y$, b — число 3 и c — число, равное 4. Для этих значений получаем $(\oplus 3)4 = 3 + 4 = 7$.

Только с аппликативными формами много полезных конструкций не сформировать. Одни объекты взаимодействуют с другими, они должны подвергаться воздействию со стороны своих соседей по аппликативным формам, а также сами воздействовать на другие объекты. В результате взаимодействия должны происходить трансформации объектов, а эти ожидаемые интуитивные представления об объекте показаны на рис. 3.

Объект снабжается уникальным именем, идентифицирующим его в предметной области. По сути, объект рассматривается как пара

$<$ редекс, контракт $>$.

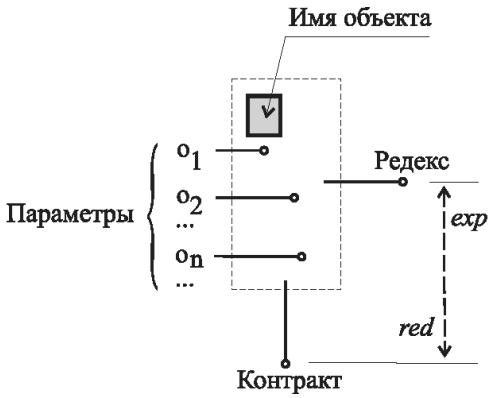


Рис. 3. Диаграмма объекта

Редексом считается исходный объект с последовательностью использованных действительных параметров, а *контрактом* — результат взаимодействия объекта со своими возможными параметрами. Редекс представляет объект *до* взаимодействия со средой, в то время, как контракт представляет объект *после* его взаимодействия со средой. Редекс является тем местом, где исходный объект рождается, начиная свое явное существование в среде. В ходе же межобъектного взаимодействия образуется контракт, а исходный объект в общем случае в нем не содержится, прекращая свое явное существование.

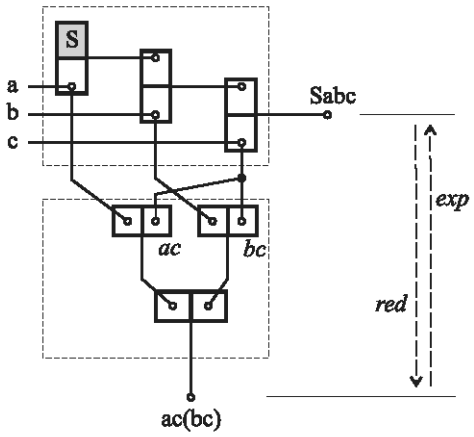
Взаимодействие со средой осуществляется через посредство параметров, которые, в свою очередь, являются объектами. Число параметров не ограничивается, а их влияние на объект проявляется по мере их использования в вычислениях. Процесс, ведущий к построению контракта, носит название *редукции* и символически обозначается через *red*, а процесс, ведущий к построению редекса — *экспансии* и обозначается через *exp*. Наибольший потенциал вычисления имеет редекс, а наименьший — контракт. В контракте имеется *виртуальное* представление исходного объекта, в то время как в редексе — его *действительное* представление.

Наиболее важным положением является то, что объект находится в процессе *конверсии*, то есть в двунаправленном процессе редукции-экспансии — переходе от состояния редекса к состоянию контракта и обратно. Редуцирование объекта связывают с построением результата вычисления, а экспансию — с формированием объекта, способного привести к имеющемуся результату вычисления [15].

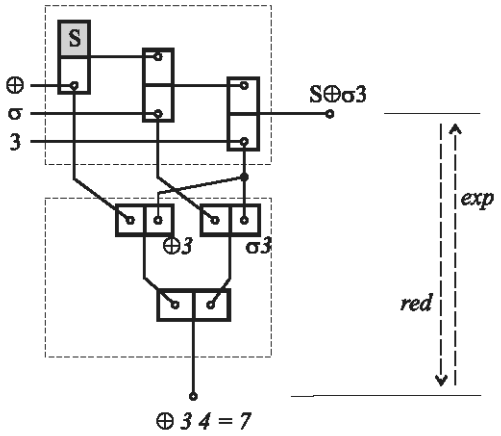
Для осуществимости этих преобразований, которые и происходят в действительности, приходится для простоты предполагать, что имеется *фиксированный* запас примитивных объектов, которые способны вызвать изменения в аппликативной среде. Идеально, в чистой теории объектов — комбинаторной логике, — обычно берется два таких объекта-комбинатора — S и K (см. [18, 19]).

Это ограничение нельзя признать существенным, а метод можно будет применять для разных форм комплексирования, когда будут разрешены иные объекты, находящиеся в иных средах. Будем предполагать, что сама операция аппликация и эти примитивные объекты в ходе взаимодействия объектов не изменяются. Один из этих активных объектов, вызывающий распределение вычисления по ветвям, представлен на рис. 4а.

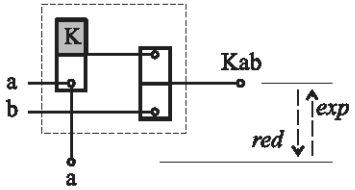
Примитивный и *константный* объект S во взаимодействиях с другими объектами в аппликативной среде проявляет арность, равную 3-м. Это означает, что он вынуждает свой первый аргумент воздействовать на третий, а второй аргумент в то же самое время воздействует также на третий. Эти взаимодействия идут по параллельным ветвям вычисления и относительно независимы. Точнее говоря, возможность выполнения вычислений в этих параллельных ветвях обусловлена *существованием* третьего объекта. При условии существования первого и второго аргументов, вычисления по ветвям «запускаются», как только установлено существование третьего аргумента. Это и означает, что объект S проявляется свою арность только во взаимодействиях с другими объектами, а число его аргументов заранее *не фиксировано* и, кстати



(a) Диаграмма распределителя S. Вычисления могут выполняться независимо друг от друга как в ветви ac, так и в ветви bc



(b) Пример трансформации аппликативной среды. Комбинатор S взаимодействует с объектами $\oplus$, σ и 3



(c) Диаграмма канцелятора K

Рис. 4. Исходные объекты-комбинаторы: распределитель S и канцелятор K

говоря, он может воздействовать и на другой экземпляр объекта S, то есть допускается *самоприменимость*. Если имеется три объекта, два из которых играют роль функций, а третий — роль аргумента этих функций, то в аппликативной среде S проявляет себя как трансформация, запускающая две ветви вычислений. Когда первая ветвь вычислений завершится, то она ожидает готовности вычисления во второй ветви. Если результаты обоих вычислений готовы, то они готовы и подвергнуться воздействию со стороны аппликативной среды. Тогда объект, порожденный вычислением в первой ветви, применяется, или, по иной терминологии, *апплицируется* к объекту, порожденному вычислением во второй ветви.

