

Структура компьютинга и конструирование вычисления

08, август 2010

авторы: Вольфенгаген В. Э., Исмаилова Л. Ю., Косиков С. В.

УДК.004

Кафедра ПКИ и ИТ, Институт “ЮрИнфоР-МГУ”, Москва, 119437 РФ,
vew@jmsuice.msk.ru,
lyu.ismailova@gmail.com,
kosikov.s.v@gmail.com,

Введение

Компьютинг и его развитие ставит целый ряд вопросов, на большую часть из которых ответы либо неполны, либо неизвестны. Некоторые из них: что такое ‘вычисление’? что такое ‘информация’? что можно узнать, пользуясь компьютером? чего нельзя узнать, пользуясь компьютером? – имеют фундаментальное значение.

Эти вопросы сопровождали компьютер, начиная с 1940-х гг. Казалось, на них имеются ответы, но и сегодня эти же самые вопросы задаются всеми и повсюду, во всех областях науки, инженерии, бизнеса и даже политики.

Долгое время бытовала традиция, в соответствии с которой компьютер рассматривался как наука о явлениях, сопровождающих компьютеры, и этот взгляд не вызывал сомнения. Компьютер всегда был и остается наукой об информационных процессах. Приблизительно с 1995 г. специалисты из разных областей науки, один за другим, стали заявлять, что ими в их же области обнаружены естественные информационные процессы. Эти открытия ввели в обиход иную традицию, в соответствии с которой компьютер стал считаться наукой о естественном и об искусственном одновременно.

По старой традиции компьютер наиболее естественно описывался идеями из *базовых технологий* – программирования, графики, сетей и суперкомпьютера. Нынешняя же традиция настоятельно требует выразить компьютер в терминах фундаментальных принципов или даже вывести компьютер как таковой из некоторых фундаментальных принципов. Если компьютер выводить из принципов, то не только вскроются его глубокие структуры, но также прояснятся и их применимость в других областях науки. При этом также вскрываются общие аспекты отдельных технологий, создавая возможности для инноваций. Вместе с тем открываются принципиально новые пути стимулирования во многом утраченного интереса к компьютеру среди молодежи.

В 1940-х гг. вычисления рассматривались просто как инструмент для решения уравнений, расшифровки кодов, анализа данных и управления бизнес-процессами. Но в 1980-е гг. вычисления развились до такой степени, что превратились в новый научный метод, соединяя традиционное понимание теории с экспериментом. А в 1990-е гг. последовал следующий сдвиг в понимании роли и места компьютера, поскольку многие исследователи в разных научных областях пришли к выводу, что они столкнулись с информационными процессами в естественнонаучных глубинных структурах. Например, это и квантовые эффекты в физике, ДНК в биологии, мыслительные процессы в когнитологии, потоки информации в экономике. Вычисления вошли в жизни вместе с новыми способами решения задач, новыми формами искусства, музыки, кино, а также вместе с новыми формами коммерции, новыми подходами к обучению и даже новым сленгом, на котором стали говорить.

Обозначенные в самом начале фундаментальные вопросы компьютера стали важными и во множестве тех областей, в которых люди в своей работе существенно опираются на вычисление и вычислительные методы. На самом деле, изучение аспектов компьютера, приносящих наибольшую пользу в традиционных науках, больше всего помогает установлению фундаментальных основ и принципов самого компьютера.

Метафорические речевые обороты повседневной речи пополнились фразами наподобие: “я запрограммирован на такое поведение” или “мои мозги засбоили и требуют перезагрузки”. Бурное *прорастание* компьютера во все сферы повседневной жизни привело даже к тому, что от студентов Вашингтонского университета стали требовать “беглого владения информационными технологиями”, что предполагает их хорошее знание и умение применять в различных ситуациях. Возникла потребность и требования к “вычислительному мышлению”, что предполагает использование принципов компьютера и в науке, и в жизни. Вычисления проникли повсюду, а также стали всюду

обнаруживаться.

У установления принципов и объяснения происходящего, исходя из них, есть еще одно преимущество по сравнению с технологиями: принципам легче научить, чем технологиям. Описание сферы деятельности на языке технологий хорошо срабатывало раньше, когда известных базовых технологий было относительно немного. Например, Ассоциацией по вычислительной технике в 1989 г. насчитывалось 9 базовых технологий, а 2001 г. их стало уже 36, образуя 630 прямых взаимосвязей, что стало проблематичным для непосредственного изучения по прежним образовательным шаблонам.

На сегодня пока приходится констатировать, что компьютеринг не сумели выразить в терминах фундаментальных принципов, он все еще остается рецептурно-технологическим. В других же науках сложилась иная ситуация, в них круг явлений устанавливается, исходя из установленных базовых принципов. Это свидетельствует об известной зрелости таких наук, но не компьютеринга, который только-только достигает состояния в своем развитии, находясь в котором можно говорить о его принципах.

1 Инварианты компьютеринга

Установление принципов компьютеринга представляет собой вовсе непростой процесс. Когда в компьютеринге принялись за дело и он стал интенсивно разрабатываться, то его сущность казалась само собой разумеющейся, а продуцировались его формы. Возникший в 1970-е гг. бум *моделей вычислений* не прекращается по настоящее время. В начале этого бума ничто не сдерживало разработчиков, а открывающиеся возможности сулили безграничные перспективы.

Все же время от времени возникал и возникает вопрос, как устроен компьютеринг, но от него чаще всего просто отмахивались. Подобный интерес выглядит чисто академическим, а согласно бытующему мнению, информационные и вычислительные технологии – это удел больших компаний и мощных коллективов. Но общая картина компьютеринга только выиграет, если в нем удастся обнаружить те *инварианты*, которые сохраняются при смене форм.

Инварианты играют роль глобальных констант, а с вычислительной точки зрения – примитивных ‘строительных блоков’, пользуясь которыми можно сконструировать тот или иной ‘мир’ компьютеринга. Поскольку на каждую из имеющихся форм компьютеринга его разработчики четко и жестко заявляют свои права, то и разработчики, и пользователи ревностно отстаивают сконструированный ими мир. Они относятся к этому миру, как к среде своего информационного обитания, пресекая попытки вторжения в него извне, а если бы удалось установить единство форм, то это способствовало бы улучшению взаимопонимания в разноплеменном информационном сообществе.

То, что подобные инварианты существуют, было известно. Вопрос ставится иначе, как ими воспользоваться. Действительно, стоит только вспомнить о комбинаторах, сравнительно недавно открытыми в мире метаматематики, то это подкрепляет уверенность в имеющемся единстве форм. С интуитивной точки зрения *среда* вычислений вмещает в себя все или почти все, что относится к построению *результата*: в ней имеются переменные и их фактические значения, а сами они разнесены не только позиционно, но и контекстно.

Все, что делается в компьютеринге, может быть сведено к некоторой первооснове, которая многими признается, а абсолютным большинством не отвергается: берется то, что считается *идентификатором*, и для него *относительно среды строится* то, что будет считаться его *значением*. Вычислением считается именно этот процесс построения, а сам компьютеринг разрабатывает технологии осуществления построения. Итак, отношение между идентификатором и его значением параметризовано средой. Число всевозможных комбинаций идентификатор-значение столь велико и внушительно, что не удастся построить гипотетическую таблицу, пользуясь которой можно все относящееся к идентификатору рассчитывать. Да, для осуществления подобной стратегии расчета пришлось бы завести огромную базу данных, которая находилась бы в состоянии непрерывной модернизации и модификации. На современном уровне понимания это считается технологически неприемлемым. С позиций теории баз данных и реляционной модели отношение рассматривается как *явное* перечисление всевозможных комбинаций индивидов, в которых они могут оказаться, оставаясь в рамках именно этого отношения. С математической точки зрения речь идет об отображении, для которого *заранее известны* область определения – его *домен*, – и область значения – его *диапазон*. Мало того, что они велики по объему, но еще и подвержены изменениям, а разработка теории отображений с переменными доменами-диапазонами находится в эмбриональном состоянии. Д. Скотт в (D.S. Scott, [17]) предложил рассматривать конструкции *переменных доменов*, но в области компьютерных наук его не поддержали, поскольку разразившийся несколько позже компьютеринговый бум сулил иные многообещающие перспективы сразу по многим направлениям, а за получение плодов не надо было вносить обременительную плату разработки сложной и междисциплинарной теории, продуктивность которой еще и надо были отстаивать.

