Алгол 68 и его влияние на программирование в СССР и России

А.Н.Терехов, СПбГУ

a.terekhov@spbu.ru

История создания языка

В начале 60-ых годов XX века сложилась следующая ситуация с языками программирования: в Америке – безраздельно царствовал Фортран, в Европе все большую популярность получал язык Алгол 60 [1], кстати, в его разработке принимали участие и американские учёные. Этот язык был довольно стройный, имел более-менее точное описание синтаксиса, в нём были некоторые новые интересные языковые черты (прежде всего, - рекурсия, чего не было в Фортране). В то же время язык обладал массой недостатков, как технических (например, использование целых числе в качестве меток) так и, собственно, языковых (например, никак не был стандартизован ввод/вывод, не была поддержана обработка литер и строк, а это уже в то время было довольно важной частью программирования, не было сложных структур данных). Поэтому авторы языка продолжили работу и в 1964 году выпустили Пересмотренное сообщение [2]. Чтобы организационно оформить эти работы, международная федерация IFIP (International Federation for Information Processing) в 1962 г. создала Рабочую группу Working Group 2.1 по алголоподобным языкам. После выпуска Пересмотренного сообщения об Алголе 60 эта группа приступила к разработке планов следующих языков программирования – наследников Алгола 60. Была выпущена так называемая Белая книга, которая содержала несколько очень интересных статей. Например, статья Ральфа Лондона, одного из создателей языка Alphard – в котором были некоторые предпосылки для доказательств корректности программ. В статье Барбары Лисков «Язык CLU» впервые были сформулировано понятие абстрактных типов данных. Была статья голландского ученого ван Вейнгаардена о двухуровневых грамматиках. Дело в том, что контекстно-свободные грамматики, которыми традиционно пользовались в то время, обычно в нормальной форме Бэкуса-Наура (Бэкус – разработчик Фортрана, а Наур – главный редактор сообщения об Алголе 60), были удобными и используются, кстати, до сих пор, но все-таки недостаточно сильными и выразительными, например, нельзя было описать контекст использования конструкции. Грамматики ван Вейнгаардена имеют двухуровневую структуру и по выразительной мощности эквивалентны машине Тьюринга, т.е. в принципе, с помощью такой грамматики можно описать любой алгоритм.

На основе Белой книги, а именно, на основе предложения ван Вейнгаардена, было предложено создавать новый язык, существенно более точный, с более формализованным описанием не только синтаксиса, но и семантики. В результате, после многолетних дискуссий Рабочей группы (РГ) 2.1, в работе которой приняли участие множество известных ученых из Америки и Европы, в декабре 1968 года IFIP приняла сообщение о языке Алгол 68 [3]. Надо сказать, что этот язык в ту пору был очень тяжелым и трудным в понимании, поэтому РГ 2.1 продолжила свою работу, расширила состав авторов языка, и к 1973 году было подготовлено Пересмотренное сообщение об Алголе 68 [4], в котором на основе всех базовых идей исходного языка и грамматик ван Вейнгаардена был определен существенно более приемлемый язык, более простой в реализации.

Начало работ по Алголу 68 в СССР

В СССР первую информацию о работах по Алголу 68 привез член РГ 2.1, будущий академик, а тогда еще член-корреспондент АН СССР – Андрей Петрович Ершов из Академгородка (Новосибирск). Через него эта информация распространилась по стране, в том числе её получил мой научный руководитель доктор физ-мат наук Григорий Самуилович Цейтин, который в то время руководил лабораторией математической лингвистики НИИММ ЛГУ. Он принял активное участие в обсуждение языка, писал весьма дельные замечания авторам и удостоился благодарности в предисловии к публикации по Алголу 68. Понятно, что язык с новыми выразительными возможностями, с новым способом описания синтаксиса и семантики вызвал большой интерес среди программирующей общественности, стали формироваться группы, готовящиеся к его реализации. В первые годы, правда собственно до реализации дело не доходило. Например, группа математиков из Академгородка всерьез занялась переводом сообщения об Алголе 68 на русский язык. Понятно, что русское описание языка в стране, где много миллионов людей говорят по-русски, вещь важная. Я за этой работой только наблюдал и, первое время, не принимал участия. Честно скажу, мне казалось, что они слишком много времени тратят на выбор терминов, даже на обсуждение того, каким шрифтом какие термины должны быть опубликованы, а типографские возможности того времени были совсем не такими богатыми, как сейчас. Всё это вызывало массу трудностей, но это было как-то не интересно для меня.