Итак, комбинатор S *распределяет* вычисления. Они могут выполняться *независимо* друга от друга как в ветви ac, так и в ветви bc: — имеется *виртуальный* объект a. Другими словами, нужно обеспечить лишь *возможность конструирования* объекта a; — имеется конструкция объекта b. Как только получен объект c, можно распараллелиться и построить конструкции для объекта ac и объекта bc.

Наконец, остается — по имеющимся или известным конструкциям ac и bc, — построить конструкцию объекта $ac(bc)$, применив конструкцию объекта ac к конструкции объекта bc. Этот процесс отражен на рис. 4a, который подчеркивает то обстоятельство, что объект-комбинатор S распределяет вычисления ценой своего существования — от редекса $S\,a\,b\,c$, где S содержится как действительный объект, выполняется редукция к контракту $ac(bc)$, где S уже не содержится.

Для того, чтобы лучше понять картину происходящих преобразований среды, рассмотрим пример, представленный конструкцией на рис. 4b.

Пример 1. Если a — операция $\oplus$ сложения для префиксной форме записи

$$\oplus x\,y = x + y,$$

b — операция σ прибавления единицы, или ‘следования за’

$$\sigma z = z + 1,$$

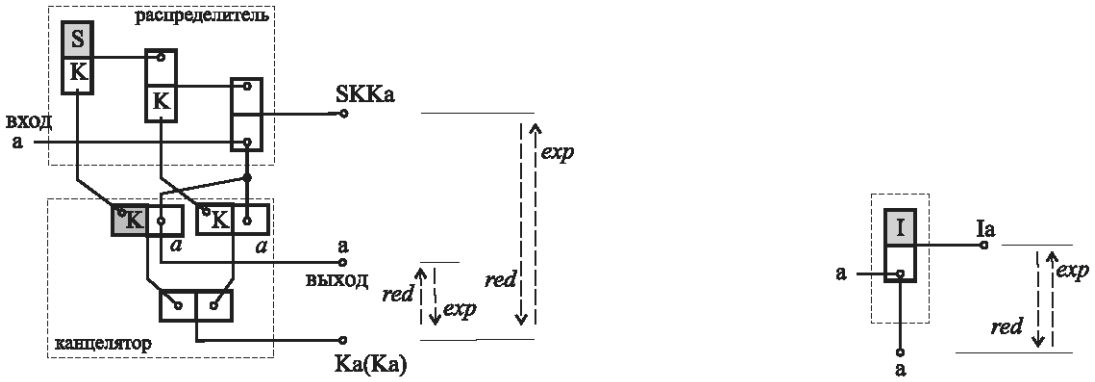
то для c, равного 3, получаем

$$\begin{array}{rcll} Sabc & \equiv & S \oplus \sigma 3 & \Leftarrow \text{ «старая» аппликативная среда} \\ & = & \oplus 3(\sigma 3) & \Leftarrow \text{ распределение вычислений} \\ & = & \oplus 3\,4 & \Leftarrow \text{ «новая» аппликативная среда} \\ & = & 3+4 & \Leftarrow \text{ применение операции} \\ & = & 7 & \Leftarrow \text{ формирование результата в среде} \end{array}$$

Эту конструкцию называют *распределением* вычислений, а объект S можно называть сокращенно распределителем, имея ввиду, что результат трансформации аппликативной среды, вызванной воздействием S, зависит от существования относительно независимых пар взаимодействующих объектов.

Другой исходный примитивный объект K, вызывающий изменение аппликативной среды, показан на рис. 4c. Этот комбинатор K во взаимодействии со средой «захватывает» в сферу своего воздействия два объекта, после чего выполняется трансформация. Она сводится к тому, что второй объект b прекращается свое существование в среде, а результатом является объект a в своем неизменном виде, который готов к взаимодействию со средой. Этот комбинатор называют канцелятором, поскольку он изымает из среды второй объект из пары, на которую он воздействует. Его вычислительный эффект — канцеляция, — состоит ровно в этом, поскольку первый объект этим воздействием никак не затрагивается.

Распределения и канцеляции являются основными строительными блоками диаграмм вычислений, а их итерирование посредством аппликации ведет к большим диаграммам, как показано на рис. 5a.



(a) Характеристика комбинатора SKK (b) Характеристика комбинатора I

Рис. 5. Производный комбинатор $I = SKK$: тождественное преобразование

Примитивные объекты помечены и это сделано для их различения и удобства ссылок на них.

Редукция red объекта $S\,K\,K\,a$ к объекту a в аппликативной среде осуществляется следующим образом. Во-первых, он проявляет арность 1, то есть для инициации взаимодействия со средой требуется произвольный объект a в сфере действия исходного объекта. Головным объектом является S, поэтому взаимодействие со средой и управляется этим комбинатором-распределителем. Во-вторых, распределенное таким образом вычисление в качестве головного элемента содержит K, поэтому далее взаимодействие со средой управляется этим комбинатором-канцелятором. Объект K, взаимодействуя со средой, устраняет второй объект, то есть $K\,a$, находящийся справа, прекращает свое существование. В то же самое время первый объект, а это объект a, передается на выход. Заметим, что произвольный объект a передается со входа на выход этой диаграммы безо всякого изменения. Другими словами, диаграмма на рис. 5a реализует *тождественное* преобразование произвольно взятого в аппликативной среде объекта. Получен тот же самый результат, что и в случае рис. 2a.