В комбинаторной логике рассуждали иначе, полагая, что надо заняться конструированием собственно отображений, не заботясь о существовании их областей определения и областей значения и усилиями М. Шейнфинкеля (M.I. Schoenfeld, [15]) и Х. Карри (H.B. Curry, [6]) была разработана такая теория. Более радикальное намерение состояло в том, чтобы отыскать минимальный набор математических сущностей, пользуясь которыми можно было бы сконструировать все здание современного математического знания, поскольку иные формы знания не рассматривались и считались не имеющими право на то, чтобы с ними всерьез считались. Важно, что открытые ими *комбинаторы* лежали в основе любого математического и метаматематического рассуждения. Позже, по мере развития информационных технологий, было осознано их поистине

фундаментальное значение для компьютеринга.

В традиционном компьютеринге центральным понятием, без которого, как обычно считается, нельзя обходиться, является представление о *переменной*. Переменная играет роль ‘числа вообще’, что помогает строить общие утверждения и анализировать их свойства. Сразу же было осознано, что переменные надо рассматривать шире, считая их ‘сущностями вообще’, некими индетерминантами, не сводя их сферу действия только к чистой арифметике. Комбинаторы же воспринимаются как ‘константы вообще’, предполагая, что строение знания по своей фактуре гранулировано именно константами. Вместе с тем не является неожиданными, что ни в математике, ни в компьютеринге не располагают надежным определением ни переменной, ни константы. Такая ситуация подрывает доверие к накопленным в этих областях фундаментальным истинам, формальное выражение которых с необходимостью опирается и на константы, и на переменные.

Особенное отношение к переменным как к конструктивам, без которых никак нельзя обойтись, породило современную реалию информационных технологий, выраженную в языках программирования. Пристрастие к переменным порождает трудности принципиального характера, когда речь идет о применении компьютеринга. Закрепление знания по форме осуществляется в *книгах*, способ выражения и устройство которых считается настолько общеизвестным, что, фактически, не подвергается изучению. Исключением является проект Automath Н. де Брейна (N. G. de Bruijn, [8]). Стартовав в 1967 г., он преследовал цель построить среду выражения математических теорий в форме, пригодной для компьютерной проверки их корректности. В основе лежала гипотеза, что если утверждение выражено правильно, то оно и является правильным. Других норм правильности не вводилось. Не надо только забывать, что ‘правильный’ означает попросту ‘основанный на правилах’ и остается только разобраться с вопросом, что такое *правило*, но это самый дискуссионный вопрос не только современной метаматематики, но и компьютеринга.

Считается, что язык по структуре образует *логику*, что немедленно приводит к необходимости разбираться с ее фундаментальными трудностями, которые хотя и известны, но нельзя считать их вполне осознанными. Предположительно в Automath не использовалось ни логики, ни математических оснований. В этом проекте весь математический материал образовывал *книги*, написанные на языке, а вот язык базировался на лямбда-исчислении с типами, в терминах – в *правилах*, – которого выражались представления об ‘определении’, ‘теореме’, ‘доказательстве’, ‘аксиоме’. Книга представляет собой конструкцию, состоящую из вложенных блоков, причем открывание блока соответствует введению *типовой переменной*. Переменными представлены математические объекты или математические доказательства, а система их взаимных связей строится совершенно аналогично. Другими словами, с самого начала в проекте заложена идея *доказательство-как-объект*.

На долю логического метасредства приходится – всего лишь, – интерпретация этой блочной структуры. Открытым остается вопрос, много это или мало. Достаточно ли, чтобы привнести в выраженный таким способом текст обременяющие метаматематику фундаментальные трудности? Однако, по форме своего выражения проект представляет инновационный способ рассматривать связь логики с математикой, по крайней мере, с позиций возможностей современного компьютеринга. Обработка же книг, выражающих математическое знание в языке Automath, становится подкупающе простой и естественной для доминирующей общематематической практики. Это дает известные дидактические и образовательные возможности: преподавание математики таким образом, чтобы студенты могли ее изучить, получает точную основу в имеющихся и развиваемых информационных технологиях. Более того, преподавание из разряда искусства переходит в разряд технологии, когда машине достаточно ‘растолковать’ – на языке проекта, – конструктивное устройство текста по форме, но не по содержанию. При этом никакого предварительного логического или математического ‘подтекста’ не привносится, а заключение о правильности самого текста возлагается на среду компьютеринга. По замыслу, подход не охватывает автоматизацию ни процесса самого математического изобретения, ни процесса поиска доказательства сформулированных теорем: Automath играет роль внимательного читателя правильно представленного по форме материала.

Как оказалось, в компьютеринге понадобилось разработать *виртуального читателя*, который будет правильно осуществлять процесс чтения *виртуальной книги*, которой должна быть придана правильная форма. Это обусловлено чрезвычайным усложнением математизированного знания, когда его усвоение или оценка правильности превышают обычные человеческие способности. Но не возможности, поскольку, беря на помощь компьютеринг и его среду, знание из *действительного* превращается в *возможное*. В новейших информационных технологиях это возобновило повышенный интерес к семантическим сетям (Т. Berners-Lee et al., [3]).

2 Новые парадигмы компьютеринга

Эксплуатируемый ныне смысл, вкладываемый в термин ‘компьютеринг’, хотя и с изменениями, но соответствует тому пониманию, которое сложилось 60 лет назад при возникновении Ассоциации вычислительной техники АСМ (Association for Computing Machinery). Намечившиеся в новейшее время сдвиги в понимании оказываются довольно существенными, имея три общие характеристики: компьютеринг не обязательно осуществляется только на основе технологии кремниевых интегральных схем; основные вычислительные элементы реализуются физически, став не просто теорией; осуществлен переход к новым порогам миниатюризации, часто составляющим от 1 до 100 нанометров. В этих условиях сложившиеся представления о компьютеринге начинают пересматриваться, но это вовсе не означает отказа от имеющихся представлений или технологий.

Некоторые из новых форм осуществления компьютеринга имеют выраженную адресность и предназначены для

решения вполне определенных задач, например, имеющих повышенную вычислительную сложность или касающихся конкретного применения. Хотя практическое и повседневное применение большинства из них только впереди, ожидая появления подходящих устройств, их модельная оригинальность увлекает своей естественной фундаментальностью, инновационностью и потенциалом. В последнем случае открывается возможность создать основу для новых форм обработки информации и, базируясь на них, развить семейства приложений. Обсуждение их технологических возможностей происходит обычно вне сферы периодической литературы по компьютерным наукам, а исследование перемещено в область таких естественных наук, как физика или химия. Это обусловлено состоянием работ, которые пока еще находятся на уровне проработки основополагающих идей – либо в самой начальной технологической фазы, либо на уровне экспериментов. По мере технологического вызревания подобные формы компьютеринга попадают и в поле зрения компьютерных наук, начиная использоваться на практике. Осуществление новых форм компьютеринга наталкивается на трудности разработки аппаратных средств, требуя выработки новых архитектур. С другой стороны, трудности вызывают и попытки оснастить новые формы компьютеринга подходящим программным обеспечением, поскольку требуется разрабатывать новые схемы организации вычислений и новые алгоритмы. Ситуация похожа на ту, которая была при переходе от последовательных алгоритмов к алгоритмам для параллельных вычислительных систем. Некоторые известные решения могут быть перенесены на новую область, а иные – нет.

Известные новые формы можно подразделить на две большие группы. В первую из них попадают решения, основанные на *нанотехнологиях*. К ним относятся организация схем вычислений на нановолокнах, угольных нанотрубках, органических молекулах, био-ДНК и квантовых эффектах. Во вторую – специальные формы компьютеринга, включающие оптические, мик-ро/наножидкостные и хаотические вычисления.

3 Пересмотр основ компьютеринга

В сложившемся компьютеринге продвинулись в понимании множеств и научились эксплуатировать модели вычислений, основанные на представлении о переменной, для которой известно, по какой области она будет пробегать – *типовые* модели вычислений, или модели вычислений с типами. Другими словами, идея *типа* получила широкое распространение и повсеместное признание, а все имеющиеся системы программирования имеют в большей или меньшей степени проработанные системы управления типами переменных/объектов.

Менее проработанными оказались модели, основанные на классах, поскольку они ведут к построению областей, элементами которых являются другие области и т.д., а для таких структур резко возрастает объем вычислений, требуемый для проверки истинности или ложности утверждений.