Из обсуждений того времени запомнился тот факт, что в русскоязычной литературе не было укоренившегося слова, обозначающего файл. Предлагались различные варианты переводы этого важного понятия: «тека» (от слова библиотека), «фонд» и так далее. Я помню, как шутил на эту тему Андрей Берс из Академогородка, что в русском языке есть только два слова, заканчивающихся на «-айл» - кайло и хайло, потом он всегда немножко молчал и с улыбкой добавлял: «ну, конечно, есть еще и Задыхайло». Задыхайло был очень известным советским программистом, не знаю, обижался ли он на эту шутку или нет, но я запомнил, что «-айл» это не очень русское окончание. Тем не менее, после долгих обсуждений был принят термин файл, которым мы пользуемся до сих пор. Кстати говоря, таких терминов в то время было принято довольно много, именно в связи с Алголом 68. Например, «сборка мусора». Первоначально казалось, что это совершенно не технический, не научный термин. Но, с другой стороны, это прямой перевод англоязычного термина garbage collection, довольно точно отражающего суть дела (в процессе динамического распределения памяти возникают участки памяти, на которые уже никто не ссылается, т.е. реальный мусор). Поэтому механизм, который уплотняет память, оставляет в памяти только те участки, на которые есть реальные ссылки это, действительно, сборка мусора. В основном перевод был работой Александра Федоровича Рара. Когда оригинальное сообщение об Алголе 68 было опубликовано (в 1969 г., так как официальное принятие осуществилось только в декабре 1968 г.), важно отметить, что уже в том же 1969 году в журнале «Кибернетика» вышел перевод сообщения об Алголе 68 [5], т.е. все признавали, насколько важно иметь описание такого нового языка на русском языке.

Первые реализации Алгола 68 в СССР

У нас в Ленинграде Г.С. Цейтин собрал группу, состоящую из нескольких кандидатов наук и довольно большого количества студентов, которой предложил заниматься

реализацией Алгола 68. Дело было новое, опыта программирования трансляторов не было ни у кого из нас. Борис Константинович Мартыненко, который в то время был руководителем лаборатории системного программирования, провел 10 месяцев в Дании на стажировке у Наура и принял участие в реализации транслятора с Алгола 60 GIER – одного из самых известных трансляторов на тот период. Насколько я знаю, он занимался лексическим анализатором (сканером, говоря в современных терминах). Конечно, он знал и общее устройство транслятора и даже читал курс в Университете на мат-мехе, который я на третьем курсе посещал. Итак, группа была сформирована, мы активно изучали язык. Язык был действительно довольно трудным для понимания, но мы были молодыми, мы были математиками и нам казалось, что это вполне естественно разбираться в сложных вещах. Разобрались, причем разобрались настолько, что находили ошибки в описании, писали авторам языка, получали ответы и продумывали реализацию.