Экспансия exp объекта a до объекта $S\,K\,K\,a$ в аппликативной среде осуществляется иначе, но также в полном соответствии с диаграммой на рис. 5a. Этот объект — взятый в качестве

первого объекта, — может быть подвергнут воздействию со стороны исходного объекта — комбинатора K (см. рис. 4c). Но для сохранения «самотождественности» объекта a этот исходный объект-канцелятор потребуется снабдить и вторым объектом, вполне произвольно взятым из аппликативной среды. Выбор в качестве этого второго объекта $K\,a$ имеет то преимущество, что будет синтезирована схема распределения вычислений $K\,a\,(K\,a)$. Это распределение может быть расширено по схеме распределения S на рис. 4a. Ее применение надо прокомментировать: в качестве первого объекта воздействия для S надо взять K, в качестве второго — другой экземпляр K, а в качестве третьего — объект a, существование которого и предопределяет запуск распределенного вычисления. А это значит, что выполнен синтез объекта $S\,K\,K$, составленного исключительно из исходных примитивных объектов-комбинаторов по правилам формирования аппликации (см. рис. 2b). Кроме того, этот синтезированный объект оказывается комбинатором, но не исходным, а производным.

Полезным приобретенным навыком оказывается умение синтезировать объект, составленный из комбинаторов с заранее запланированным поведением относительно аппликативной среды вычислений. В данном случае, научившись синтезировать объект $S\,K\,K$, дающий тождественное преобразование, можно присвоить ему уникальное имя, например, I. Теперь взаимодействие со средой этого начавшего свое обитание в аппликативной среде объекта I можно представить диаграммой на рис. 5b. Выражаясь более строго, на нем представлена характеристика комбинатора I. Как видно, объект, поступающий на вход, редуцируется к самому себе, не подвергаясь преобразованиям. Это характеризует I как тождественное преобразование, ведущее себя аналогично тривиальной конструкции на рис. 2a.

Теперь производный объект-комбинатор I имеет равные с исходными объектами S и K права на существование в аппликативной среде. Отличие состоит в том, что существование S и K предполагалось с самого начала введения в рассмотрение аппликативной среды, но в ней также удалось не только «доказать» существование I, но и указать для него совершенно точную аппликативную структуру $S\,K\,K$. Это в известном смысле позволяет говорить, что объект-комбинатор, например, I, при необходимости можно искусственно синтезирован из начальной, «бедной» комбинаторами аппликативной среды, но комбинаторы — это особые объекты среды, управляющие ее функционированием. Оказывается, что I по своей структуре не элементарен, но скомплексирован из S и K исключительно, причем аппликативным способом. Последнее обстоятельство важно подчеркнуть, поскольку оно гарантирует, что сама по себе аппликативная среда не изменилась, а это мы установили в ней существование тождественного преобразования I. Этот объект — *возможный* относительно $<S,K>$-среды, но по своему статусу он также виртуальный в силу очевидных соображений, а у $<S,K,[I]>$-среды те же возможности, что и у исходной. Таким образом, в наших глазах добавление объекта I увеличивает закономерность в аппликативной среде: атомарно-молекулярный «материал» среды компьютинга стал разнообразнее.

§ 2. Внутренняя структура операций: является ли композиция элементарной операцией?

Попробуем теперь проникнуть вглубь структуры объектов, которые на первый взгляд кажутся существенно атомарными и неделимыми относительно тех преобразований, в которых они принимают участие. В частности, попытаемся «расщепить» — естественно, не материально, а относительно выделенной системы компьютинга, — всем известную операцию композиции.

Пусть имеются объекты a, b и c, в которые, например, вкладывается тот смысл, что a, b — операции σ прибавления единицы, или ‘следования за’ $\sigma z = z + 1$, а объект c равен 3. Для композиции a с b на аргументе c получаем

$$\begin{aligned}(a \circ b)\,c &\equiv (\sigma \circ \sigma)\,3\\ &= \sigma(\sigma\,3)\\ &= \sigma(4)\\ &= 5.\end{aligned}$$

В этом состоит общеизвестный смысл, вкладываемый в операцию композиции, а ее внутренней структурой обычно не озабочены и вовсе. Применяемая же техника вычислений позволяет иначе взглянуть на обычные операции, различая в них детали внутренней структуры. Попытаемся, например, ответить на вопрос, является ли композиция элементарной операцией или в рассматриваемой системе исходных объектов-комбинаторов генерируется как производный объект. Для этого возьмем объект $S\,(K\,S)\,K$, в состав которого входят уже известные объекты-комбинаторы, компьютинг с которыми в достаточной мере осмыслен и изучим, как этот объект может взаимодействовать с другими объектами, то есть установим его комбинаторную характеристику. Исходная конфигурация объектов приведена на рис. 6a, на котором последние два аргумента анализируемого объекта заключены в скобки: (b), (c). Основания для этого сводятся к следующему: поскольку комбинатор S в вычислениях проявляет арность, равную 3-м, то аргументы b, c, заключенные в скобки, на S-вычисление непосредственного влияния не оказывают.

Шаг 1-1: переход от $S\,(K\,S)\,K\,a$ *к* $K\,S\,a\,(K\,a)$. Итак, комбинатор S в состоянии взаимодействовать с тремя своими аргументами — $K\,S$, K и a. Вычисление распределяется, и в первой ветви генерируется аппликация $K\,S\,a$, а во второй ветви — $K\,a$. В этом заключается S-редукция, которая отражена на рис. 6a слева.

Шаг 1-2: переход от $K\,S\,a\,(K\,a)$ *к* $S\,(K\,a)$. Находящийся в первой ветви объект $K\,S\,a$ можно K-редуцировать, так как самый левый объект-канцелятор K располагает необходимыми для осуществления преобразования объектами-аргументами S и a. Результат взаимодействия объектов отражен на рис. 6a справа.