Совсем не проработанными и не осмысленными на практике остались модели вычислений, в которых не предполагалось указания типа переменных – *бестиповые* модели вычислений. Среди них доминирующее положение занимает λ -исчисление и комбинаторная логика. Хотя λ -исчисление и получает признание в практике технологий программирования, но не так быстро, как оно того заслуживает, комбинаторная логика применяется явно недостаточно. Комбинаторы – основные первоэлементы комбинаторной логики, – вводились в надежде избавиться от арифметического стиля работы с числовыми данными, характерного для абсолютного большинства прежних и существующих систем программирования, и вместо этого перейти к иному стилю рассуждений в терминах объектов и применений их друг к другу. При использовании первого стиля выполняется *вычисление* – от слова ‘число’, – а при использовании второго – *компьютинг* в собственном смысле этого термина. Кроме того, комбинаторы имеют дело со *свободными* переменными, которые понимаются как индетерминанты и вовсе не имеют дела со *связанными* переменными. По сути, компьютеринг с комбинаторами выполняется в терминах констант. Еще лучше сказать: ‘константных объектов’, а это константы не в абсолютном понимании, а в *относительном*, когда объекты проявляют *свойство константности* относительно *среды*, в которой осуществляется компьютеринг. Остается сформулировать подходящее определение константы – дать характеристику свойству константности, – и также определиться с моделью среды. Это как раз и оказывается непростым делом, поскольку влияет на все элементы вычислительной архитектуры без исключения.

Теперь подходим к важному моменту – необходимости сформулировать *представление о константе*, которое оказалось бы плодотворным для компьютеринга в целом. Конечно, хотелось бы оставить его и интуитивно прозрачным и согласованным с имеющимся естественнонаучным представлением о константе.

4 Представление о константе

Много или мало мы знаем о том, что такое “константа”? По-существу, все, связанное с представлением о константе оказалось передоверенным математике. А в ней как представление о константе, так и представление о переменной считаются самоочевидными. В лучшем случае признается, что константа – это то, что – в отличие от переменной, не меняется, а переменная, в частном случае, может быть константой. Эти представления до поры оставались незыблемыми или казались таковыми. Так было бы и сейчас, если бы не компьютеринг.

Оказалось, что самое лучшее из того, что было предложено – это всеобщее соглашение о константе, другой стороной которой была переменная. При обсуждении речь обычно ведется о величинах, за которыми тут же усматриваются числа. Числа и работа с ними – вычисления, – длительное время прекрасно и безотказно работали для нашей пользы. Даже в компьютерах, пока их производительность не выросла настолько, что технология подошла к

физическому порогу миниатюризации, позволяющему точно фиксировать и различать нули и единицы. И вот теперь наступает некоторая переломная точка в понимании, что же такое компьютеринг.

Есть ли теория констант? Ответ на этот вопрос известен, даже больше: оказывается, что *нет* языка, на котором можно было бы математически точно говорить о константах. Попытки говорить о них в абсолютном смысле выглядят ограниченно и недостаточно обоснованно. В *относительном* смысле можно добиться большего: почему бы не назвать константой ‘объект’, который не зависит от определенного ‘контекста’? Или, еще лучше, не зависит от ‘среды’, которая пока рассматривается как некоторый контекст. Посмотрим, есть ли адекватный язык для точного выражения этой мысли.

Для пробы возьмем Л-выражение

$$(\lambda x. \text{объект})(\text{среда}) = \text{объект},$$

которое, очевидно, выражает тезис константности: $(\lambda x. \text{объект})$ может рассматриваться как одноместная константная функция. Но переход к Л-языку потребовал введения в рассмотрение представление о ‘переменной’, которая еще и оказывается связанной оператором абстракции! В этом моменте получаем всю мощь обычной математики, но наследуем и все известные издержки оперирования переменными.

Возьмем выражение комбинаторной логики

$$K(\text{объект})(\text{среда}) = \text{объект},$$

с которым надо действовать по схеме аксиом $K : KaB = a$. Сказать на языке комбинаторной логики, что объект - константный, то есть представляет собой константу относительно среды, значит, применить к нему комбинатор K . Тогда - относительно заданной среды, а это, возможно, иной объект, - ‘объект’ попросту цитируется, что сейчас нас вполне устраивает. Как математически точно сказать, что объект меняется? Для этого запишем

$$V(\text{объект})(\text{среда}) = \text{объект}',$$

где V - некоторый комбинатор, который действует на интересующий нас объект - старый объект, - относительно заданного контекста и вырабатывает новый объект - объект'.

И опять в Л-языке для записи этой идеи придется привнести традиционную математику.

Пишем

$$V(\lambda x. \text{объект})(\text{среда})(\text{объект-1}) = V(\text{объект})(\text{среда})_{x:=\text{объект-1}},$$

то есть, старая среда оказалась замененной на новую $(\text{среда})_{x:=\text{объект-1}}$, отличающуюся от старой разве что тем, что вместо переменной x выполнена подстановка объекта ‘объект-1’. Появилось выражение с переменной, обслуживание которой потребует введения правила подстановки со всеми известными последствиями. Таким образом, аппликативный по своему замыслу А-язык оказывается обремененным всеми известными техническими трудностями работы со свободными и связанными переменными. В языке комбинаторной логики такого бремени все еще нет.

5 Функции

Обычная идея, касающаяся функций, состоит в том, что они являются специальным случаем закона соответствия, ставящего в заранее определенное отношение элементам одной области элементы из другой области. Так что прежде всего придется определиться с областью A , которую считаем представленной константным объектом, то есть в его состав заведомо не входят свободные переменные. Для области A и комбинатора V потребуем

$$A = A \circ A \equiv \mathbf{V}AA \equiv 1_A.$$

Для отображения– объект, в состав которого не входят свободные переменные, Потребуем

$$f : A \rightarrow B, \text{ где } f$$

$$f = B \circ f \circ A \equiv B \circ (\mathbf{V}fA) \equiv \mathbf{V}B(\mathbf{V}fA).$$

Таким образом, отображения f конструируются как тройки (A, f, B) .

6 Взаимодействие объекта и среды

6.1 Среда

Очевидным представляется тезис, что для взаимодействия объектов нужен посредник – *среда*. По-крайней мере, сейчас он не подвергается сомнению. Точнее говоря, для того, чтобы объект взаимодействовал с другими объектами нужна структура, где объекты локализованы.

Противоположный случай – когда одни “блуждающие” объекты “встречаются” с другими блуждающими

объектами, – интересен, но его обсуждение пока отложим. Область программирования дает случай, когда объекты тем или иным образом уже упакованы в среде. При этом исходным разрабатываемым понятием является именно среда, под которой понимается среда вычислений. Среда оснащается системой программирования, но не наоборот. Другое обстоятельство заключается в том, что объект взаимодействует не со всей средой сразу, а с ее частью – той, которая окажется “в сфере действия” объекта.

Предструктура Для упаковки объектов используется аппликативная предструктура. В ней проявляют себя два аспекта объекта – редекс и контракт. Другими словами, предструктура дает представление вычисления как в терминах редукции – перехода от редекса к контракту, – так и в терминах экспансии – перехода от контракта к редексу.

Принцип взаимодействия отличается *несимметричностью*: есть объект, который оказывает свое действие и есть объект, на который действие оказывается. Воздействие одного объекта на другой имеет ступенчатый характер: оно осуществляется, если и только если объекты расположены *рядом*. Расположение бывает двух видов: рядом и нерядом, а во втором случае взаимодействия объектов не происходит. В случае расположения рядом объекты немедленно вступают во взаимодействие. В результате взаимодействия возникает и начинает свое существование новый объект – результат апплицирования, или применения первого объекта ко второму. Теперь, если найдется объект, расположенный рядом с ним, то начинается новый акт взаимодействия, в котором различаются два случая. В первом из них вновь сгенерированный объект воздействует на уже существующий, который оказался рядом и на который воздействие и оказывается.

Во втором случае вновь сгенерированный объект попадает в сферу действия имеющегося объекта, который и оказывает воздействие на этот последний объект.

В любом из этих случаев возникает и начинает свое существование новый объект, рассматриваемый как результат такого несимметричного взаимодействия двух объектов-родителей. Он поселяется в предструктуре на равных правах с иными объектами. Это, в частности, означает следующее: как только новый объект-результат сформирован, можно говорить о новом акте взаимодействия.

Итак, обитатели предструктуры участвуют в процессе взаимодействия, который развивается по принципу домино. Важным оказывается следующее обстоятельство: либо имеются *исходные* атомарные объекты, либо имеются *производные* неатомарные объекты, у каждого из которых имеется в точности два предка-родителя. Остается открытым вопрос, откуда берутся исходные объекты, но его обсуждение пока отложим.

6.2 Взаимодействие

Объект может себя проявить во взаимодействии с другими объектами, если он участвует в аппликации. Тогда он может проявить арность, равную 0 (константный объект) или отличную от нуля. Для простоты рассмотрим случай, когда объект проявляет арность, равную 1 (одноместная функция).

