1976

Логика и языки программирования

Д. С. Скотт

Оксфордский университет

[Д. С. Скотт является одним из двух лауреатов премии Тьюринга 1976 г., врученной ему на ежегодной конференции ACM 20 октября в Хьюстоне. Статья другого лауреата, М. О. Рабина, «Сложность вычислений» напечатана на стр. 371.]

Уже долгое время логика пытается выяснить, являются ли ответы на те или иные вопросы принципиально вычислимыми, поскольку решение этих проблем устанавливает границы возможностей формализации. Последние достижения теории сложности позволили получать точные оценки эффективности методов решений. Однако эти методы применимы только к логике, а значимость логических методов в других, более прикладных областях теории вычислений остается пока неясной.

Языки программирования, обладая весьма развитой синтаксической формализацией, очевидно, более всего подходят для выяснения этого вопроса, однако семантическая теория языков еще далека от завершенности. Несмотря на множество примеров, мы пока не можем дать достаточно общие, математически обоснованные ответы на следующие вопросы: что такое вычислительная машина? что такое процесс вычисления? каким образом (или насколько хорошо) вычислительная машина моделирует процесс? Программы, естественно, входят в описание процессов. Поэтому определение точного смысла некой программы требует объяснения того, что является объектами вычисления (в некотором смысле статику проблемы) и каким образом они трансформируются (динамику проблемы).

До сих пор теории автоматов и сетей, хотя и наиболее интересные с точки зрения динамики, формализовали только часть области, и похоже, что внимание уделялось в основном алгебраическим аспектам и конечным наборам состояний. По-видимому, понимание свойств более сложных программ вовлекает нас в область бесконечных объектов и заставляет использовать несколько уровней объяснения на пути от концептуальных идей к окончательному моделированию на реальных ЭВМ. Эти уровни могут быть математически точно обоснованы, если найти правильные абстракции для представления необходимых структур.

Многие независимые исследователи использовали метод представления типов данных в качестве решеток (или методов частичной упорядоченности), упорядоченных относительно содержания информации, и их непрерывные отображения. Эти работы продемонстрировали гибкость данного подхода в обеспечении определений и доказательств, прозрачных и относительно независимых от реализации. Тем не менее необходимо сделать еще очень много, чтобы показать, каким образом абстрактная концептуализация может (или не может) быть реализована, прежде чем заявить, что мы имеем единую теорию.

Как последнему солауреату премии Тьюринга, мне доставляет величайшее удовольствие разделить ее и эту трибуну с

Микаэлем Рабином. К сожалению, со времени написания нашей статьи в 1959 г. мы почти не имели возможности для сотрудничества, что является для меня большой потерей: при совместной работе я добиваюсь бóльших результатов. Но организовать подходящие условия для такого сотрудничества нелегко, особенно на стыке научных дисциплин, когда людей к тому же разделяют государственные границы. Тем не менее я с большим интересом и восхищением следил за его работой. Как вы сегодня уже слышали, Рабину удалось *применить* идеи логики, связанные с разрешимостью, вычислимостью и сложностью, к вопросам, представляющим действительный интерес в математике и теории вычислений. Он, как и многие другие, активно работает над созданием новых методов анализа для широкого и весьма перспективного класса алгоритмических проблем. Эти аспекты теории вычислений находятся вне моей компетенции, так как за прошедшие годы мои научные интересы и интересы Рабина разошлись. Начиная с конца 1960-х годов я сосредоточился в основном на возможности применения логических идей для лучшего *концептуального* понимания языков программирования. Поэтому я больше не буду детально говорить о нашей совместной работе, а остановлюсь на своих собственных разработках, планах и надеждах.

Трудности получения точного и всеобъемлющего представления о языке возникли в период разработки громоздких «универсальных» машинных языков. В настоящее время, как мне кажется, мы стоим на пороге еще одной технологической революции, которая полностью перевернет наше представление о вычислительных машинах и их программном обеспечении. (Я только что отметил стремление ACM полностью исключить термин «машина».) Возможно, что громоздкие языки окажутся недостаточно гибкими, однако, я думаю, проблема *семантики* наверняка останется. Хотелось бы верить, что мои исследования, проведенные опять-таки совместно с другими учеными, и в особенности с покойным Кристофером Стречи, явились существенным вкладом в основание работ по семантике. Поживем — увидим. Надеюсь также, работы по семантике недолго будут оставаться несвязанными с исследованиями в области, разрабатываемой Рабином.

ПОПЫТКА ОПРАВДАТЬСЯ НЕ ОПРАВДЫВАЯСЬ

Вообще говоря, я считаю, что выступающие не должны оправдываться: это только вызывает у аудитории чувство неловкости. Но на такой встрече, как эта, одно оправдание все же необходимо (вместе с отречением).

Те из вас, кому известна моя профессиональная подготовка, могут вспомнить сэра Николаса Гимкрэка, героя пьесы «Виртуоз», написанной Томасом Шэдуэллом в 1676 г. с целью слегка пошутить над замечательными опытами, проводившимися в то время Лондонским Королевским обществом. В одной из сцен пьесы сэр Николас лежит на столе и пытается научиться плавать, имитируя движения лягушки в чаше с водой. На вопрос, пробовал ли *он* научиться плавать *в воде*, сэр Николас отвечает, что ненавидит воду и никогда не подойдет к ней! «Мне достаточно теоретического изучения процесса плавания,— отвечает он,— мне безразлична его практическая польза. Я редко довожу что-либо до практического применения... Знание есть моя конечная цель».

Несмотря на совпадение моих и сэра Гимкрэка конечных целей, мне хотелось бы отмежеваться от пренебрежения практическими аспектами. Дело, однако, в том, что у меня отсутствует практический опыт в современном программировании, и по необходимости мне пришлось ограничиться наблюдением программирования, получая знания «из вторых рук», наблюдая действия «лягушек и других существ». К счастью, некоторые из этих «лягушек» умеют говорить, и мне пришлось изучать с ними чужой для меня язык. Однако вполне возможно, что я не до конца понимал, о чем идет речь. Но я пытался читать и быть в курсе происходящего. Прошу простить мой непрофессионализм в области программирования, и я, естественно, не хотел бы выступать с поучениями: большинство выступавших были в этом смысле более подготовлены и сообщили нам много ценного. Единственное, что я пытался сделать,— это довести некоторые результаты из области логики, которые, как мне казалось, могут быть полезны в программировании, до тех, кто мог бы их использовать. Я также пытался получить свои собственные результаты, и вам судить, насколько мне это удалось.