Первые полтора-два года эта работа носила чисто исследовательский характер, даже без возможностей для выхода на практическую реализацию транслятора. Так совпало, что примерно в то же самое время в СССР были развернуты работы по созданию системы ЕС ЭВМ. Это была полная копия серии IBM/360, руководство страны надеялось совершить такой «китайский скачок», разом догнав американцев по номенклатуре вычислительных машин и, особенно, по набору прикладных программ для них. Я не хочу сейчас вдаваться в обсуждение, хорошо это были или плохо. Лично я считаю, что это было ошибкой – у нас была довольно сильная школа советских создателей ЭВМ, были довольно интересные программы, в том числе и трансляторы с языков высокого уровня, например, для ЭВМ «Минск». Были, как минимум, три транслятора с Алгола 60, выполненные по руководством С.С.Лаврова в Подлипках (ТА-1М), А.П. Ершова в Академгородке (Альфа-транслятор) и М.Р. Шура-Буры в Москве (ТА-2).

Но решение было принято, машины серии ЕС начали проектировать, под это были отпущены определенные, довольно крупные финансовые ресурсы. В рамках процесса создания новых ЭВМ нам удалось получить финансирование для работ по отладочному транслятору с Алгола 68 для будущей ЕС ЭВМ (повторюсь, что ЕС ЭВМ в то время еще не было, она только проектировалась и разрабатывалась). Будучи студентом 5-го курса мат-меха я поехал в Москву для подписания договора с НИЦЭВТом. Так получилось, что старшие товарищи по разным причинам не смогли поехать, поэтому послали меня. Там я в первый раз встретился с А.П. Ершовым, который приехал ровно с той же целью – подписывать договор на создание транслятора с Алгола 68 для ЕС ЭВМ (проект Бета). На самом деле идея Ершова была существенно более глобальной – он хотел разработать семейство трансляторов с языков Алгол 68, PL/I, Симула 67 для разных машин, то есть реализовать идею UnCoL. Кстати, я только недавно узнал, что первые публикации по UnCoL (Universal Common Language) появились в Communications of ACM еще в 1958 году – идея при создании трансляторов с m языков на n ЭВМ изменить число компиляторов с m*n на m+n (сначала программы со всех входных языков переводятся на некий универсальный промежуточный язык, а потом из него делаются генераторы в коды разных ЭВМ). Эта идея довольно старая, но, насколько я знаю, в полном виде она до сих пор не была осуществлена. А М. Шварцман? http://ershov.iis.nsk.su/archive/eaindex.asp?lang=1&gid=1464

Ершов хотел заняться именно такой глобальной темой, и была достигнута договоренность между Ершовым и Цейтиным, что мы будем делать трансляторы со строго одинакового стандарта входного языка, но наш ленинградский транслятор будет играть роль отладочного, более-менее быстрого транслятора, а новосибирская система Бета будет глубоко оптимизирующим транслятором. Понятно, что наши коллеги из Академгородка хотели использовать опыт, который они накопили при проектировании и разработке транслятора Альфа с Алгола 60.

Итак, работа из чисто научно-исследовательской превратилась в хозяйственно-договорную со строгими этапами и сроками. Не всем участникам нашей группы это понравилось. Одно дело, когда ты спокойно работаешь, пишешь статьи, выступаешь на

конференциях и никто на тебя не давит, никто не напоминает, что завтра приедут заказчики и им надо предъявить такой-то отчет. Совсем другое дело – производственная работа. Однако, в те годы мы все это понимали не слишком сильно. Вольница была довольно большая, в Университете каждый занимался, чем хотел, и, кстати, научные результаты от этого не только не страдали, но, может быть, даже и выигрывали. Но даже само слово «плановость» было для нас чуждым в то время.

Так началась наша работа в тесном контакте с новосибирской группой. Эта работа приобрела и организационные формы.