Поскольку взаимодействие осуществляется через посредника - среду, - то потребуются мета-операторы. Пока ограничимся двумя метаоператорами: A - каррирование и $\| \cdot \|$ - оценивающее отображение. Для произвольного объекта M проверим, сможет ли он проявить арность 1 в среде z . Для этого понадобится записать

$$\|M\|i d_0,$$

что представляет собой значение объекта M в среде z . Если значение объекта M проявляет арность 1, то появляется конструкция значения объекта в среде

$$A\|M'\|i d_0,$$

где M' совпадает с объектом M всюду, кроме переменной, которой надо было присвоить значение do : вместо этой переменной записывается число Де Брейна 0 - прообраз указателя на do в среде z' . Среда z' совпадает со средой z всюду, за исключением образа этой подстановочной переменной, которой теперь присвоено значение do :

$$\|M'\|[i, d_0].$$

На самом деле понадобился один скомплексированный метаоператор

$$A\| \cdot \| : \text{объект} \times \text{среда} \rightarrow \text{значение},$$

который является объектообразующим, порождая функцию арности 1. Действительно,

$$A\|M'\|i d_0 = \|M'\| \underbrace{[i, d_0]}_{\equiv i'}.$$

Например, если M - тождественное преобразование / с характеристикой $Ido = do$, то достаточно положить, что M' является подстановочной переменной, которой присваивается значение do в среде i :

$$\begin{aligned}
\|I\|id_0 &\equiv \Lambda\|\bar{0}\|id_0 \\
&= \|\bar{0}\| \underbrace{[i, d_0]}_{\equiv i'} \\
&= Snd[i, d_0] = d_0,
\end{aligned}$$

что и ожидалось. Здесь $\Lambda\|M'\|id_0$, - образ объекта I , полученный в результате его взаимодействия со средой. Это должен быть попросту указатель Snd на d_0 , расположенный в модифицированной среде.

Другой пример. Если M - канцелятор K с характеристикой $Kd\bar{d}o = d\bar{d}$, то достаточно положить, что M' является подстановочной переменной, которой присваивается значение $d\bar{d}$ в среде i :

$$\begin{aligned} \|K\|id_1d_0 &\equiv \Lambda(\Lambda\|\bar{1}\|)i d_1d_0 \\ &= \Lambda\|\bar{1}\| \underbrace{[i, d_1]}_{i'} d_0 \\ &= \|\bar{1}\| \underbrace{[[i, d_1], d_0]}_{i''} \\ &= (Snd \circ Fst)i'' \\ &= Snd i' = d_1, \end{aligned}$$

что соответствует характеристике.

И еще один пример. Если M - распределитель S с характеристикой $Sd_2d_1d_0 = d_2d_0(d_1d_0)$, то

$$\begin{aligned} \|S\|id_2d_1d_0 &\equiv \Lambda(\Lambda(\Lambda\|\bar{2}\bar{0}(\bar{1}\bar{0})\|))id_2d_1d_0 \\ &= \Lambda(\Lambda\|\bar{2}\bar{0}(\bar{1}\bar{0})\|) \underbrace{[i, d_2]}_{i'} d_1d_0 \\ &= \Lambda\|\bar{2}\bar{0}(\bar{1}\bar{0})\| \underbrace{[[i, d_2], d_1]}_{i''} d_0 \\ &= \|\bar{2}\bar{0}(\bar{1}\bar{0})\| \underbrace{[[[i, d_2], d_1], d_0]}_{i'''} \\ &= \|\bar{2}\|i'''(\|\bar{0}\|i''')(\|\bar{1}\|i'''(\|\bar{0}\|i''')) \\ &= Snd \circ Fst \circ Fst i'''(Snd i''')(Snd \circ Fst i''')(Snd i''') \\ &= Snd \circ Fst i''d_0(Snd i''d_0) \\ &= Snd i'd_0(d_1d_0) = d_2d_0(d_1d_0), \end{aligned}$$

что и ожидалось.

7 Принципы компьютеринга

Формулируя принципы компьютеринга, приходится принимать допущения, причем, как оказывается, некоторые фундаментальные послышки, возможно, в силу их кажущейся простоты и обманчивой самоочевидности, ускользают от внимания исследователей.

Одно из них и, возможно, самое основное, состоит в принятии аппликативной структуры как среды осуществления компьютеринга. Это значит, что одни сущности/объекты применяются, или *аплицируются* к другим сущностям/объектам. Например, объект-функция *применяется* к объекту-аргументу, а результат этого применения рассматривается как *значение* данной функции на данном аргументе. Все это выглядит вполне очевидным, но почти никогда не формулируется явно, что приводит к различным смещениям акцентов.

Другое – и опять всем известное, – допущение может быть сведено к простой формулировке оперирования с идентификаторами: берется то, что считается *идентификатором*, и для него *относительно среды строится* то, что будет считаться его *значением*. Вычислением считается именно этот процесс построения, а сам компьютеринг разрабатывает технологии осуществления построения. Итак, отношение между идентификатором и его значением параметризовано средой.

В среде не все объекты изолированы, наибольший интерес представляет именно взаимодействие объектов, но оно проявляется себя через аппликацию. Это структурная метаоперация, которая управляет взаимодействием объектов и было бы нежелательно, чтобы в процессе выполнения вычисления значения идентификатора она меняла свои свойства. Значит, аплицирование будет рассматриваться как инвариант относительно вычисления значения, а это свойство требует явной характеристики принятием общего принципа вычисления значения аппликации (В. Э. Вольфенгаген, [22]).

Принцип вычисления значения аппликации Основная послышка состоит в следующем: значение аппликации равно аппликации значений.

Данная формулировка нуждается в уточнении, поскольку вычисление значения можно производить как при заранее зафиксированной среде, так и в случае, когда среда не фиксируется. В первом случае для фиксированной среды i примем:

$$\|MN\|i = (\|M\|i)(\|N\|i)$$

для произвольных объектов M, N . Тогда по чисто формальным причинам, вытекающим из общих свойств

вычислений,

$$\begin{aligned} (\|M\|i)(\|N\|i) &= (\lambda r.r\|M\| \|N\|)Si \\ &= CIS(\lambda r.r\|M\| \|N\|) \\ &\equiv \mathcal{S}[\|M\|, \|N\|]i \end{aligned}$$

для $S \equiv \lambda xyz.xz(yz)$, $C \equiv \lambda xyz.xzy$, $I \equiv \lambda x.x$, $\mathcal{S} = CIS$ и упорядоченной пары $[x, y] \equiv \lambda r.rxy$.
Следовательно, сформулируем правило

(правило 1) $\|MN\| = \mathcal{S}[\|M\|, \|N\|]$

которое выводимо в $\lambda\eta\xi$ -исчислении.

Принцип вычисления значения упорядоченной пары Основную посылку сформулируем в виде равенства значение упорядоченной пары равно упорядоченной паре значений.
Формально это равенство перепишем в виде

$$\|[M, N]\|i = [\|M\|i, \|N\|i]$$

для произвольных термов M, N и соотношения \mathcal{Z} . С использованием постулатов Л-исчисления, из исходного
(η), (ξ) и (τ)

принципа выводимо правило:

(правило 2) $\|[M, N]\| = \langle \|M\|, \|N\| \rangle$,

где $\langle f, g \rangle \equiv \lambda t.[ft, gt]$ для произвольных f, g . Оба сформулированных принципа и выведенные из них правила позволяют получить и обосновать стандартные семантические средства вычислительных моделей. В качестве наиболее важных для средств концептуализации укажем два следствия, связанные с вычислением значений А-выражений и аппликаций.

Вычисление значения А-выражения Рассмотрим аппликацию $(\lambda.M)\bar{d}$, где M, \bar{d} - произвольные термы, а $(\lambda.M)$ обозначает абстракцию по (некоторой) переменной. Тогда справедлива цепочка равенств:

$$\begin{aligned} \|(\lambda.M)\bar{d}\|i &= (\|(\lambda.M)\|i)(\|\bar{d}\|i) \quad (\text{по принципу 1}) \\ &= (\|\lambda.M\|i)di \quad (\text{полагаем } (\|\bar{d}\|i) = d) \\ &= A\|M\|id \quad (\text{по определению } A) \\ &= \|M\|[i, d] \quad (\text{в силу } Ah = \lambda xy.h[x, y]). \end{aligned}$$

Следовательно, справедливо правило:

(правило 3) $\|(\lambda.M)\bar{d}\|i = \|M\|[i, d]$.

В дальнейшем иногда будем считать, что это правило определяет производящую функцию: вычисление значения M “производит” подстановки d , удовлетворяющие M .