К моему большому удовлетворению, сегодня мне не нужно извиняться за недостаток опубликованного материала; мне нужно было бы это сделать, если бы я написал свою речь в день получения приглашения. В августе была напечатана замечательная статья Роберта Теннета [14] по денотационной семантике, и я горячо рекомендую ее как отправную работу по этой теме. Теннет не только дает серьезные примеры, идущие намного дальше опубликованного мной и Стречи, но и представляет хорошо продуманную библиографию.

В прошлом месяце вышла значительная книга Милна и Стречи. К несчастью, совершенно неожиданная и безвременная кончина Стречи не позволила ему приступить к пересмотру и исправлению рукописи. С его смертью мы потеряли много в стиле и проницательности (не говоря уже о вдохновении), однако

Роберт Милн замечательно довел до конца их совместную работу. Самым значительным в этой книге является то, что проблема сложного языка обсуждается в ней от *начала* до *конца*. Кому-то изложение может показаться слишком строгим, однако суть в том, что семантика в этой работе — не плод произвольных догадок, а реально обоснована. Она является результатом серьезного и глубокого осмысления, и у нас есть возможность детально разобраться и решить, может ли данный подход быть плодотворным. Милн построил описание таким образом, что стало возможным восприятие языка на многих уровнях, вплоть до конечного компилятора. Он не пытался обойти трудности. Хотя эта книга и не написана тем язвительным и беспечным языком, каким обычно пользовался сам Стречи, тем не менее ее можно считать достойным памятником заключительному этапу работы Стречи, к тому же она содержит немало оригинальных идей Милна. (Я могу так говорить потому, что сам не участвовал в написании этой книги.)

Недавно была опубликована не очень большая, но интересная работа Донахью [4], в которой содержатся не обсуждавшиеся ранее или обсуждавшиеся, но с другой точки зрения, вопросы. Она написана совершенно независимо от меня и Стречи, и я был очень рад ее появлению.

Скоро выйдет учебник Джо Стоя [13], который дополнит работы предыдущих авторов и, видимо, явится хорошим учебным пособием, так как Стой имеет немалый опыт преподавательской работы как в Оксфордском университете, так и в Массачусетском технологическом институте.

Из фундаментальных работ сейчас должна появиться моя собственная переработанная статья [12]. Она была написана с точки зрения использования операторов перечисления в более «классической» теории рекурсии, и ее связь с практическим программированием на первый взгляд может показаться не совсем очевидной. Но меня успокаивает то, что другие работы объясняют использование этой теории именно так, как я и предполагал.

К счастью, все вышеперечисленные авторы широко цитируют литературу, и поэтому сегодня нет необходимости вдаваться в исторические детали. Я позволю себе только сказать, что и многие другие исследователи использовали различные идеи Стречи, как и мои идеи, и упоминание об их работах можно найти не только в библиографиях, но и, например, в последних докладах Мана [7] и Бёма [1]. Пусть меня простят те, кого я невольно не назвал сейчас, — они знают, как я ценю их содействие и интерес к моим работам.

НЕМНОГО О СЕБЕ

Я родился в Калифорнии и приступил к работе в области математической логики, будучи студентом младших курсов в Беркли в начале 1950-х годов. Больше всего на меня повлияли, конечно, Альфред Тарский, его коллеги и студенты Калифорнийского университета. Наряду с другими предметами под руководством Рафаэля и Джулии Робинсон, которых я хочу поблагодарить за многие ценные идеи, я изучал теорию рекурсивных функций. В то же время самостоятельно я познакомился с λ-исчислением Чёрча и Карри (которое в начале в буквальном смысле было для меня кошмаром). Особенно большое влияние на меня оказало изучение семантики Тарского и его определение истинности для формализованных языков. Эти концепции, как вы знаете, до сих пор широко обсуждаются в философии естественного языка. Я пытался применить идеи подхода Тарского к алгоритмическим языкам, преимущество которых заключается по крайней мере в том, что они достаточно хорошо синтаксически формализованы. Возможно, требует обсуждения, действительно ли мне удалось, руководствуясь схемами Стречи и других ученых, найти правильные определения терминов. Именно я первым заявил о том, что *не все* проблемы решаются одним лишь подбором денотатов к некоторым языкам: для языков типа (чистого) λ-исчисления нужные определения в большинстве случаев найдены, однако многие понятия программирования до сих пор остаются неопределенными.

Свою дипломную работу я закончил в Принстоне в 1958 г. Руководителем моим был Алонсо Чёрч, который в то же время руководил диссертацией Микаэля Рабина. Тогда мы и встретились с Рабином, а нашу совместную работу по теории автоматов сделали в 1957 г. во время летней практики в IBM. Конечно, мы не были единственными в этой области, однако нам удалось разработать несколько основных идей. В то время я уже думал о проекте математического определения вычислительной машины. Сейчас мне кажется, что метод конечных состояний представляет лишь частичный интерес и не имеет особого практического значения. Правда, большинство физических вычислительных машин можно смоделировать как устройства с конечным числом состояний, однако эта *ограниченность* (конечность) едва ли является их наиболее значительной чертой, и взгляд на ЭВМ как на автомат часто бывает весьма поверхностным.

Два последних достижения в области теории автоматов кажутся мне наиболее интересными, по крайней мере с точки зрения математики: многоуровневая иерархия Хомского и связь с полугруппами. С точки зрения алгебры (во всяком случае,

по моему мнению) Эйленберг, этот Евклид теории автоматов, в своих работах [5] действительно сказал последнее слово. Я хочу подчеркнуть и то, что он сумел обойтись без теории абстрактных категорий. Категории могут привести к хорошим решениям (см. Мэйнз [7]), но слишком раннее их использование значительно затрудняет понимание. Это мое личное мнение.