Шаг 2-1: переход от $S\,(K\,a)\,b\,c$ *к* $K\,a\,c\,(b\,c)$. Сгенерированная конфигурация объектов показана на рис. 6b слева. Осуществление взаимодействия объектов происходило следующим образом. В результате K-редукции предыдущего шага формируется результат S. Как известно, распределитель S формирует результат, аппликируя объект своей левой ветви к объекту правой ветви. Результат этой аппликации $S\,(K\,a)$ далее не редуцируется, поскольку у самого левого объекта S имеется только первый аргумент $K\,a$.

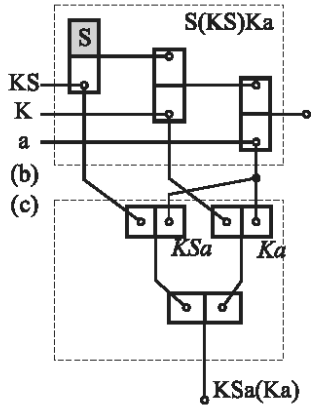
Для продолжения осуществления взаимодействия объектов понадобятся дополнительные объекты-аргументы b и c, которые на рис. 6a заключены в скобки. После этого оказывается, что у самого левого объекта S имеется все три объекта-аргумента — $K\,a$, b и c, — и все подготовлено для осуществления следующего шага редукции. Поскольку инициирует взаимодействие опять объект S, то вычисление S-редукции вновь распределяется, распадаясь на ветвь $K\,a\,c$ и $b\,c$.

Шаг 2-2: переход от $K\,a\,c\,(b\,c)$ *к* $a\,(b\,c)$. Объект первой из ветвей $K\,a\,c$ в качестве самого левого содержит объект-канцелятор K, располагающий необходимой для осуществления взаимодействия парой аргументов a и c, а выполнение K-редукции генерирует результат a. Это отражено на рис. 6b справа.

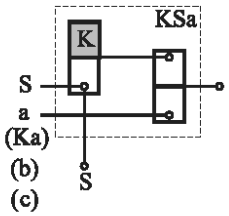
Объект второй из ветвей нередуцируем, поскольку нет никакой информации о структуре соответствующего самого левого объекта b, так что теперь предстоит выполнять аппликирование результата выполнения вычисления в первой ветви к результату вычисления во второй — a аппликируется к bc. Это будет в точности $a\,(b\,c)$, или в обычно применяемой инфиксной форме записи композиции $(a \circ b)\,c$.

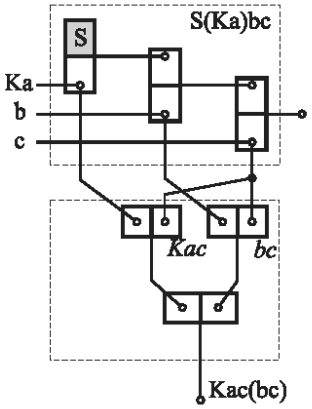
Тем самым показано, что $S\,(K\,S)\,K$ является виртуальным представлением операции композиции: по произвольным объектам a, b и c, но взятым именно в такой последовательности, объект $S\,(K\,S)\,K$ строит аппликацию объекта a к результату аппликации объекта b к c, а это и есть общеизвестный и общепринятый смысл композиции функций a и b для произвольного аргумента c второй функции. В данном случае результат S-редуцирования для первой ветви — это аппликирование $K\,a$ к c, а для второй ветви -— $b\,c$.

Шаг 3: формирование комбинатора композиции B*, позволяющего получить* $a\,(b\,c)$. Теперь нужно эту смысловую конструкцию зафиксировать в виде диаграммы, элементы которой уже подготовлены. Представление композиции найдено, и это объект $S\,(K\,S)\,K$. Остается для

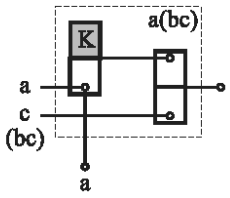


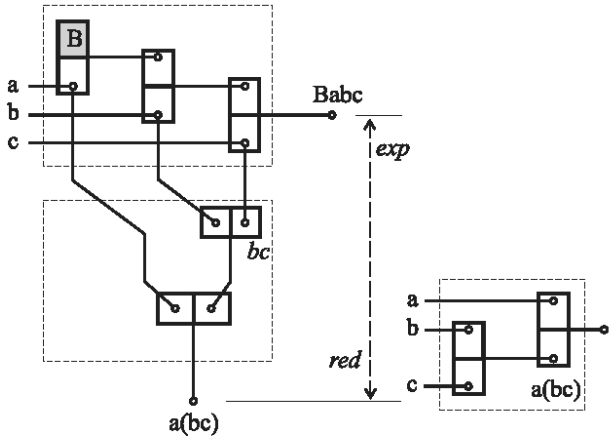
(a) Шаг 1: переход от $S(KS)Ka$ к $KSa(Ka)$





(b) Шаг 2: переход от $S(Ka)bc$ к $Kac(bc)$





(c) Шаг 3: формирование комбинатора композиции B, позволяющего получить $a(bc)$

Рис. 6. Синтез комбинатора композиции $B = S\,(K\,S)\,K$

сокращения ссылки на этот виртуальный объект $S\,(K\,S)\,K$ положить $B \equiv S\,(K\,S)\,K$, что полностью завершает процесс редукции и приводит к диаграмме на рис. 6c.

Коротко резюмируем вычислительный процесс в направлении редукции для объекта B, что соответствует выполнению анализа его потенциальной внутренней структуры.

Анализ вычислительного поведения комбинатора $S\,(K\,S)\,K$ дает следующий результат:

$$\begin{aligned} S\,(K\,S)\,K\,a\,b\,c &= K\,S\,a\,(K\,a)\,b\,c \\ &= S\,(K\,a)\,b\,c \\ &= K\,a\,c\,(b\,c) \\ &= a\,(b\,c) \end{aligned}$$

Это вычисление в направлении *редукции* по шагам отражено на рисунках 6a, 6b и 6c.