Второй способ вычисления значения аппликации При вычислении значения аппликации воспользуемся определением аппликатора ε :

$$\begin{aligned} \|MN\|i &= (\|M\|i)(\|N\|i) \quad (\text{в силу принципа 1}) \\ &= \varepsilon[\|M\|i, \|N\|i] \quad (\text{по определению } \varepsilon) \\ &= (\varepsilon \circ \langle \|M\|, \|N\| \rangle)i \quad (\text{по определению } \langle \cdot, \cdot \rangle) \end{aligned}$$

Из этой цепочки равенств выводимо правило

(правило 4) $\|MN\| = \varepsilon \circ \langle \|M\|, \|N\| \rangle$.

Наконец, выделим случай вычисления значения индивидуальных констант:

$$\|c\|i = c,$$

откуда выводимо (в $\lambda\eta\xi$ -исчислении) правило

(правило 5) $\|c\| = \lambda i.c$.

Список правил Для справки представим полный список полученных правил:

Индивидуальные константы не зависят от конкретного соотношения, то есть ведут себя статично.

(правило 1)	$\ MN\ = \mathcal{S}[\ M\ , \ N\]$
(правило 2)	$\ [M, N]\ = \langle \ M\ , \ N\ \rangle$
(правило 3)	$\ (\lambda.M)d\ i = \ M\ [i, d]$
(правило 4)	$\ MN\ = \varepsilon \circ \langle \ M\ , \ N\ \rangle$
(правило 5)	$\ c\ = \lambda i.c.$

8 Взаимодействие объектов

8.1 Нефиксированное число аргументных мест

Прежде всего хотелось бы не связывать взаимодействие объектов с традиционными математическими представлениями. По крайней мере, не делать это сразу и безо всякой видимой необходимости. Математическая интуиция подсказывает, что если одно действует на другое, то первое надо считать функцией, а второе – аргументом. В этих условиях результат взаимодействия рассматривается как значение, которое, в свою очередь, является объектом.

Но если есть функция, то она характеризуется арностью – числом ее аргументов или, точнее, аргументных мест. В обычной математике число аргументов у функции известно заранее, но в случае воздействия одного объекта на другие заранее неизвестно, на сколько именно объектов он сможет оказать воздействие. Если уж объект считать функцией, то число ее аргументных мест оказывается заранее не фиксированным, а свою арность объект проявляет именно во взаимодействиях. Известны ли такие математические функции? Как оказываются ими оказываются только комбинаторы.



Рис. 1. Объект X действует на объекты a и b, где a не действует на b.

8.2 Объект и его сфера действия

Пусть имеется некоторый набор объектов, расположенных в виде структуры. Начнем с того, что не всякий объект будет взаимодействовать со всяким. Если говорим, что объект a действует на объект b, имеем ввиду, что a применяется, или апплицируется к объекту b. Из рис. 1 видно, что в сфере действия объекта X находятся и объект a, и объект b, но a не действует на b. Математически записываем:

$$X..ab.. \equiv (..(((X..)a)b)..).$$

Из рис. 2 видно, что в сфере действия объекта X находятся объект a, который действует на

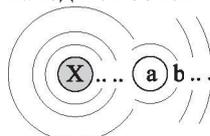


Рис. 2. Объект X действует на объект (ab), где a действует на b, так что X действует на результат действия объекта a на объект b.

Математически записываем:

$$X..(ab).. \equiv (..((X..)(ab))..).$$

Скобки, в которые заключены объекты a и b, являются *существенными*, и их нельзя опустить.

9 Система исходных объектов

Возможно, придется сделать некоторые предположения. Они будут относиться к наличию некоторого набора первичных объектов – ведь нужно же иметь в наличии актуальные, действительно существующие объекты. Вместе с тем предпримем попытку вести обсуждение отношений между объектами, по возможности, без тех допущений, в особенности, касающихся существования объектов, без которых можно обойтись.

Во-первых, понадобится возможность *порождения* произвольного объекта b. Пусть это будет всегда сопровождаться возникновением и применением константного объекта K. Иная версия этого рассуждения может выглядеть иначе: хотелось сформулировать утверждение, что некоторый объект a не зависит от среды b. Это означает также, что чисто синтаксически объект K берет на себя функцию *цитирования* a в среде или контексте b. Применение K также означает *инкапсуляцию* объекта a средой b. Каждую из этих объяснительных систем можно использовать в

зависимости от контекста, в котором ведется исследование объектов.

С другой стороны, возникает и симметричная возможность *устранения*, или *прекращения существования* – канцеляции, – произвольного объекта b . Прекращение существования объекта b вызывается прекращением существования и константного объекта K , непосредственно действующего на некоторый объект a , причем объект a остается и продолжает свое существование.

Из рис. 3 можно уяснить поведение объекта-канцелятора K . При экспансии произвольного объекта a порождается K , начиная свое существование, а объект a оказывается непосредственно в сфере его влияния. Кроме того, порождается объект b , который попадает в сферу влияния результата взаимодействия объекта K с объектом a . При редукции канцелятор K устраняет объект b , который прекращает свое существование, но и этот экземпляр K также больше не существует.

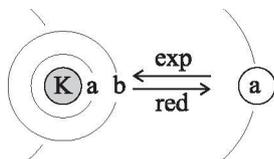


Рис. 3. Характеристика объекта K : $a = Kab$.

Вместе с тем может потребоваться *устранять клоны* объекта, оставляя только его единственный экземпляр. Конечно, при этом ожидается и симметричная операция распределения действий на различные экземпляры произвольного объекта c , выполняющая их *клонирование*. Попробуем это осуществить следующим образом: при устранении клона объекта c будет порождаться константный объект S специального вида, а при клонировании c – наоборот, один из экземпляров объекта S будет вынужден прекратить свое существование.

Все это означает, что различные экземпляры объекта в аппликативной среде будут неразличимы. Эта идея неразличимости клона объекта присутствует в схеме вычисления, определяющей поведение распределителя S : $ac(bc) = Sabc$. Как видно из рис. 4, объект S – в аппликативной среде, – *применяется* к объектам a , b и c , удерживая их. Это и означает, что арность исходного примитивного объекта-комбинатора S равна 3-м. На остальных объектах в среде его влияние непосредственно не распространяется. Но вся конструкция $Sabc$ *редуцируется* к $ac(bc)$, объект c клонируется, а вычисление распределяется: один экземпляр c непосредственно оказывается в сфере апплицирования a , а его второй экземпляр – сфере b .

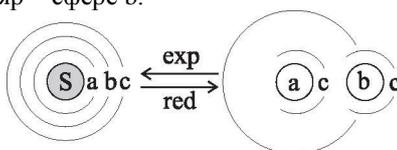


Рис. 4. Характеристика объекта S : $ac(bc) = Sabc$.

Вычисление распределяется, но результат, полученный при взаимодействии b с c , сам оказывается в сфере действия результата, полученного при взаимодействии a с c . Самое существенное, что при порождение нового объекта S : $ac(bc) = Sabc$ клон объекта c прекращает свое существование, а объект S свое существование начинает, становясь обитателем аппликативной среды. В этом суть *S-экспансии*. С другой стороны, при воздействии объекта S на удерживаемые им в среде объекты a , b и c происходит клонирование c , этот клон начинает свое существование в среде, но в то же самое время S прекращает свое существование. Это характеризует *S-редукцию*.

10 Система производных объектов

Теперь путь “обнаружения” в аппликативной среде объектов с заданными характеристиками уже, в общих чертах, ясен.

Процесс обнаружения нового объекта оказывается весьма конструктивным: приходится предъявлять конструкцию из уже обнаруженных, старых объектов. Таким образом, новые объекты оказываются *производными*, в то время, как постепенно складывается представление об *исходных* объектах, все или часть которых оказываются *первичными*. Рассмотрим на примере, как синтезируется новый объект с заданной характеристикой. Пусть на рис. 5 приведена характеристика такого объекта: это комбинатор-пермутатор C , выполняющий перестановку местами объектов b и c .

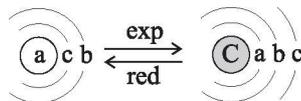


Рис. 5. Синтез объекта C с заданной комбинаторной характеристикой: $Cabc = acb$.

Синтез объекта C выполняем следующим образом. Начинаем с объекта acb , который по структуре является результатом применения к b результата применения a к c .

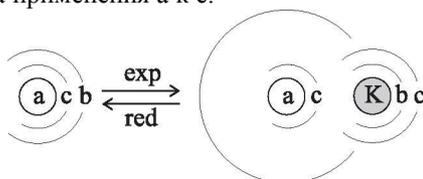


Рис. 6. K -экспансия и порождение второго экземпляра объекта c : $acb = ac(Kbc)$.