В некоторых аспектах иерархия Хомского в конце концов вызывает разочарование. Контекстно-свободные языки действительно очень важны, и всем необходимо ознакомиться с ними. Однако мне не совсем ясно, что за этим следует, если вообще что-либо следует. Существует множество других семейств языков, но из хаоса не возникло так уж много порядка. Я не думаю, что в этой области уже сказано последнее слово. Не зная истинного направления и разочаровавшись в теории автоматов, казавшейся мне чрезмерно сложной, я оставил работу в этой области. Однажды я попытался связать автоматы и языки программирования, предложив более систематический способ отделения машины от программы. Эйленбергу эта идея совершенно не понравилась, но мне было очень приятно увидеть в последней книге Кларка и Коуэлла [2] ее элегантное воплощение, предложенное Питером Ландином. Признаю, что это не алгебра, но мне кажется, что это не что иное, как (элементарное, хотя и в некоторой степени теоретическое) программирование. Хотелось бы увидеть следующий шаг, который должен привести к чему-то между Манном [8] и Милном — Стречи [9].

В Принстоне я впервые познакомился с настоящей вычислительной машиной, «доисторической» машиной фон Неймана. И благодарить за это я должен Эктона Формана. Эта машина кажется сейчас совсем устаревшей, однако она была *действующей*, и работа на ней доставила мне и Хейлу Троттеру большое удовлетворение. Как печально было видеть ее мертвый корпус в Смитсонианском музее без каких-либо признаков того, какой она была при жизни.

Из Принстона я перешел в Чикагский университет, где в течение двух лет преподавал на математическом факультете. Там я и встретился с Бобом Эшенхёрстом и Ником Метрополисом. К сожалению, мое короткое пребывание в университете не позволило мне поработать с ними: как обычно, факультеты слишком отдалены друг от друга. (Естественно, что я говорю только о связях с программированием и не пытаюсь говорить о своей работе в области математики и логики.)

Затем в течение трех лет я работал в Беркли, где через Гарри Хаски и Рене де Вожелера познакомился со многими специалистами в области программирования. Последний также ознакомил меня в деталях с языком Алгол-60. В то время в Беркли еще не было факультета информатики. Через некоторое

время по личным причинам мне пришлось переехать в Стэнфорд. Таким образом, хотя я и преподавал теорию вычислений в течение одного семестра, моя работа в Беркли не имела особого значения. Единственное, о чем я всегда буду сожалеть, — это то, что я не изучил досконально работу Дика и Эммы Лехмер, чей способ получения *результатов* в теории чисел с помощью ЭВМ восхищает меня до сих пор. Теперь, когда с помощью ЭВМ решена проблема четырех красок, мы становимся свидетелями развития методов автоматического доказательства теорем. Мне очень жаль, что я не занимался этой проблемой.

Все согласятся с тем, что с начала 1960-х годов факультет информатики в Стэнфорде был одним из лучших в стране, и, конечно, непонятно, почему я все-таки ушел оттуда. Возможно, дело в том, что моя должность объединяла работу на факультетах философии и математики. Мне кажется, что мой личный недостаток заключается в том, что я не всегда знаю, где мне следует быть и чем я хочу заниматься. Однако оставим личные переживания. У меня сложились хорошие отношения с аспирантами на факультете в Форсайте, где мы провели замечательные лекции и семинары. Лично на меня и на мои представления в области теории вычисления оказали большое влияние Джон Маккарти и Пэт Саппес, а также остальные члены их группы. Вместе с Солом Феферманом и Георгом Крайзелем мы создали сильную группу логиков. В эту группу аспирантов-логиков входил и Ричард Платек, работа которого произвела на меня особое впечатление несколько лет спустя, когда я понял способ применения некоторых из его идей.

В это же время я получил годичный отпуск для поездки в Амстердам, который неожиданно оказался поворотным пунктом в моем творческом развитии. Я не буду углубляться в детали, так как это сложная история, отмечу только, что 1968/69 академический год стал для меня годом глубокого кризиса, и мне до сих пор больно вспоминать об этом. К счастью или к несчастью, Пэт Саппес предложил включить меня в рабочую группу 2.2 Международной федерации по обработке информации (в настоящее время — группа формального описания концепций программирования). Председателем ее был тогда Том Стил, и на заседании в Вене я впервые встретился с Кристофером Стречи. Дискуссии, возникавшие в этой группе, носили настолько острый характер, что я был действительно рад тому, что не принимал участия в работе комитетов по более значительным вопросам, как, например, комитета по Алголу. Тем не менее я полагаю, что дискуссии являются отличным терапевтическим средством: они выявляют лучшие и худшие качества людей. Во всяком случае, нужно научиться защищать себя. Из всех участников этих баталий я выделил Стречи, его стиль, его

мысли мне более всего импонировали, хотя, как мне показалось, он иногда сильно преувеличивал значение некоторых своих идей. Однако некоторые из них побудили меня заняться более глубоким изучением проблемы.

Лишь в конце моего годичного пребывания в Амстердаме я приступил к обсуждению некоторых проблем с Жако де Баккером, и только в переписке, которую мы вели в течение всего лета, удалось окончательно оформить наши идеи. Значительное влияние на меня оказала и Венская группа IBM, с деятельностью которой я ознакомился, участвуя в рабочей группе 2.2. К этому времени я решил перейти на факультет философии Принстонского университета, но прежде, чем возвратиться домой, я попросил дополнительный отпуск и осенью 1969 г. заехал к Стречи в Оксфорд. Этот период стал для меня периодом наиболее активной деятельности: в течение многих дней у меня было что-то вроде нервной горячки. Сотрудничество со Стречи на протяжении этих нескольких недель стало лучшим временем в моей профессиональной жизни. Наша новая встреча состоялась летом следующего года в Принстоне. К сожалению, к 1972 г., когда я окончательно перебрался в Оксфорд, мы оба были настолько заняты преподавательской работой и административными проблемами, что наше сотрудничество оказалось практически невозможным. К тому же Стречи был настолько обескуражен постоянным отсутствием средств на исследовательские работы и помощи в преподавательской деятельности, что в конце концов он ушел из университета и занялся вместе с Милном подготовкой к публикации своей книги. (Для него это было перенапряжением и отрицательно сказалось на здоровье. Как много бы я дал, чтобы он увидел свою работу опубликованной!)