Только что проделанная редукция порождает вопрос, а откуда было заранее известно, что объект-композиция B имеет свое виртуальное представление $S\,(K\,S)\,K$ в мире взаимодействия объектов, порожденном объектами-комбинаторами S и K? Для того, чтобы снять подобные возражения, попробуем, начиная с *результата* композирования объектов a и b, записываемого посредством $(a \circ b)c \equiv a(b\,c)$, сгенерировать объект-композитор B. Это в точности означает, что B взаимодействует с последовательностью объектов a, b и c, редуцируясь в их композицию $a(b\,c)$.

Предстоит решать задачу синтеза объекта с заранее заданной характеристикой взаимодействия, а это требует выполнения не редукции, а *экспансии*. Символизация этого процесса позволяет кратко записать всю цепочку экспансий, ведущую к желаемому результату.

Синтез комбинатора B, имеющего характеристику композиции, выполняется следующим образом:

$$\begin{aligned} a(bc) &= K\,a\,c\,(bc) \\ &= S\,(K\,a)\,b\,c \\ &= K\,S\,a\,(K\,a)\,b\,c \\ &= S\,(K\,S)\,K\,a\,b\,c \\ &\equiv B\,a\,b\,c. \end{aligned}$$

Это вычисление в направлении *экспансии*. Забегая наперед, отметим, что для этого достаточно будет прочитать только что приведенные рисунки в обратном порядке: от рис. 6c к рис. 6b, а от него — к рис. 6a, а каждый из рисунков в отдельности — в направлении справа налево. Для того, чтобы проследить осуществление цепочки экспансий, начнем с рис. 6c, на котором приведена диаграмма для объекта B, фиксирующая его свойства.

Шаг 1: переход от $a\,(b\,c)$ *к* $K\,a\,c\,(b\,c)$. Располагаем результатом композиции объектов a и b, записанным в виде $a\,(b\,c)$ для произвольного аргумента c. Это исходная конфигурация объектов, а целевая, которую предполагается достигнуть в результате выполнения экспансии, представляет собой $(a \circ b)\,c$, или, что эквивалентно, $B\,a\,b\,c$. По чисто формальным соображениям надо просто «раскрыть скобки», в которые заключены объекты b и c. Единственный способ это проделать — использовать S-экспансию, но для этого нужно предварительно располагать дубликатом объекта c. А это нетрудно обеспечить, воспользовавшись K-экспансией объекта a. Этот процесс K-экспансии изображен на рис. 6b справа, что подготавливает последующее осуществление S-экспансии в соответствии с 6b слева.

Шаг 2: переход от $K\,a\,c\,(b\,c)$ *к* $S\,(K\,a)\,b\,c$. За раскрытие скобок пришлось заплатить появлением дубликата объекта c. Но этот дубликат можно исключить, а для этого понадобится выполнить S-экспансию соответствии с рис. 6b слева.

Шаг 3: переход от $S\,(K\,a)\,b\,c$ *к* $K\,S\,a\,(K\,a)\,b\,c$. Вновь понадобится раскрыть скобки, — а они *существенные*, — в которые заключена аппликация $K\,a$. Для этого выполняется K-экспансия в соответствии с рис. 6a справа.

Шаг 4: переход от $K\,S\,a\,(K\,a)\,b\,c$ *к* $S\,(K\,S)\,K\,a\,b\,c$. Результатом осуществления предыдущего шага явился дубликат объекта a, который понадобится устранить. Для этого выполняется S-экспансия в соответствии с рис. 6a слева.

На последнем шаге целевая конфигурация объектов достигнута. Синтезирован объект $S\,(K\,S)\,K$, а по своей структуре — это объект-композитор, осуществляющий всем известную операцию композиции. Теперь этому синтезированному объекту можно присвоить уникальное имя B, а объект B использовать на равных правах с объектами S и K с той лишь разницей, что S и K — исходные константные объекты, а B — производный.

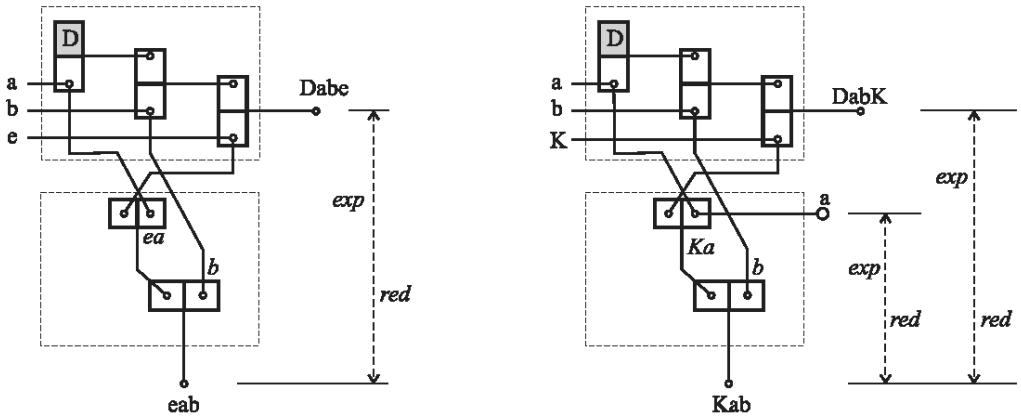
И вновь, как и в случае раздела 1 в конце, добавление объекта B — аппликативного образа композиции $\circ$, — усиливает видимое многообразие атомарно-молекулярного материала среды компьютинга: к исходным объектам-комбинаторам S и K оказались, скажем, последовательно добавлены I (тождественное преобразование) и B (композиция).