Рабочая группа по Алголу 68

В рамках ГКНТ – Государственного комитета по науке и технике – была создана Рабочая группа, которую возглавил А.П. Ершов. Первое время его заместителем был Г.С. Цейтин. Примерно к 1975 году Цейтин во многом потерял интерес к этой сугубо практической работе, увлекся языками искусственного интеллекта, семантическими сетями и другими интересными вещами (например, языками проектирования поведения роботов) и постепенно стал отходить от работ по Алголу 68. Тогда Ершов назначил меня заместителем председателя Рабочей группы. Не скрою, это было очень лестно для меня, но это была не синекура – приходилось довольно много работать, и Ершов следил за тем, чтобы мы занимались существенно более широким кругом вопросов. Алгол 68 был в некоторой степени только поводом для встреч специалистов из разных городов, которые проходили 5-6 раз в год (чаще в Ленинграде и Академгородке, но и в других городах тоже, в Москве, Ростове, иногда в курортных местах, чтобы участникам был и дополнительный интерес отвлечься от основной работы и посетить заседание Рабочей группы).

Довольно часто Ершов давал членам Рабочей группы различные статьи на реферирование. Дело в том, что Ершов был одним из немногих советских специалистов, который реально мог выезжать за границу, посещать капиталистические страны – в те годы это было довольно трудно. Я помню, каким событием было, когда Цейтина отпустили на конференцию то ли в Румынию, то ли в Венгрию – даже такие поездки были в то время редкостью. А Ершов ездил, причем довольно свободно, у него были широкие связи в среде научной общественности, многие присылали ему свои статьи как уважаемому человеку. Ершов стремился, чтобы эти статьи не просто лежали на полке, раздавал их членам рабочей группы, мы их читали, а потом делали доклады на заседаниях рабочей группы и, главное, обсуждали эти статьи. Например, мне как-то раз Ершов поручил прочитать (и выступить с докладом по прочитанному материалу) статью С. Джонсона «Yet another compiler compiler» [6]. Сейчас все знают, что такое Yacc, то есть компилятор компиляторов – компилятор, который на входе получает грамматику языка, а на выходе – компилятор (на самом деле, только анализатор) с этого языка. Но в те годы это все было в новинку, и мне было очень интересно это прочитать. Или, например, статья Вулфа про оптимизации [7]. Он сделал оптимизирующий транслятор: вначале был придуман некий универсальный промежуточный язык (ПЯ), и любая оптимизация не меняла структуры текста на ПЯ, только добавлялись параметры в узлы графов, представляющих операторы программы. Была выполнена серия забавных экспериментов: было задано несколько оптимизаций, которые выполнялись в разной последовательности. Оказалось, что некоторые оптимизации надо выполнять несколько раз, так как применение одной оптимизации дает материал для других. Последовательность оптимизаций также влияет на качество кода – это была интересная статья, я ее изучил и потом рассказал о ней на заседании Рабочей группы.

Также на заседаниях обсуждались различные учебные материалы по Алголу 68, сравнивались различные реализации. В СССР появилось довольно много групп, реализующих Алгол 68. Попробую перечислить эти группы (работа шла в 70-ые годы XX века). Очень интересная группа работала в Киеве (научный руководитель – Екатерина Логвиновна Ющенко, среди авторов были Штетельмен, Штейнбук, Макогон). Эта группа

реализовывала не столько транслятор, сколько некоторую базу данных, их интересовали Persistent Data (сохраняемые данные). Если пользователь заканчивал работу, он мог нажать специальную кнопку, и все данные, созданные в процессе сеанса, запоминались в некоторой базе данных, а в следующий раз можно было начать с прерванного места. Тогда это была довольно большая новинка и интересная работа, но для нас, для Рабочей группы по Алголу 68 это было отчасти посторонней работой. Например, группа в Киеве даже не перешла на Пересмотренное сообщение об Алголе 68, а продолжала работать в соответствии с первым Сообщением, которым позже уже никто не пользовался. Тем не менее, эта работа была интересна.