Возвращаясь к 1969 г., должен сказать, что я начал с того, что показал Стречи его *полную неправоту* и необходимость делать все совершенно иначе. В самом начале Роджер Пенроуз привлек его внимание к λ-исчислению, и он разработал удобный способ использования этой нотации функциональной абстракции в объяснении концепций программирования. Тем не менее этот способ был *формальным*, и я пытался доказать, что у него не было математической базы. Я уже рассказывал эту историю и, чтобы не быть многословным, скажу только, что мне удалось убедить его отказаться от бестипового λ-исчисления. Однако вскоре, по мере того как одно следствие из моего предложения следовало за другим, я стал понимать, что вычислимые функции могут быть определены в большом разнообразии пространств. Настоящим шагом вперед стало понимание того, что функциональные пространства действительно являются хорошими пространствами, и я ясно помню, как логик Анджей

Мостовски, который в это же время находился в Оксфорде, просто не мог поверить, что тип функциональных пространств, который я определил, вообще мог иметь конструктивное описание. Но когда я сам в этом убедился, у меня родилась мысль о существовании более удивительных, чем мы могли предположить, возможностях использования функциональных пространств. При появлении сомнений в полезности принудительной жесткости логических типов, в чем я пытался убедить Стречи, я пришел к выводу, что одно из этих пространств изоморфно своему собственному функциональному пространству, которое обеспечивает модель «бестипового» λ-исчисления. Окончание этой истории описано в литературе.

[Интересные сведения, проливающие свет на историю λ-исчисления, дает участие в ней Алана Тьюринга. Он учился в Принстоне вместе с Чёрчем и связал вычислимость с (формальным) λ-исчислением в 1936—1937 гг. У Кроссли можно найти пояснения об отношении Стива Клини к его работе (и к последующему влиянию λ-исчисления) (Кроссли [3]). (Конечно, поздние взгляды Тьюринга на компьютеры оказали на Стречи большое влияние, однако сейчас не время проводить полный исторический анализ.) Хотя я никогда не встречался с Тьюрингом (он умер в 1954 г.), соприкосновение с ним через Чёрча, Стречи и моих нынешних коллег по Оксфорду, Ле Фокса и Робина Ганди, оказалось достаточно близким несмотря на то, что в то время я только заканчивал университет, Чёрч больше не занимался λ-исчислением и мы никогда не обсуждали его работу с Тьюрингом.]

Я нахожу очень странным то обстоятельство, что мои модели λ-исчисления не были найдены кем-либо ранее, но я очень воодушевлен тем, что новые типы моделей с новыми свойствами, как, например, домены Гордона Плоткина [10], открываются сейчас. Лично я уверен, что для этого все уже подготовлено как на теоретическом, так и на прикладном уровнях. Джон Рейнолдс и Роберт Милн независимо друг от друга ввели новый индуктивный метод доказательства эквивалентностей; Робин Милнер продолжает в Эдинбурге интересную работу по LCF и технике доказательств. Начало направлению доказательства через модели было положено теоремой Дэвида Парка о связи оператора неподвижной точки с так называемым парадоксальным комбинатором λ-исчисления, и это положило начало изучению бесконечных, но вычисляемых операторов, которое продолжается по многим линиям. Другое направление разрабатывается в Новосибирске под руководством Ю. Л. Ершова, а Карл Х. Хофманн и его группа ознакомили меня с очень интересными связями с топологической алгеброй. К сожалению, я не могу здесь перечислить всех, работающих в этой области.

Мне особенно приятно доложить этому собранию, что Тони Хоар недавно согласился принять кафедру вычислений в Оксфордском университете, которая освободилась после того, как умер Стречи. Это назначение открывает новые возможности для сотрудничества как с Хоаром, так и с теми студентами, которых он привлечет в свою группу, как только займет этот пост в будущем году. Как вы понимаете, практическим аспектам использования и создания компьютерных языков и методологии программирования в Оксфорде будет придаваться особое значение (что делал и Стречи), и это действительно пойдет на пользу всем; я надеюсь, что открываются и новые прекрасные возможности для теоретических исследований.

НЕКОТОРЫЕ СЕМАНТИЧЕСКИЕ СТРУКТУРЫ

Обращаясь к технической стороне, я хотел бы кратко пояснить сущность действия моей конструкции и возможности ее развития. Здесь мы не будем доказывать «правильность» этих абстракций, ее обоснование можно найти в упомянутых ранее работах.

Проще всего показать сущность этой конструкции на двух областях: $\mathscr{B}$ — *области булевских значений*, и $\mathscr{S}=\mathscr{B}^{\infty}$ — области *бесконечных последовательностей* булевских значений. Важным моментом является то, что мы собираемся принять идею *частичных* функций, при математическом представлении которых функции время от времени принимают *частичные значения*. Использование $\mathscr{B}$ делает эту идею очень простой. Запишем:

$$\mathscr{B} = \{\text{истина, ложь, } \bot\},$$

где $\bot$ является дополнительным элементом, называемым «неопределенность». Для того чтобы закрепить за $\bot$ его место, введем на области $\mathscr{B}$ частичный порядок $\sqsubseteq$, где

$x \sqsubseteq y$ тогда и только тогда, когда или $x = \bot$, или $x = y$,

для всех x, y из B. Мы можем читать «$\sqsubseteq$» как то, что информация, содержащаяся в x, *содержится* в информации, содержащейся в y. Элемент $\bot$ поэтому содержит пустую информацию. Эта схема показана на рис. 1.

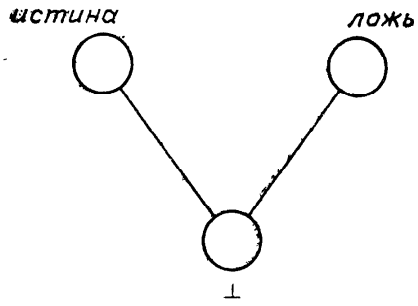

Рис. 1. Булевские значения.