§ 3. Условные конструкции

Напоминает ли аппликативная среда обычные конструкции программирования? Объекты взаимодействуют друг с другом, надо бы иметь возможность их тестировать и переключать среду их взаимодействия соответствующим образом в зависимости от результатов тестирования. В аппликативной среде нет необходимости вводить дополнительные исходные объекты, осуществляющие переключение взаимодействия. Как оказалось, уже имеющиеся исходные примитивные объекты позволяют выразить такие всем известные конструкции, как условные выражения: пользуясь экспансией, их можно синтезировать, а далее — либо включить в систему объектов среды компьютинга как *действительные* объекты, либо сохранить за ними статус *виртуальных* объектов. В данном случае все зависит от целей конструируемой системы компьютинга и среды вычислений. Для подобных объектов-переключателей для простоты будем

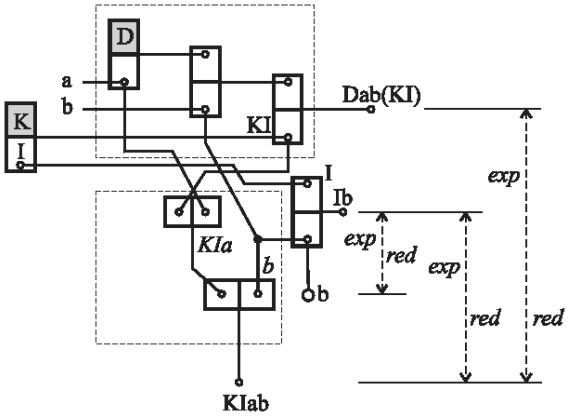
рассматривать только заранее фиксированный их набор. Но такое ограничение не является принципиальным, а метод комплексирования диаграмм будет также применим и для иных видов компонования переключателей. Кроме того, полагаем, что воздействие переключателя на объект не ведет к изменению последнего, а проходящий через переключатель объект, хотя и тестируется, но не подвергается преобразованию.

На роль такого переключателя подходит объект $D\,a\,b$, представленный на рис. 7a.



(a) Комбинатор пары D: $Dabe = eab$

(b) Первая проекция пары: $DabK = a$



(c) Вторая проекция пары: $Dab(KI) = Ib = b$

Рис. 7. Характеристика пары $D\,a\,b$

Как следует из этого рисунка, объект D воздействует на три объекта a, b и e, организуя их взаимодействие. Объект же $D\,a\,b$ представляет собой тест для e, в зависимости от результата которого выбирается либо объект a, либо объект b, но не оба сразу. По своему действию $D\,a\,b$ напоминает всем известную условную конструкцию *if e then a else b*:

$$\begin{array}{c} \textit{if } e \textit{ then } a \textit{ else } b \\ D\,a\,b\,e = \begin{cases} a & \text{для } e \equiv K; \\ b & \text{для } e \equiv K\,I. \end{cases} \end{array}$$

Выбор альтернативы происходит в зависимости от значения e. В случае, если значением e является объект K, в качестве результаты выполнения переключения берется объект a, а если значением e является объект $K\,I$, то b.

Из рис. 7b видно, что комбинатор K играет роль логической константы *true*, а из рис. 7c видно, что комбинатор $K\,I$ играет роль логической константы *false*.

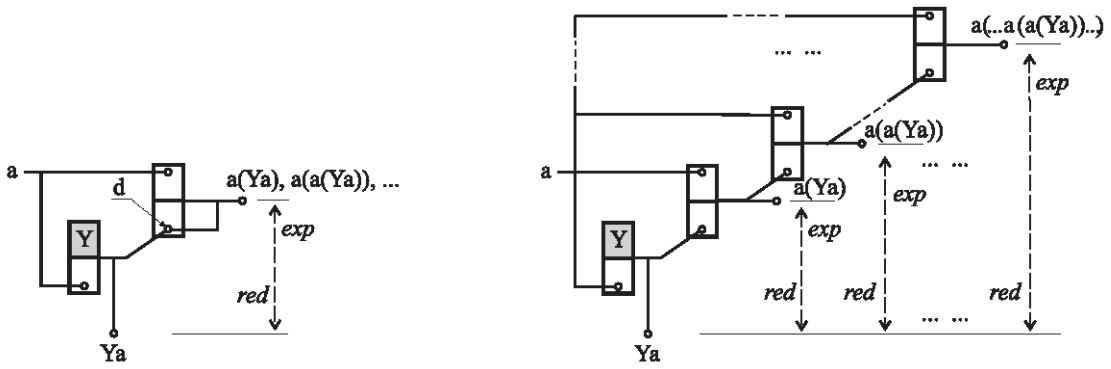
Заметим, что при необходимости объект-комбинатор D можно также синтезировать через S и K. Для этого можно, например, пойти таким путем: сперва синтезировать объект $C\,abc = a\,c\,b$, где $C \equiv S\,(B\,B\,S)\,(K\,K)$, $B \equiv S\,(K\,S)\,K$, затем принять, что $D \equiv B\,C\,(C\,I)$ и т. д.

КОМПЬЮТЕРНЫЕ НАУКИ 2009. Вып. 2

§ 4. Бесконечные конструкции

Может сложиться впечатление, что в рассматриваемой среде вычислений имеют право на существование только конечные конструкции, но это не так.

На рис. 8a показана наиболее хорошо известная конструкция, которая позволяет циклически выполнять преобразования: так называемая конструкция с *неподвижной точкой*[1]. Она представляет, как всем известно, одну из основных идей преобразований вообще. С интуитивной точки зрения, это одно из наипростейших представлений: объект a попадает в сферу действия другого объекта, то есть Y. В области аппликирования Y объект a подвергается преобразованию и результат преобразования направляется в сферу действия объекта a. В свою очередь, результат a-преобразования объекта $Y\,a$ вновь направляется в сферу действия объекта a для повторного преобразования, выполняемого в цикле. Поскольку на такое циклическое преобразование ограничений не накладывается, то оно повторяется раз за разом.



(a) Конечная конструкция, содержащая цикл

(b) Бесконечная конструкция

Рис. 8. Характеристика комбинатора Y

Ни одна из до сих пор встречавшихся конечных конструкций не имеет подобного вида. Может сложиться впечатление, что какие-то детали оказались просто упущены. Но это не так, и класс объектов полон. Для подтверждения этого наблюдения, зададимся вопросом, не является ли конструкция на рис. 8a сокращением для более простых диаграмм. Имеется ряд способов для сокращения рисования диаграмм, которые избавляют от громоздких повторов. Циклы просто являются предельным случаев повторений. Действительно, вместо обратного перенаправления объекта с выхода конструкции на ее вход, преобразование можно изображать снова и снова, до бесконечности. Бесконечная конструкция преобразований показана на рис. 8b.