S -экспансию на основе объекта $ac(Kbc)$ в соответствии с рис. 7. При ее выполнении второй Во-первых, выполним клонирование c . Для этого произведем K -экспансию на основе объекта b , порождая второй экземпляр объекта c , что можно увидеть на рис. 6. Во-вторых, выполним

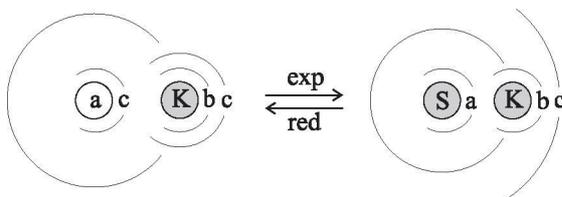


Рис. 7. S -экспансия и устранение второго экземпляра объекта c с перестановкой объекта Kb : $ac(Kbc) = Sa(Kb)c$. экземпляр объекта c устраняется, но генерируется распределитель S .

В-третьих, выполним V -экспансию в соответствии с рис. 8.

В-четвертых, выполним V -экспансию, которая устраняет композицию объектов V и S . В то же самое время выполним клонирование объекта a , пользуясь K -экспансией на основе объекта K . Порождаются новые экземпляры объектов K и a , начиная свое существование в среде. Эта трансформация выполняется в соответствии с рис. 9.

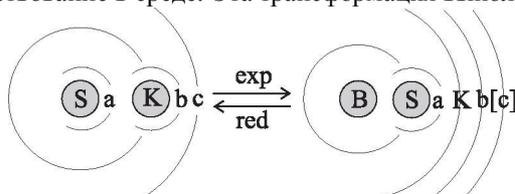


Рис. 8. V -экспансия и устранение композиции объектов Sa и K : $Sa(Kb)c = V(Sa)Kbc$.

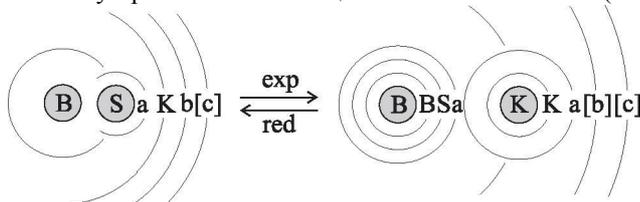


Рис. 9. V -экспансия и устранение композиции объектов V и S , выполняемая совместно с K -экспансией на основе K с порождением клона объекта a : $V(Sa)Kbc = BBSa(KKa)bc$.

И, наконец, в-пятых, выполним S -экспансию, которая устраняет распределение вычислений $BBSa$ и KKa вместе с клоном объекта a . Второй экземпляр объекта a прекращает свое существование, а объект S – начинает свое существование в среде. Это отражено на рис. 10. Теперь в среде достигнут желаемый порядок следования объектов a , b и c , то есть объекты c и b поменялись места.

В этом процессе использован комбинатор V с характеристикой $Vabc = a(bc)$. Представляется очевидным, что его можно также синтезировать, пользуясь исключительно комбинаторами K и S . На рис. 11 приведена характеристика

комбинатора В.

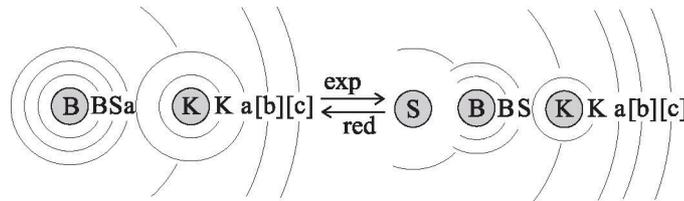


Рис. 10. S-экспансия и устранение клона объекта a: $BBSa(KKa)bc = S(BBS)(KK)abc$.

Покажем, что комбинатор В может быть получен комбинированием К и S. Прежде всего зафиксируем объект $a(bc)$. Идея состоит в том, что нужно освободить объект с от непосредственного влияния объекта b. Математически это означает, что потребуются раскрыть скобки. Для этого синтезируем распределение вычислений, породив еще один экземпляр объекта с, что достигается появлением экземпляра комбинатора К. Символически записываем:

$$a(bc) = Kac(bc).$$

Далее, элиминируем один из экземпляров объекта с, что требует появления экземпляра комбинатора S, но попутно оставшийся экземпляр с выводится из-под зависимости от b:

$$Kac(bc) = S(Ka)bc.$$

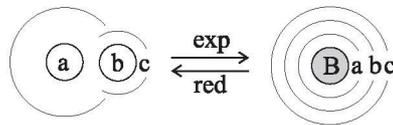


Рис. 11. Характеристика комбинатора В.

Аналогичным приемом выведем теперь объект a из-под зависимости от объекта К. Это придется делать в два приема. Сначала породим второй экземпляр a, распределив вычисление и сгенерировав экземпляр комбинатора К:

$$S(Ka)bc = K Sa(Ka)bc.$$

На рис. 12 показана вся цепочка вывода

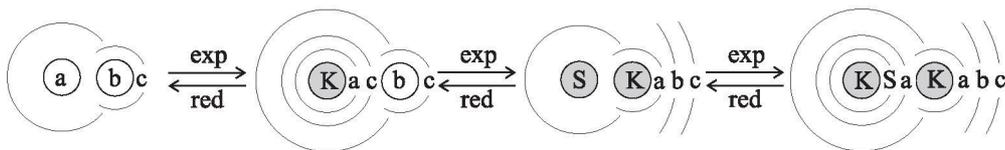


Рис. 12. Вывод комбинатора В: $a(bc) \triangleleft Kac(bc) \triangleleft S(Ka)bc \triangleleft K Sa(Ka)bc$.

$$a(bc) \triangleleft Kac(bc) \triangleleft S(Ka)bc \triangleleft K Sa(Ka)bc.$$

Теперь будем устранять один из двух экземпляров объекта a, для чего придется сгенерировать объект S:

Целевой объект синтезирован, остается только положить

$$S(KS)Kabc \equiv Babc.$$

На рис. 13 показано продолжение вывода, начиная с $KSa(Ka)bc$:

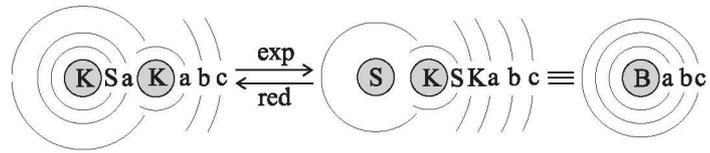


Рис.13. Вывод комбинатора B : $KSa(Ka)bc \xrightarrow[\text{red}]{\text{exp}} S(KS)Kabc = Babc$.

$$K Sa(Ka)bc \triangleleft \underbrace{S(KS)K abc}_{\equiv B} \equiv Babc.$$

Таким образом, полная цепочка вывода имеет вид

$$a(bc) \triangleleft Kac(bc) \triangleleft S(Ka)bc \triangleleft K Sa(Ka)bc \triangleleft \underbrace{S(KS)K abc}_{\equiv B} \equiv Babc.$$

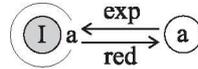


Рис. 14. Характеристика комбинатора I : $Ia = a$.

Точно также можно синтезировать тождественный комбинатор I с характеристикой $Ia = a$, что показано на рис. 14. Это – особый комбинатор, под его воздействие произвольный объект a не меняется, а точнее говоря, остается самождественным, переходит сам в себя.

Вывод для комбинатора I приведен на рис. 15. Начинаем с того, что фиксируем объект a .

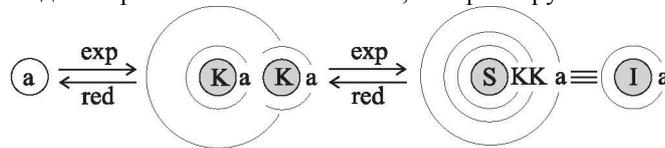


Рис. 15. Вывод комбинатора I : $a = Ka(Ka) = SKKa \equiv Ia$.

Порождаем объект (Ka) , но при этом возникает также и экземпляр объекта K , начиная свое существование. Теперь оказалось, что имеется два экземпляра объекта a , структурное расположение которых позволяет один из них элиминировать. Но при этом порождается объект S , который начинает свое существование. Теперь оказалось, что объект SKK по своей характеристике ничем не отличается от требуемого при синтезе объекта I . Так что цель достигнута, а математически это можно записать посредством $I \equiv SKK$.

Исходные и полученные в ходе формирования объектов, некоторые из производных комбинаторов сведены в таблицу 1.

11 Вывод комбинаторов

Располагая подобным атомно-молекулярным конструктором, можно синтезировать объекты с заранее заданными вычислительными свойствами – комбинаторными характеристиками.