(Примечание: во многих статьях я отстаивал использование решеток, которые как частичные порядки имеют и «дно» («нижний» элемент) $\bot$, и «верх» («верхний» элемент) $\top$, что позволяет ут-

верждать, что $\bot \subseteq x \subseteq \top$ верно для всех элементов области. Это предположение не было безоговорочно принято по многим причинам, о которых я не буду говорить. Некоторые рассуждения о разумности этого можно найти у Скотта [12], хотя, конечно, исследуемые им структуры очень специфичны. Вероятно, лучше всего не включать, но и не исключать $\top$, и для простоты я не буду более упоминать об этом сегодня.)

Рассмотрим теперь $\mathscr{S}$, область последовательностей. Мы будем использовать нотацию, где нижние индексы обозначают координаты; таким образом, для всех $x \in \mathscr{S}$

$$x = \langle x_n \rangle_{n=0}^{\infty}.$$

Каждый терм таков, что $x_n \in \mathscr{B}$, потому что $\mathscr{S} = \mathscr{B}^{\infty}$. Здесь имеется в виду «прямое произведение», и *формально* мы определяем частичный порядок $\subseteq$ на $\mathscr{S}$ следующим образом:

$x \subseteq y$ тогда и только тогда, когда $x_n \subseteq y_n$ для всех n.

Неформально это означает, что последовательность y «лучше» информационно, чем последовательность x, тогда и только тогда, когда некоторые координаты x, которые были «неопределенными», становятся «определенными» при переходе от x к y. Например, каждая из следующих последовательностей находится в отношении $\subseteq$ к следующей за ней:

$\langle \bot, \bot, \bot, \bot, \ldots \rangle$,

$\langle$*истина*, $\bot, \bot, \bot, \ldots \rangle$,

$\langle$*истина*, *ложь*, $\bot, \bot, \ldots \rangle$,

$\langle$*истина*, *ложь*, *истина*, $\bot, \ldots \rangle$.

Очевидно, что этот список может быть бесконечно расширен, и не обязательно рассматривать координаты в строгом порядке $n = 0, 1, 2 \ldots$. Таким образом, отношение $\subseteq$ на $\mathscr{S}$ является более сложным, чем определенное ранее на $\mathscr{B}$.

Очевидным различием между $\mathscr{B}$ и $\mathscr{S}$ является то, что $\mathscr{B}$ конечно, в то время как $\mathscr{S}$ имеет бесконечно много элементов. В $\mathscr{S}$ также некоторые элементы имеют бесконечное информационное содержание, тогда как в $\mathscr{B}$ это не так. Однако мы можем использовать частичный порядок на $\mathscr{S}$ для абстрактного объяснения того, что мы подразумеваем под «конечной аппроксимацией» и «пределами». Последовательности, представленные выше, являются конечными в $\mathscr{S}$, так как они имеют только конечное число координат, отличных от $\bot$. Любой данный $x \in \mathscr{S}$ можно обрезать до конечного элемента, определяя

$$(x \upharpoonright m)_n = \begin{cases} x_n, & \text{если } n < m; \\ \bot, & \text{если это не так.} \end{cases}$$

Из наших определений видно, что

$$x \restriction m \subseteq x \restriction (m+1) \subseteq x,$$

так что $x \restriction m$ «стремятся» к пределу; и фактически этот предел есть первоначальный x. Запишем это следующим образом:

$$x = \bigcup_{m=0}^{\infty} (x \restriction m),$$

где $\bigcup$ есть sup или операция вычисления наименьшей верхней грани в частично упорядоченном множестве $\mathscr{S}$. Дело в том, что $\mathscr{S}$ имеет много sup; и для любых элементов $y^{(m)} \subseteq y^{(m+1)}$ в $\mathscr{S}$ (независимо от того, конечны они или нет) мы можем определить предел z, где

$$z = \bigcup_{m=0}^{\infty} y^{(m)}.$$

(Совет: спросите себя, какие координаты будут у z.) Мы не можем пересказать здесь все детали, но $\mathscr{S}$ действительно является топологическим пространством и z действительно является пределом. Поэтому, хотя $\mathscr{S}$ и бесконечно, мы можем вернуться в область *конечных* операций и обсуждать *вычислимые* операции на $\mathscr{S}$ и на более сложных областях.

Кроме последовательностей и частично упорядоченных структур на $\mathscr{S}$ мы можем определить много видов алгебраических структур. Именно поэтому $\mathscr{S}$ является подходящим примером. Например, с точностью до изоморфизма пространство $\mathscr{S}$ удовлетворяет

$$\mathscr{S} = \mathscr{S} \times \mathscr{S},$$

где в правой части равенства подразумевается обычное бинарное прямое произведение. Область $\mathscr{S} \times \mathscr{S}$ состоит из всех упорядоченных пар $\langle x, y \rangle$, где $x, y \in \mathscr{S}$, отношение $\subseteq$ определяется на $\mathscr{S} \times \mathscr{S}$ как

$\langle x, y \rangle \subseteq \langle x', y' \rangle$ тогда и только тогда, когда $x \subseteq x'$ и $y \subseteq y'$.

Но для всех практических целей можно отождествить $\langle x, y \rangle$ с последовательностью, уже содержащейся в $\mathscr{S}$. Действительно, покоординатно мы можем определить

$$\langle x, y \rangle_n = \begin{cases} x_k, & \text{если } n = 2k; \\ y_k, & \text{если } n = 2k+1. \end{cases}$$

Введенный выше критерий для $\subseteq$ между парами будет выполнен, и мы можем сказать, что на $\mathscr{S}$ определена парная функция (отображающая два значения в одно).

Парная функция $\langle .,. \rangle$ на $\mathscr{S}$ обладает многими интересными свойствами. В сущности, мы уже заметили, что она *монотонна* (как только увеличивается информационное содержание x и y, увеличивается информация, содержащаяся в $\langle x, y \rangle$). Более важно, что функция $\langle .,. \rangle$ является *непрерывной* в следующем определенном смысле:

$$\langle x, y \rangle = \bigcup_{m=0}^{\infty} \langle x \upharpoonright m, y \upharpoonright m \rangle,$$

что означает, что функция $\langle .,. \rangle$ хорошо ведет себя при конечных аппроксимациях. И это только один пример. Для данного подхода важна целая *теория* монотонных и непрерывных функций.