Мне всегда очень нравилась работа московского ЦЭМИ (Центральный экономико-математический институт) под руководством Михаила Рувимовича Левинсона. Они разрабатывали транслятор с Алгола 68 для DEC-вской архитектуры PDP11 – тогда были советские копии СМ3, СМ4, СМ1420, и для этих машин москвичи сделали транслятор. Они предложили интересную идею отказаться от ограничений на область действия объектов, все данные размещать в куче, это давало возможность перейти к функциональным языкам (частичная параметризация, функции, выдаваемые в качестве результата вызовов других функций, и так далее). Тогда на первый план выходит, конечно, сборка мусора, и она занимает довольно большой процент времени счета, но, тем не менее, Левинсону и его коллегам удалось преодолеть эти трудности, и транслятор заработал. Очень интересная пионерская работа!

Одной из наиболее удачных реализаций являлась работа другой московской группы под руководством Александра Николаевича Маслова, куда входили Валерий Броль, Владимир Гущин и Владимир Яковлев. Эти молодые люди реализовали транслятор с Алгола 68 для новой советской машины Эльбрус. Эльбрус мы все очень любили, поскольку это была одна из немногих в то время оригинальных советских машин. Злые языки, конечно, говорили, что Эльбрус чем-то похож на Burroughs, но, поскольку я хорошо знаю архитектуру этих двух машин, то считаю, что это именно злые языки – так как многие идеи Эльбруса были абсолютно оригинальными. Для этой хорошей машины был сделан транслятор с более-менее полного языка Алгола 68, и я даже участвовал в некой комиссии по приемке этого транслятора на Эльбрусе.

Надо сказать, что этой группе было легче работать, так как Эльбрус – это high-level language (HLL)-computer, то есть компьютер, ориентированный на языки высокого уровня, и поэтому многие вещи, которые, например, нам на ЕС ЭВМ приходилось реализовывать многими командами (вырезки, циклы, вызовы процедур), в Эльбрусе реализовывались существенно проще – для всех этих конструкций были специальные машинные команды.

Еще пару слов про Академгородок и группу Ершова. Первоначально входными языками в проекте Бета были Алгол 68, Симула 67 и PL/1, а целевыми машинами были ЕС ЭВМ и БЭСМ 6. Эта группа вела огромное количество исследований, были очень интересные публикации, интересные результаты, но до практического воплощения дошли только трансляторы для БЭСМ-6 с Симулы 67 и Паскаля (Георгий Степанов и Сергей Покровский) и транслятор с Модулы 2, который выполнил Леонид Захаров. Называлось еще подмножество языка Ада, который реализовал Сергей Тен, но у меня об этом трансляторе мало информации.

Транслятор А68 ЛГУ

Итак, мы реализовали транслятор с Алгола 68 на ЕС ЭВМ. Поскольку ЕС ЭВМ появилась только в 1974, а реально работающей машиной стала в 1975, то нам пришлось некоторое время работать на других машинах. Начали мы с польской ЭВМ Одра 1204, там был довольно хороший транслятор с Алгола 60, поэтому анализирующую часть транслятора с Алгола 68 мы написали на Алголе 60 и отлаживали на Одре 1204. Попутно пришлось

решать довольно много новых задач в технике трансляции, например, даже структура компилятора оказалась нестандартной, нетрадиционной. В Алголе 68 можно описывать новые типы и операции над ними. Так вот, если написано **m** a; , то это может быть описанием переменой a, если **m** – тип, а может быть унарной операцией, если **m** – операция. Мы этого не поймем, пока не идентифицируем **m**. Но, чтобы идентифицировать **m**, нам надо составить таблицу всех описаний с учетом блочной структуры, а это невозможно сделать без синтаксического анализа. Получается некий замкнутый круг.