На рис. 16 в схеме для комбинатора S объект b замещен на комбинатор I . Это запускает редукцию в направлении от $Sale$ к $ac(Ic)$. В последнем объекте содержится редекс Ic , который замещается на контракт c , так что вся редукция работает следующим образом:

$$Sale > ac(Ic) > ace$$

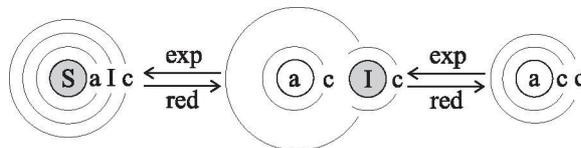


Рис. 16. Частная конструкция комбинатора S для $b = I$.

Таблица 1. Исходные и производные комбинаторы.

Комбинаторная характеристика	Взаимодействие объектов
$Kab = a$	
$Sabc = ac(bc)$	
$Ia = a$	
$Wab = abb$	
$Cabc = acb$	
$Babc = a(bc)$	
$\Psi abcd = a(bc)(bd)$	
$\Phi abcd = a(bd)(cd)$	

Но внутри объекта $Salc$ необходимо провести преобразования: это касается позиции комбинатора I . Преобразование будем вести экспансией:

$$Salc \triangleleft \underbrace{CSI}_{\equiv W} ac \equiv Wac,$$

а обе цепочки синтеза объекта можно слить воедино:

$$Wac \equiv CSIac \triangleright Salc \triangleright ac(Ic) \triangleright acc.$$

Процесс синтеза объекта W представлен на рис. 17. Теперь к уже имеющимся комбинаторам можно добавить вновь синтезированный комбинатор W , который, как оказалось, ведет себя в точности, как CSI .

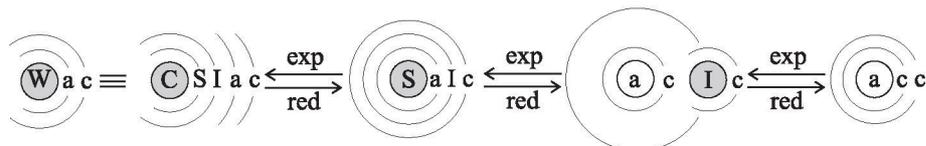


Рис. 17. Синтез конструкции комбинатора W с характеристикой $Wac = acc$.

На рис. 18 приведена характеристика комбинатора W .

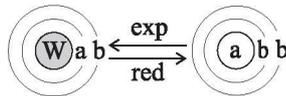


Рис. 18. Характеристика комбинатора W.

12 Редукция и экспансия объектов

Еще раз вернемся к особенностям получения комбинаторного представления объекта-пермутатора C. На рис. 19 приведены вспомогательные объекты, которые получают в ходе синтеза целевого

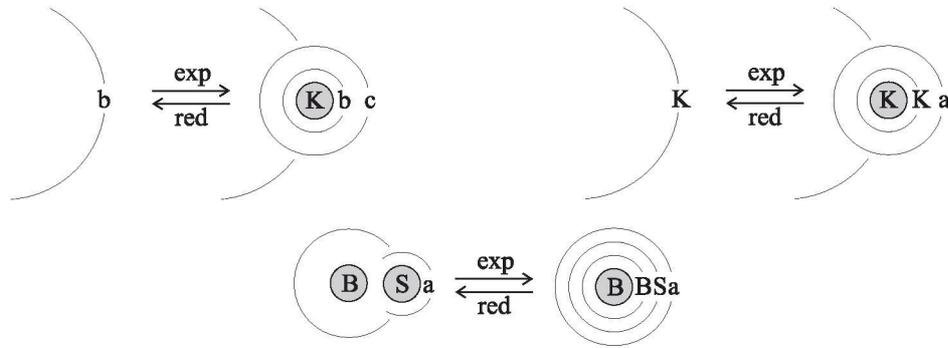


Рис. 19. Синтез вспомогательных объектов: $b = Kbc$, $K = KKa$, $B(Sa) = BBSa$.

объекта C с заданной комбинаторной характеристикой

$$Cabc = acb.$$

Все, что требуется – это поменять местами второй и третий объекты в acb . Однако, надо помнить что перестановку приходится производить в аппликативной среде. Первая из возникающих идей состоит в том, чтобы попробовать найти преобразование acb , выполнив которое можно затем применить преобразование по схеме S. На рис. 6 такая попытка отражена и потребовалось сгенерировать еще один экземпляр объекта c – его *клон*, но не копию. Другими словами, различные экземпляры объекта в аппликативной среде будут неразличимы. Эта идея неразличимости клона объекта присутствует в схеме вычисления, определяющей поведение распределителя S: $ac(bc) = Sabc$. Как видно из рис. 4, объект S – в аппликативной среде, – применяется к объектам a, b и c, удерживая их.

13 Синтез объекта с заданной комбинаторной характеристикой

Вернемся к рассмотрению рис. 5.

Понадобится клонировать объект c, а затем его элиминировать, воспользовавшись S-экспансией. Во-первых, выполним клонирование c. Для этого произведем K-экспансию на основе объекта b, порождая второй экземпляр объекта c, что можно увидеть на рис. 6.

Во-вторых, выполним S-экспансию в соответствии с рис. 7.

В-третьих, выполним B-экспансию в соответствии с рис. 8.

В-четвертых, выполним B-экспансию, которая устраняет композицию объектов B и S. В то же самое время выполним клонирование объекта a, пользуясь K-экспансией на основе объекта K. Порождаются новые экземпляры объектов K и a, начиная свое существование в среде. Эта трансформация выполняется в соответствии с рис. 9.

И, наконец, в-пятых, выполним S-экспансию, которая устраняет распределение вычислений BBSa и KKa вместе с клоном объекта a. Второй экземпляр объекта a прекращает свое существование, а объект S – начинает свое существование в среде. Это отражено на рис. 10.

Теперь в среде достигнут желаемый порядок следования объектов a, b и c, то есть объекты c и b поменялись места. Производный объект, выполняющий такую перестановку, назовем комбинатором-пермутатором и обозначим его через C. Из рис. 10 следует, что $C \equiv S(BBS)(KK)$. Как оказалось, C существует, а его характеристика представлена на рис. 5.

14 Бесконечные конструкции

Объекты могут приводить к бесконечным конструкциям, а их взаимодействие носит цепной характер. На рис. 20 приведена комбинаторная характеристика – то характеристическое равенство,

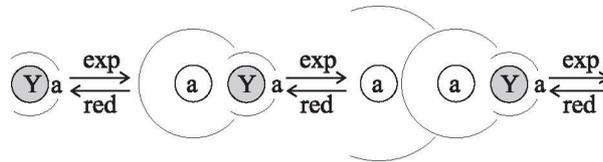


Рис. 20. Синтез для объекта a неподвижной точки: $Y a = a(Y a) = a(a(Y a)) = \dots$ которое надо добавить к структуре комбинаторов, чтобы дать право на существование комбинатору Y :

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

a это парадоксальный комбинатор Х.Б. Карри (¹ Как известно, объект Y имеет комбинаторное представление: $Y = WS(BWB)$). Комбинатор Y оживляет довольно однородную структуру объектов и комбинаторов, открывая возможность организации нетривиальных циклических вычислений.

15 Множественность миров комбинаторов

Попытки ответа на вопрос, чем же – на самом деле, – являются *константы* часто выглядят довольно скучными, утомительными и обремененными множеством более или менее существенных деталей. Справедливости ради отметим, что чаще всего, из приведенных деталей большая часть оказывается не относящейся к делу или, по крайней мере, не проливающей большего света. Возможно, в области компьютеринга, можно будет получить недостающие детали, а на старые взглянуть под иным углом зрения. По крайней мере, идея константности оказывается *относительной*, т.е. ее полезно рассматривать не вообще универсально, а оставаясь в рамках той или иной системы компьютеринга. Попробуем на примере разобраться, чем это может оказаться полезным.

Множество объектов вместе со структурирующими их комбинаторами выглядит довольно однородным, даже вместе с возможностями порождения циклических конструкций с применением комбинатора Y . Может показаться, что в подобной структуре нет места каким-либо системам объектов с интересным и практически значимым поведением. Но это не так.

Попробуем установить, найдется ли среди объектов такой объект V , который характеризуется распределением относительно аппликации:

$$V(ab)\rho = Va\rho(Vb\rho).$$

Таким образом, все, что имеем – это структуру объектов, к которой добавлено это равенство, действие которого проиллюстрировано на рис. 21.

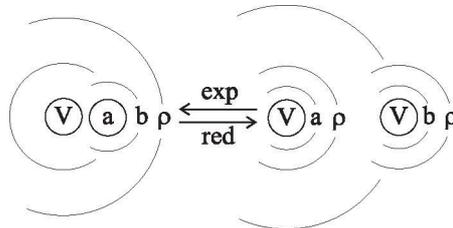


Рис.21. Характеристическое равенство для объекта V : $V(ab)p = Vap(Vbp)$.