Используя даже такую небольшую структуру, которую мы определили на $\mathscr{S}$, можно дать описание *языка*. Для иллюстрации остановимся на двух изоморфизмах, которым удовлетворяет $\mathscr{S}$, а именно $\mathscr{S}=\mathscr{B}\times\mathscr{S}$ и $\mathscr{S}=\mathscr{S}\times\mathscr{S}$. Первый определяет $\mathscr{S}$ как нечто связанное с (бесконечными) последовательностями булевских значений, тогда как второй напоминает о приведенной выше парной функции. На рис. 2 приводится краткое БНФ-определение (использующее форму Бэкуса — Наура) языка с двумя видами выражений: булевское значение (β) и последовательность булевских значений (σ).

β : : = **true** | **false** | **head** σ
σ : : = β^* | $\beta\sigma$ | **tail** σ |
if β **then** σ' **else** σ'' |
even σ | **odd** σ | **merge** $\sigma'\sigma''$

Рис. 2. Краткий язык.

Этот язык действительно очень краткий: без переменных, без объявлений, без присваиваний, только небольшой набор константных термов. Заметим, что выбранная нотация предназначена для того, чтобы сделать значение этих выражений очевидным. Так, если σ обозначает последовательность x, то **голова** σ (head) должна обозначать первый терм x_0 в последовательности x. Так как $x_0\in\mathscr{B}$ и $x\in\mathscr{S}$, то мы сохраняем смысл введенных нами типов объектов.

В более строгом смысле для каждого выражения мы сможем определить его (константное) значение $[[\cdot]]$; так что $[[\beta]]\in\mathscr{B}$ для булевских выражений β и $[[\sigma]]\in\mathscr{S}$ для выражений над последовательностями. Так как БНФ-определение языка содержит десять видов предложений, мы могли бы выписать десять равенств, чтобы полностью определить семантику приведенного языка; мы удовольствуемся здесь некоторыми из

них. Принимая во внимание замечания предыдущего абзаца, запишем:

$$[\![\,\mathbf{head}\ \sigma\,]\!] = [\![\,\sigma\,]\!]_0.$$

Выражение β^* создает бесконечную последовательность булевских значений:

$$[\![\,\beta^*\,]\!] = \langle [\![\,\beta\,]\!], [\![\,\beta\,]\!], [\![\,\beta\,]\!], [\![\,\beta\,]\!], \ldots\rangle.$$

(Эта нотация, хотя и приблизительная, но понятная.) Точно так же определим

$$[\![\,\beta\sigma\,]\!] = \langle [\![\,\beta\,]\!], [\![\,\sigma\,]\!]_0, [\![\,\sigma\,]\!]_1, [\![\,\sigma\,]\!]_2, \ldots\rangle;$$

и

$$[\![\,\mathbf{tail}\ \sigma\,]\!] = \langle [\![\,\sigma\,]\!]_1, [\![\,\sigma\,]\!]_2, [\![\,\sigma\,]\!]_3, [\![\,\sigma\,]\!]_4, \ldots\rangle.$$

Далее

$$[\![\,\mathbf{even}\ \sigma\,]\!] = \langle [\![\,\sigma\,]\!]_0, [\![\,\sigma\,]\!]_2, [\![\,\sigma\,]\!]_4, [\![\,\sigma\,]\!]_6, \ldots\rangle;$$

и

$$[\![\,\mathbf{merge}\ \sigma'\,\sigma''\,]\!] = \langle [\![\,\sigma'\,]\!], [\![\,\sigma''\,]\!]\rangle.$$

Этого достаточно, чтобы дать представление о самой идее. Но нужно понять, что здесь мы обсуждаем одно из возможных представлений, что $\mathscr{S}$ удовлетворяет значительно большему числу изоморфизмов (например, $\mathscr{S}=\mathscr{S}\times\mathscr{S}\times\mathscr{S}$) и существует очень много способов — причем вычислимых — разбивать на части и комбинировать последовательности булевских значений.

ФУНКЦИОНАЛЬНОЕ ПРОСТРАНСТВО

Неверно было бы считать, что вся моя идея исчерпывается только содержанием изложенного. Это соответствовало бы элементарному уровню программных схем (см. ван Эмден — Ковальски [6] или Манна [8], последняя глава). То, что некоторые называют «семантикой неподвижной точки» (мне самому термин «fixpoint» не нравится), является лишь *первой* частью моей работы. Вторая же часть включает функции, в качестве аргументов которых выступают также функции, т. е. функции

высших порядков, и тем самым мы выходим за уровень программных схем. В самом деле, методы неподвижной точки могут быть применены и к функциям высшего порядка, но это не единственное их достоинство. Для более полного определения в качестве семантической структуры необходимо использовать *функциональное пространство.* Я пытался обратить на это особое внимание еще в 1969 г., но многие тогда не понимали меня.

Предположим, что $\mathscr{D}'$ и $\mathscr{D}''$ — два домена обсуждаемого нами вида (скажем, $\mathscr{B}$, или $\mathscr{B}\times\mathscr{B}$, или $\mathscr{S}$, или что-либо еще). Под $[\mathscr{D}'\to\mathscr{D}'']$ будем понимать домен *всех* монотонных и непрерывных функций f, отображающих $\mathscr{D}'$ в $\mathscr{D}''$. Именно это я и имею в виду под термином «*функциональное пространство*». Это несложно с математической точки зрения, но и не совсем очевидно, что $[\mathscr{D}'\to\mathscr{D}'']$ опять-таки является доменом того же самого вида, хотя, возможно, и более сложной структуры. Я не могу этого здесь доказывать, но по крайней мере я могу определить отношение $\sqsubseteq$ на функциональном пространстве:

$f\sqsubseteq g$ тогда и только тогда, когда $f(x)\sqsubseteq g(x)$ для всех $x\in\mathscr{D}'$.