У нас сразу было принято разделение труда – все конструкции были поделены между участниками группы (например, Цейтин взял себе вызов процедур и работал с параллельными предложениями. Это, действительно, самые трудные куски Алгола 68). Структурой компилятора в целом никто не занимался. Этим стал заниматься я и придумал некую схему из шести просмотров, которые чередуются (прямой, обратный) и в каждой прямой фазе собирается некоторая информация к концу конструкции, а на обратном просмотре эта информация передается к началу конструкции. Я предложил выполнить сначала некий первичный, более-менее примитивный синтаксический анализ только с целью проидентифицировать индикаторы видов и операций. Потом выполняется полный видовой анализ и генерация кода. В те годы даже опубликовать эти работы было практически негде – не было журналов по программированию. В 1975 году был образован академический журнал «Программирование», который издается до сих пор. Это один из немногих отечественных журналов по нашей специальности, который индексируется в Web of Science и Scopus. Буквально во втором номере этого журнала опубликована моя статья “Процессы идентификации и структура компилятора с языка Алгол 68” [8].

Еще раз повторю, что на Одре работал только анализатор, генератор кода мы стали писать на языке макрогенератора ассемблера будущей ЕС ЭВМ, поскольку мы получили доступ к ее прообразу – к оригинальной американской машине IBM/360 – уж не знаю, какими путями удалось купить эти две машины (тогда были жесткие ограничения на поставку в СССР высокотехнологичной продукции), я даже помню их названия – Озон и Лотос. Они были установлены в НИЦЭВТЕ, головной организации, занимающейся разработкой ЕС ЭВМ, в Москве на Варшавском шоссе. Мы ездили туда еженедельно на 2-3 ночи. Конечно, днем на машинах работали москвичи, а ночное время давали нам, приезжим. Вначале это были чисто американские машины, затем рядом с каждым устройством стали ставить его аналог (рядом с оригинальным дисководом – немецкий, рядом с оригинальным устройством печати – болгарский аналог, что-то было из Чехословакии). В первое время эти аналоги работали довольно плохо, а так как мы работали ночами, за нами особенно не присматривали, мы научились быстро передергивать кабельные разъемы и работали только на оригинальных американских устройствах, чтобы особо не страдать. Конечно, это не приветствовалось, зато мы быстрее работали. То же можно сказать и про операционную систему. Пока коллеги из НИЦЭВТа прочитают версию программы операционной системы, пока разберутся, пока найдутся какие-то описания, уже выходила другая версия. В конце концов, в одном из американских университетов была разработана вообще другая операционная система VM (Virtual Machine). IBM купила эту систему у университета, доработала, сделала ее существенно более широкой по функциональности, но во много раз хуже по эффективности. Поэтому все передавали друг другу магнитную ленту с оригинальной операционной системой от университета, а не официальную IBM-овскую версию, которая была рекомендована компаний и НИЦЭВТом.

Это было тяжелое время, на протяжении многих и многих недель надо было каждую неделю ночами работать на компьютере, это было очень непросто, и все мои старшие товарищи отошли от этой работы: еженедельные поездки в Москву с тяжелейшими чемоданами, набитыми колодами перфокарт (магнитные ленты появились и стали реально использоваться намного позже), проживание в гостинице, ночная работа отнимали много сил. Работать на новой технике, когда и спросить было не у кого при возникновении проблем, было и трудно, и интересно.

В результате такой тяжелой работы мы сделали генерирующую часть из промежуточного языка в коды ЕС ЭВМ, всё как-то заработало. К этому времени у нас на мат-мехе уже появилась ЕС ЭВМ, причем это была первая машина такого класса (ЕС1030), поставленная в открытой, а не в военной организации – об этом писали даже ленинградские газеты.

Наконец, мы решили перенести весь транслятор на ЕС ЭВМ, применив метод раскрутки – переписали генерирующую часть на Алголе 68 и транслировали каждую из нескольких сот процедур сначала на Одре 1204, получали текст на ПЯ в виде перфоленты, которую потом несли на ЕС ЭВМ, вводили (это был текст на макроязыке) и транслировали с макроассемблера в коды ЭВМ. Эта работа заняла около года, в это время было найдено много ошибок в трансляторе – ведь это же шикарный, огромный тест на Алголе 68. Попутно выяснилось, что многие конструкции мы реализовали не слишком удачно.