Вид левой части этого равенства наводит на мысль, что имеется композиция объектов-отображений V и a , которая применена к объекту-аргументу b , а результат этого применения, в свою очередь, применен к объекту p , играющему роль среды:

$$V(ab)\rho = ((V \circ a)(b))(\rho) \equiv (V \circ a)b\rho \equiv BVab\rho.$$

Эта математическая идея отражена на рис. 22.

Займемся теперь правой частью равенства, характеризующего поведение объекта V , и попытаемся получить ее каноническое представление. К счастью, большую часть комбинационной

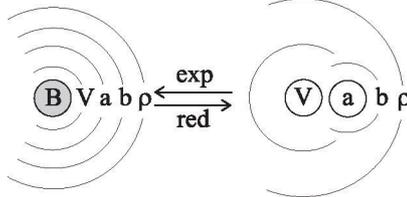


Рис.22. Каноническое представление левой части характеристического равенства объекта V : $V(ab)p = BVabp$. работы по распределению вычислений среди объектов берут на себя комбинаторы Φ и Ψ , а соответствующее приведение выглядит достаточно лаконично:

$$Vap(Vbp) = IVap(Vbp) = \Phi I(Va)(Vb)\rho = \Psi(\Phi I)Vab\rho$$

Соответствующие преобразования отражены на рис. 23.

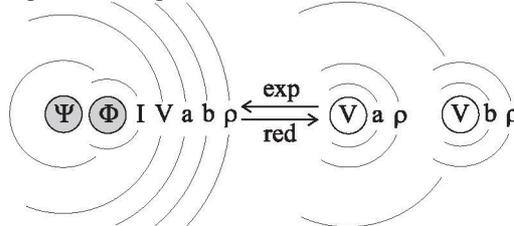


Рис.23. Каноническое представление правой части характеристического равенства объекта V : $Vap(Vbp) = \Psi(\Phi I)Vab\rho$.

Для целей дальнейшей систематизации сведем воедино выполненные преобразования:

$$BVabp = V(ab)p = Vap(Vbp) = \Psi(\Phi I)Vabp$$

Каркасная часть цепочки вывода представлена на рис. 24.

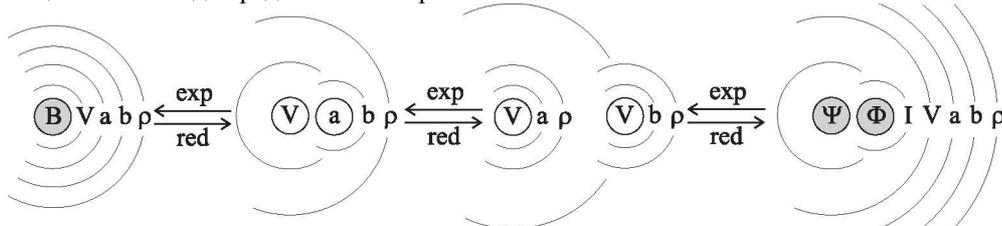


Рис.24. Результирующее характеристическое равенство для объекта V : $BVabp = V(ab)p = Vap(Vbp) = \Psi(\Phi I)Vabp$.

Что же следует из проведенных преобразований? Таких следствий можно получить несколько.

Прежде всего, нетрудно убедиться, что для $V = K$ приведенное построение работает, так что объект V существует.

Вместе с тем можно записать математически лаконичное предложение, фиксирующее основной достигнутый результат:

$$\exists V. \forall a, b, \rho. BVab\rho = \Psi(\Phi I)Vab\rho.$$

То есть, существует объект V такой, что для произвольных объектов a, b, ρ выполняется равенство

$$B = \Psi(\Phi I),$$

которым как раз и характеризуется объект V . Отметим, что объект ρ может рассматриваться как *среда* в том смысле, который вкладывается в этот термин в теории языков программирования (см. [20]; [22], гл. 12). Приведенные соображения приводят к конструкции, приведенной на рис. 25.

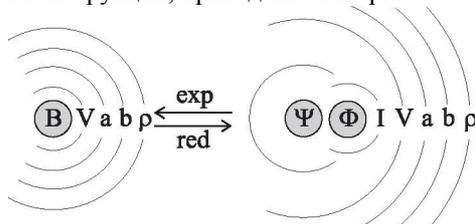


Рис.25. Комбинаторная характеристика, порождающая объект V : $3V. \forall a, b, \rho. BVab\rho = \Phi\{\Phi I\}Vab\rho$.

Благодарности

Несомненно, что обсуждение компьютеринга на конференции «Аппликативные вычислительные системы» – ABC-2008, проведенной в апреле-мае 2008 г. в Институте «ЮрИнфор-МГУ», послужило действенным стимулом для некоторой систематизации изложенных соображений.

Заключение

Естественно, это фиксация ряда ключевых идей, но разработка множественности миров комбинаторов представляется заслуживающей внимания и оправдывающей усилия на подготовку предварительного текстуального материала. Часть из них оказалась включенной в Труды ABC-2008, см. URL <http://jurinfor.exponenta.ru/ACS2008>. Другая часть предполагается к последовательному изложению.

Список литературы

- Barendregt H., Wiedijk F. *The Challenge of Computer Mathematics*. – Transactions of the Royal Society, Vol. 363, №1835, 2005. – pp. 2351-2375. <ftp://ftp.cs.ru.nl/pub/CompMath.Found/Barendregt-Wiedijk.pdf>
- Bell G., Dourish P. *Yesterday's tomorrows: notes on ubiquitous computing's dominant vision*. – Personal Ubiquitous Comput., Vol. 11, №2, 2007. – pp. 133–143. DOI <http://dx.doi.org/10.1007/s00779-006-0071-x>.
- Berners-Lee T., Hall W., Hendler J., O'Hara K., Shadbolt N., and Weitzner D. *A framework for Web science*. – Foundations and Trends in Web Science, Vol. 1, Issue 1, 2006. – pp. 1-130. <http://www.nowpublishers.com/web/>
- Проект «ЛАМБДА», открытый для сотрудничества 25
- Cardelli L., Davies R. *Service combinators for Web computing*. - HP Labs Technical Reports SRC-RR-148, June 1, 1997. - 15 p.

<http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-148.html>

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>

Curry H. B. *Functionality in combinatory logic*. - Proc. National Academy of Sciences of the USA, Vol. 20, 1934. - pp. 584-590. **1**

de Bruijn N.G. *Lambda-calculus notations with nameless dummies: a tool for automatic formula manipulation*. - Indag. Math. 1972, №34, pp. 381-392.

de Bruijn N. G. *A survey of the project Automath*. - In: To H.B. Curry: Essays in combinatory logic, lambda calculus and formalism, Academic Press, 1980. -pp. 579-606. **1**

Denning P. J. *Computing is a natural science*. - Commun. ACM, Vol. 50, №7, 2007. - pp. 13-18.

DOI <http://doi.acm.org/10.1145/1272516.1272529>

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

Kennedy A. *Functional Pearls: Pickler Combinators*. - Journal of Functional Programming, special issue on Functional Pearls, Vol. 14, №6, Cambridge University Press, Nov 2004. - pp. 727-739,

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>

13. MacLennan B. J. *Combinatory Logic for Autonomous Molecular Computation*. - Preprint of paper invited for Information Sciences, 2003.

<http://www.cs.utk.edu/~mclennan/UPIM/CLAMC-IS.pdf>

14. 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>

Schonfinkel M.I. *Über die Bausteine der mathematischen Logik*. Math. Annalen 92, 1924. - pp. 305-316. **1**

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, pp. 311-372.

Scott D. S. *Relating theories of the lambda calculus*. - Hindley J., Seldin J. (eds.) To H.B. Curry: Essays on combinatory logic, lambda calculus and formalism. - N.Y. & L.: Academic Press, 1980, pp. 403-450. **1**

Seldin J. P. *The Logic of Curry and Church*. - In: (Dov Gabbay and John Woods, eds.) *Handbook of the History of Logic*, Vol. 5, Elsevier, 2006. <http://people.uleth.ca/~jonathan.seldin/CCL.pdf>

Selinger P. and Valiron B. *A lambda calculus for quantum computation with classical control*. - Mathematical Structures in Computer Science, 16(3), 2006. -pp. 527-552.

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

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.

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. **7, 15, 15**

Исмаилова Л. Ю. *Логика объектов*. - В кн. [], с. 613-630.

Косиков С. В. *Логика функциональности*. - В кн. [], с. 595-612.

Косиков СВ. *Информационные системы: категорный подход*. - Под ред. Л.Ю. Исмаиловой. - М.: «ЮрИнфоР-Пресс®», 2005. - 96 с.

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