В трактовке функций как абстрактных объектов нет ничего нового; но то, что они могут выступать в качестве объектов вычисления, требует проверки. Отношение $\sqsubseteq$ на $[\mathscr{D}'\to\mathscr{D}'']$ является первым шагом на пути доказательства этого, и он приводит нас к знакомому понятию конечной аппроксимации функций (прошу прощения, но у меня нет времени уточнять). Как только мы в этом убеждаемся, мы видим возможность *итерации* функциональных пространств; то же самое и в случае $[[\mathscr{D}'\to\mathscr{D}'']\to\mathscr{D}''']$. Это утверждение не так уж невероятно, как может показаться на первый взгляд, т. е. наша теория определяет $f(x)$ как бинарную вычислимую функцию от переменной f и переменной x. И как операция $f(x)$ может рассматриваться как элемент функнционального пространства:

$$[[[\mathscr{D}'\to\mathscr{D}'']\times\mathscr{D}']\to\mathscr{D}''].$$

Это только начало теории таких операторов (или *комбинаторов*, как их называют Карри и Чёрч).

Усвоив все это, давайте попытаемся написать бесконечную итерацию функциональных пространств, взяв $\mathscr{S}$ в качестве начального приближения. Пусть $\mathscr{F}_0=\mathscr{S}$ и $\mathscr{F}_{n+1}=[\mathscr{F}_n\to\mathscr{S}]$. Тогда $\mathscr{F}_1=[\mathscr{S}\to\mathscr{S}]$ и

$$\mathscr{F}_4=[[[\mathscr{S}\to\mathscr{S}]\to\mathscr{S}]\to\mathscr{S}]\to\mathscr{S}].$$

Поверьте мне на слово, что все это очень конструктивно (именно *потому*, что мы использовали только непрерывные функции).

Совершенно очевидно, что существует естественная трактовка всей этой совокупности. Сначала $\mathscr{S}$ «содержится» в прост-

ранстве $[\mathscr{S}\to\mathscr{S}]$ как подпространство. Сопоставим каждому $x\in\mathscr{S}$ соответствующую константную функцию из $[\mathscr{S}\to\mathscr{S}]$. Из нашего определения следует, что это соответствие сохраняет упорядоченность. Точно так же каждую $f\in[\mathscr{S}\to\mathscr{S}]$ можно (грубо) аппроксимировать константой, скажем $f(\perp)$ (это самый «лучший» элемент, предшествующий в отношении $\subseteq$ всем значениям $f(x)$. Эту связь пространства и аппроксимации пространств будем обозначать $\mathscr{S}\lhd[\mathscr{S}\to\mathscr{S}]$.

Далее мы можем сказать, что

$$[\mathscr{S}\to\mathscr{S}]\lhd[[\mathscr{S}\to\mathscr{S}]\to\mathscr{S}],$$

но теперь по другой причине. Разобравшись с соотношением $\mathscr{S}\lhd[\mathscr{S}\to\mathscr{S}]$, рассмотрим теперь пространство более сложных структур вида $\mathscr{F}_n$. Для начала предположим $f\in[\mathscr{S}\to\mathscr{S}]$. Мы хотим отобразить f в следующее пространство, скажем $i(f)\in\in[\mathscr{S}\to\mathscr{S}]\to\mathscr{S}$. Для каждого элемента $g\in[\mathscr{S}\to\mathscr{S}]$ по определению будем считать, что $i(f)(g)\in\mathscr{S}$. Теперь для $g\in[\mathscr{S}\to\mathscr{S}]$ мы имеем обратную проекцию $j(g)=g(\perp)\in\mathscr{S}$. Если это наилучшая аппроксимация для g, которую мы можем получить в $\mathscr{S}$, мы обязаны определить

$$i(f)(g)=f(j(g)).$$

Тем самым мы получаем следующее отображение $i:\mathscr{F}_1\to\mathscr{F}_2$. Соответствующую проекцию $j:\mathscr{F}_2\to\mathscr{F}_1$ определим аналогичным способом:

$$j(\phi)(x)=\phi(i(x)),$$

где $\phi\in[\mathscr{S}\to\mathscr{S}]\to\mathscr{S}$, а $i(x)\in[\mathscr{S}\to\mathscr{S}]$ является константной функцией со значением, равным x. Используя те же рассуждения, определим $i:\mathscr{F}_2\to\mathscr{F}_3$ и $j:\mathscr{F}_3\to\mathscr{F}_4$ и так далее:

$$\mathscr{F}_0\lhd\mathscr{F}_1\lhd\mathscr{F}_2\lhd\ldots\lhd\mathscr{F}_n\lhd\mathscr{F}_{n+1}\lhd\ldots.$$

С учетом всего этого было бы жаль не перейти к пределу (на этот раз среди *пространств*), и это именно то, что я хотел бы вам сообщить. Что может следовать из утверждения, что существует пространство

$$\mathscr{F}_\infty=\lim_{n\to\infty}\mathscr{F}_n?$$

Так как $\mathscr{F}_{n+1}=[\mathscr{F}_n\to\mathscr{S}]$ соответствует одному шагу итерации, то естественно предположить

$$\mathscr{F}_\infty=[\mathscr{F}_\infty\to\mathscr{S}]$$

(по крайней мере с точностью до изоморфизма). Это на самом деле так, но я могу только обрисовать основу (и обоснован-

ность) этого изоморфизма. Вначале отдельные пространства $\mathscr{F}_n$ помещаются одно в другое, что не только создает башню пространств, но и трактует $f(x)$ как алгебраическую операцию от *двух* переменных. $\mathscr{F}_\infty$ является в точности результатом объединения $\mathscr{F}_n$; т. е. *внутри* этих областей мы можем представить башни *функций*, каждая из которых аппроксимирует следующую (с использованием отображений i и j), т. е. в $\mathscr{F}_\infty$ эти башни имеют свои пределы. В случае усеченных башен мы можем утверждать, что каждое пространство $\mathscr{F}_n \lhd \mathscr{F}_\infty$.