Очевидно, что бесконечная конструкция даст тот же самый результат, что и циклическая. Но в действительности это нуждается в доказательстве. Имеется ли точный смысл в приведенных соображениях? Для подтверждения этого потребуется произвести некоторую символизацию. Выполняемое преобразование символизируется как a, используется комбинатор Y. Выход конструкции, полученный в результате применения Y к a, назовем d. Вернемся к рис. 8b. После первого преобразования конструкция сама себя воспроизводит. Этот простой штамп легко записать символически:

$$d = a(d).$$

Другими словами, преобразование выполняется неограниченное число раз, а конструкция содержит сама *себя* как собственную часть.

Полученный результат всем хорош, но только остается выяснить, действительно ли имеется такой комбинатор Y, который по преобразованию a позволяет находить его *неподвижную точку* d.

[1]Говорят, что отображение f имеет неподвижную точку p, если выполняется $p = f(p)$.

Комбинатор Y позволяет строить конструкции, содержащие *неподвижную точку* в соответствии с характеристикой

$$Y\,a = a\,(Y\,a),$$

то есть, $Y\,a = a\,(Y\,a) = a\,(a\,(Y\,a)) = \ldots = a(\ldots(a\,(a\,(Y\,a)))\ldots)$. В качестве объекта Y возьмем $S\,(B\,W\,B)\,(B\,W\,B)$, то есть положим $Y \equiv S\,(B\,W\,B)\,(B\,W\,B)$. Комбинаторные характеристики применяемых объектов B, S и W заданы следующим образом: $B\,a\,b\,c = a\,b\,c$, $S\,a\,b\,c = a\,c\,(b\,c)$, $W\,a\,b = a\,b\,b$. Приложим сконструированный таким образом объект Y к a:

$$\begin{aligned}
Y\,a &\equiv S\,(B\,W\,B)\,(B\,W\,B)\,a && \text{(по предположению)}\\
&= B\,W\,B\,a\,(B\,W\,B\,a) && \text{(по схеме } S\text{)}\\
&= W\,(B\,a)\,(B\,W\,B\,a) && \text{(по схеме } B\text{)}\\
&= B\,a\,(B\,W\,B\,a)\,(B\,W\,B\,a) && \text{(по схеме } W\text{)}\\
&= a\,(B\,W\,B\,a\,(B\,W\,B\,a)) && \text{(по схеме } B\text{)}\\
&= a\,(S\,(B\,W\,B)\,(B\,W\,B)\,a) && \text{(по схеме } S\text{)}\\
&\equiv a(Y\,a) && \text{(по предположению)}
\end{aligned}$$

Как оказалось, в преобразованиях обнаружена периодически повторяющаяся часть, причем она в точности совпадает с комбинатором Y. Таким образом, действительно существует объект Y, выражаемый через известные комбинаторы и такой, что $Y\,a$ является неподвижной точкой для произвольного объекта a.

Заключение

За разнообразием форм, в которых ныне себя проявляет компьютинг, непросто увидеть единство информационных сущностей и немногие общие механизмы, приводящие в движение потоки цифровой информации. Обнаруживается отчетливая тенденция реструктурирования компьютинга, прежние формы которого породили больше вопросов, чем дали ответов. Начиная с простых и непосредственно воспринимаемых идей о диаграммах объектов, построена расширяемая среда вычислений. Структурно она себя воспроизводит каскадированием компонентных диаграмм, а управление компьютингом осуществляется конечным набором исходных объектов-комбинаторов. «Разрешающая способность» вычислительной системы также оказывается регулируемой, позволяя не ограничиваться только анализом внутренней структуры исходных операций, что также немало, но и синтезировать объекты с заранее запланированным вычислительным поведением.

Благодарности

Благодаря многочисленным обсуждениям с А.Г. Пантелеевым и к.ф.-м.н. Ю.Р. Габовичем удалось добиться более отчетливого понимания роли и места объектов в системе компьютинга. Выражаю признательность проф. Х. Барендрегту, проф. Р. Хиндли и проф. Дж. Селдину за обсуждение ряда вопросов, связанных с аппликативными вычислительными технологиями. В особенности это касается идей о связях естественного компьютинга, многообразия известных моделей вычислений и комбинаторной логики.

СПИСОК ЛИТЕРАТУРЫ

1. Kennedy A. Functional Pearls: Pickler Combinators. // Journal of Functional Programming, special issue on Functional Pearls. Cambridge University Press. — Nov 2004. — Vol. 14. — № 6. — P. 727–739.

2. Barendregt H., Wiedijk F. The Challenge of Computer Mathematics. // Transactions of the Royal Society —2005. — Vol. 363. — № 1835. — P. 2351–2375.
ftp://ftp.cs.ru.nl/pub/CompMath.Found/Barendregt-Wiedijk.pdf

3. Hindley J. R., Lercher B., Seldin J. P. Introduction to Combinatory Logic. — London: Cambridge University Press, 1972.

4. Scott D. S. The Lattice of Flow Diagrams. // Lecture Notes in Mathematics, 188, Symposium on Semantics of Algorithmic Languages. — Berlin, Heidelberg, New York: Springer-Verlag, 1971. —P. 311–372.

5. Seldin J. P. The Logic of Curry and Church. // Handbook of the History of Logic /Dov Gabbay and John Woods, eds./ — Elsevier. — 2006. — Vol. 5.
http://people.uleth.ca/ jonathan.seldin/CCL.pdf

6. Шаумян С. К. Аппликативная грамматика как семантическая теория естественных языков. — М.: Наука, 1974. — 204 с.

7. Selinger P., Valiron B. A Lambda Calculus for Quantum Computation with Classical Control // Mathematical Structures in Computer Science. — 2006. — Vol. 16. — №. 3 — P. 527–552.

8. Bell G., Dourish P. Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. // Personal Ubiquitous Comput. — 2007. — Vol. 11 — № 2. — P. 133–143.
http://dx.doi.org/10.1007/s00779-006-0071-x.