Итак, почему же $\mathscr{F}_\infty$ изоморфизм? Возьмем функцию (непрерывную) из $[\mathscr{F}_\infty \to \mathscr{S}]$. Так как эта функция непрерывна, то она определяется тем, как она воздействует на конечные уровни $\mathscr{F}_n$, т. е. она будет иметь все лучшую и лучшую аппроксимацию в $[\mathscr{F}_n \to \mathscr{S}] = \mathscr{F}_{n+1}$; таким образом, аппроксимации «находятся» в конечных уровнях $\mathscr{F}_\infty$. Их предел должен дать нам снова ту же функцию из $[\mathscr{F}_\infty \to \mathscr{S}]$, с которой мы начали. Аналогично *любой* элемент из $\mathscr{F}_\infty$ может рассматриваться как предел аппроксимирующих функций в пространствах $[\mathscr{F}_n \to \mathscr{S}]$. Возможно, необходимо проверить некоторые детали; однако при переходе к пределу нет действительного различия между $\mathscr{F}_\infty$ и $[\mathscr{F}_\infty \to \mathscr{S}]$: бесконечный уровень функций высших порядков является его *собственным* функциональным пространством. (Как всегда, это является следствием непрерывности.)

$$\begin{aligned} \mathscr{F}_\infty \times \mathscr{F}_\infty &= [\mathscr{F}_\infty \to \mathscr{S}] \times [\mathscr{F}_\infty \to \mathscr{S}] \\ &= [\mathscr{F}_\infty \to \mathscr{S} + \mathscr{S}] \\ &= [\mathscr{F}_\infty \to \mathscr{S}] \\ &= \mathscr{F}_\infty \end{aligned}$$

Рис. 3. Первая цепочка изоморфизмов.

Под описанным здесь скрывается гораздо большая структура, чем я предполагал вначале. Рисунок 3 иллюстрирует цепочку изоморфизмов, показывающих, что $\mathscr{F}$ получает многие из свойств $\mathscr{S}$, с которыми мы уже знакомы. Это верно по следующим соображениям. Во-первых, мы трактуем $\mathscr{F}_\infty$ как функциональное пространство. Затем *парам функций* можно изоморфно поставить в соответствие функции, принимающие *пары значений*. Однако, как мы уже знаем, $\mathscr{S} \times \mathscr{S} = \mathscr{S}$. Последний переход как раз отображает функции на $\mathscr{F}_\infty$ обратно в элементы из $\mathscr{F}_\infty$.

Используя изоморфизм на рис. 3, мы можем получить следующий результат, показанный на рис. 4. Рассуждения совершенно очевидны. Возьмем функцию, действующую из $\mathscr{F}_\infty$ в $\mathscr{F}_\infty$. Значения этой функции можно интерпретировать как функции.

$$\begin{aligned}\mathscr{F}_\infty \to \mathscr{F}_\infty &= [\mathscr{F}_\infty \to [\mathscr{F}_\infty \to \mathscr{P}]] \\ &= [[\mathscr{F}_\infty \times \mathscr{F}_\infty] \to \mathscr{P}] \\ &= [\mathscr{F}_\infty \to \mathscr{P}] \\ &= \mathscr{F}_\infty.\end{aligned}$$

Рис. 4. Вторая цепочка изоморфизмов.

Но учтем, что функция, значениями которой являются функции, есть в точности *функция от двух аргументов* (с точностью до изоморфизма). Как мы уже видели на рис. 3, $\mathscr{F}_\infty \times \mathscr{F}_\infty = \mathscr{F}_\infty$, поэтому мы получаем последний переход (с точностью до изоморфизма).

Мы обрисовали, почему $\mathscr{F}_\infty$, пространство функций бесконечного порядка, является моделью λ-исчисления. λ-исчисление — язык (здесь не рассматривавшийся), в котором каждый терм может *одновременно* обозначать *как* аргумент (значение), *так и* функцию. *Формальные* детали довольно просты, а *семантические* — как раз те, которые мы рассматривали: каждый элемент пространства $\mathscr{F}_\infty$ может в то же время быть элементом пространства $[\mathscr{F}_\infty \to \mathscr{F}_\infty]$; таким образом, $\mathscr{F}_\infty$ дает модель, которая является одной из многих.

Не претендуя на точность, мы в общих чертах обрисовали денотационную семантику для чисто процедурного языка (а также для пар и т. д., см. рис. 2). В перечисленных ниже работах по реальным языкам программирования указаны все остальные черты (присваивание, упорядочение, объявления и т. д.). В них установлено, что метод семантического определения действительно работает. Я надеюсь, что и вы заинтересуетесь им.

ЛИТЕРАТУРА

1. Bohm C., Ed. λ-Calculus and Computer Science Theory. Lecture Notes in Computer Science, Vol. 37. Springer-Verlag, New York, 1975.
2. Clark K. L., Cowell D. F. Programs, Machines and Computation. McGraw-Hill, New York, 1976.
3. Crossley J. N., Ed. Algebra and Logic. Papers from the 1974 Summer Res. Inst. Australian Math. Soc., Monash U. Clayton, Victoria, Australia.
4. Donahue J. E. Complementary Definitions of Programming Language Semantics. Lecture Notes in Computer Science, Vol. 42, Springer-Verlag, 1976.
5. Eilenberg S. Automata, Languages, and Machines. Academic Press, New York, 1974.
6. van Emden M. H., Kowalski R. A. The semantics of predicate logic as a programming language. J. ASM 23, 4 (Oct. 1976), 733—742.
7. Manes E. G., Ed. Category Theory Applied to Computation and Control. First Int. Symp. Lecture Notes in Computer Science, Vol. 25 Springer-Verlag, New York, 1976.

8. Manna Z. Mathematical Theory of Computation. McGraw-Hill, New York, 1974.
9. Milne R., Strachey C. A. Theory of Programming Language Semantics. Chapman and Hall, London, and Wiley, New York, 2 Vols., 1976.
10. Plotkin G. D. A powerdomain construction. SIAM J. Comptng. 5 (1976), 452—487.
11. Rabin M. O., Scott D. S. Finite automata and their decision problems. IBM J. Res. and Develop. 3 (1959), 114—125.
12. Scott D. S. Data types as lattices. SIAM J. Comptng. 5 (1976), 522—587.
13. Stoy J. E. Denotational Semantics — The Scott — Strachey Approach to Programming Language Theory. M. I. T. Press, Cambridge, Mass.
14. Tennent R. D. The denotational semantics of programming languages. Comm. ACM 19, 8 (Aug. 1976), 437—453.