9. MacLennan B. J. Molecular Combinator Reference Manual. // UPIM Report 2, Technical Report UT-CS-02-489, Department of Computer Science, University of Tennessee. — Knoxville, 2002. — 16 p.
http://www.cs.utk.edu/ mclennan/UPIM/CombRef.pdf

10. MacLennan B. J. Combinatory Logic for Autonomous Molecular Computation. — Preprint of the paper invited for Information Sciences, 2003.
http://www.cs.utk.edu/ mclennan/UPIM/CLAMC-IS.pdf

11. A Framework for Web Science /Berners-Lee T., Hall W., Hendler J., O'Hara K., Shadbolt N., Weitzner D. // Foundations and Trends in Web Science. — 2006. — Vol. 1, Issue 1. — P. 1–130.
http://www.nowpublishers.com/web/

12. Carpenter B. The Internet Engineering Task Force: Overview, Activities, Priorities. — ISOC BoT, 2006-02-10, 2006.
http://www.isoc.org/isoc/general/trustees/docs/Feb2006/IETF-BoT-20060210.pdf

13. Cardelli L., Davies R. Service Combinators for Web Computing. // HP Labs Technical Reports SRC-RR-148. —1997. — June 1. — 15 p.
http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-148.html

14. Denning P. J. Computing is a Natural Science. // Commun. ACM. — 2007. — Vol. 50. — № 7. — P. 13–18.
http://doi.acm.org/10.1145/1272516.1272529

15. Вольфенгаген В. Э. Конструкции языков программирования. Приемы описания. — М.: АО «Центр ЮрИнфоР», 2001. — 276 с.
Издание поддержано грантом РФФИ, проект 01-01-14068-д.

16. Вольфенгаген В. Э. Комбинаторная логика в программировании. Вычисления с объектами в примерах и задачах. — М.: МИФИ, 1994. — 204 с.; 2-е изд., М.: АО «Центр ЮрИнфоР», 2003. — 336 с.

17. Вольфенгаген В. Э. Методы и средства вычислений с объектами. Аппликативные вычислительные системы. — М.: JurInfoR Ltd., АО «Центр ЮрИнфоР», 2004. — xvi+789 с.
Издание поддержано грантом РФФИ, проект 03-01-14055-д.

18. Исмаилова Л. Ю. Логика объектов. // Методы и средства вычислений с объектами. Аппликативные вычислительные системы. — М.: JurInfoR Ltd., АО «Центр ЮрИнфоР», 2004. — с. 613–630.

19. Косиков С. В. Логика функциональности. // Методы и средства вычислений с объектами. Аппликативные вычислительные системы. — М.: JurInfoR Ltd., АО «Центр ЮрИнфоР», 2004. — с. 595–612.

20. MacLennan B. J. Molecular Combinatory Computing for Nanostructure Synthesis and Control. — IEEE Nano 2003, San Francisco, August 12-14, 2003.
http://www.cs.utk.edu/ mclennan/UPIM/MacLennan-MCCNSC.pdf

КОМПЬЮТЕРНЫЕ НАУКИ 2009. Вып. 2

Поступила в редакцию 03.03.09

V. E. Wolfengagen, L. Yu. Ismailova, S. V. Kosikov, A. D. Laptev, V. N. Nazarov, V. V. Roslovtsev, I. S. Safarov, A. L. Stepanov

Combinators: objects for knowing the structure of computing. Atomic and molecular granularity of computing environment

In everyday computing the entities are involved into operations whose inner structure is not paid much of attention. Nevertheless many of usual operations consist of more primitive constructs combined by a mode of combining. An interaction of constructs take place within environment of «applicative interaction» and studying the properties of this environment allows to get familiar with the nature of computations.
This paper pays main attention to knowing the technological features of computations with objects. Their interaction is assumed in applicative environment allowing to know the intrinsic structure of usual operations and this knowledge allows to know their properties in turn. The choice of primary constant entities is discussed and these entities are assumed as initial and are referred to as combinators. These primary entities are used as main "building blocks" participating in interaction each with other in the applicative environment. This interaction results in the constructs giving rise to the representative sets of usual operators and embedded computational systems.

Keywords: combinatory logic, computing, applicative environment, embedded computational systems.

Mathematical Subject Classifications: 03B40, 68Q05, 68Q85, 68N18

Вольфенгаген Вячеслав Эрнстович,
д.т.н., проф., заведующий кафедрой перспективных компьютерных исследований и информационных технологий, группа компаний «ЮрИнфоР»,
119435, Россия, г. Москва, М. Пироговская ул., д.5, офис 37,
E-mail: vew@jmsuice.msk.ru

Исмаилова Лариса Юсифовна,
к.т.н., в.н.с. кафедры кибернетики МИФИ,
115409, Россия, г. Москва, Каширское шоссе, д. 31,
E-mail: larisa@jurinfor.ru

Косиков Сергей Владимирович,
с.н.с. кафедры кибернетики МИФИ,
115409, Россия, г. Москва, Каширское шоссе, д. 31,
E-mail: kosikov_s_v@mtu-net.ru

Лаптев Андрей Дмитриевич,
аспирант кафедры кибернетики МИФИ,
115409, Россия, г. Москва, Каширское шоссе, д. 31,
E-mail: info@jurinfor.ru

Назаров Виктор Николаевич,
аспирант кафедры кибернетики МИФИ,
115409, Россия, г. Москва, Каширское шоссе, д. 31,
E-mail: info@jurinfor.ru

Рословцев Владимир Владимирович,
аспирант кафедры кибернетики МИФИ,
115409, Россия, г. Москва, Каширское шоссе, д. 31,
E-mail: info@jurinfor.ru

Сафаров Ильдар Сергеевич,
аспирант кафедры кибернетики МИФИ,
115409, Россия, г. Москва, Каширское шоссе, д. 31,
E-mail: info@jurinfor.ru

Степанов Александр Леонидович,
зав. лаб. факультета информационной безопасности МИФИ,
115409, Россия, г. Москва, Каширское шоссе, д. 31,
E-mail: a-stepanov@ichtiosfera.ru