Например, мне запомнилась борьба за эффективность вызова. Первая реализация требовала 6 команд в порождении кода и 23 команды в подпрограмме. Работало это довольно медленно, мы предприняли большое количество усилий, чтобы ускорить процесс. Главный прорыв произошел тогда, когда мы, зная уже устройство UNIX, пустили динамическое распределение памяти справа налево, так что базовый регистр стал указывать и на вершину стека, и на статику текущей процедуры. В этих условиях нам удалось сделать так, чтобы на вызов приходилось полторы команды – одна четырехбайтовая и одна двухбайтовая команда и всего 11 команд в подпрограмме. В те годы, когда аналогичный транслятор с PL/I F занимал порядка 150 команд на каждый вызов, наш результат я оцениваю довольно высоко.

Как только закончилась первая раскрутка, мы тут же приступили ко второй раскрутке, продолжая улучшать качество кода, уменьшая время трансляции и так далее. В результате, после трех раскруток, получился довольно приемлемый транслятор, которым начали пользоваться программисты, в первую очередь, из военных организаций, потому что именно там требовалась особо высокая надежность.

Влияние Алгола 68 на другие языки

Как известно, Алгол 68 мирового признания не получил. В академических кругах им пользовались, было реализовано несколько трансляторов в разных странах. Возможно, это связано с тем, что Алгол 68 действительно был тяжеловат для широкой публики, он был рафинированным языком для Академии (так на Западе называют сотрудников университетов и исследовательских центров). Возможно, тому есть и другие причины. Появился Паскаль, который занял нишу простых языков. Никлаус Вирт на симпозиуме, посвященном его собственному восьмидесятилетию, который я недавно посетил (кстати, сам Никлаус жив-здоров и до сих пор активно работает, его и сейчас весьма интересно слушать, его идеи до сих пор продуктивны), рассказал историю языков, как он её видит. Он был членом Рабочей группы 2.1 и только позже он вышел из нее вместе с Хоаром и Дейкстрой, подписав так называемый Minority Report (Мнение меньшинства). Позже он создал язык Паскаль как ответ на вызов Алгола 68 – создал выразительный, но простой язык.

Как обычно, эта простота не далась даром. В нашей ленинградской группе, как только Паскаль появился, мы параллельно с нашими работами по Алголу 68 приступили к работам по созданию и транслятора с Паскаля (руководил этими работами Аркадий Попов). Мы сразу же наткнулись на кучу несоответствий. В стремлении упростить язык Вирт не обратил внимания на очень многие вещи. Например, Паскаль – язык с более-менее строгой типизацией. Целой переменной присвоить вещественное значение нельзя. Но, если процедура передается параметром другой процедуре, то корректность ее параметров уже никак не проверяется. Это такая дырка в описании.

Еще один пример – было опубликовано описание Паскаля на русском языке под редакцией известного программиста Д.Б. Подшивалова из Москвы. На слова: «Паскаль

существенно превосходит по мощности Алгол 60» была сноска редактора: «С этим утверждением трудно согласиться, например, на Паскале трудно написать универсальную программу умножения матриц», так как в Паскале границы массива входят в его тип, так что, формально говоря, нужна своя процедура умножения матриц для массивов разных размеров [9].

Тем не менее, влияние Алгола 68, конечно же, очевидно, если смотреть на набор конструкций Паскаля.

На мой взгляд, еще более ярким примером преемственности к Алголу 68 является язык С, опубликованный в 1972 году. В С, как и в Алголе 68, возможна выдача значения условным выражением, присваиванием, последовательностью операторов, а операция с накоплением a+=b – это же явно из Алгола 68 взято. Последовательные предложения, когда (a=op1, b=op2, выражение)+1 – это же чистая калька последовательного предложения из Алгола 68, только вместо «;» ставится «,». Эта конструкция очень удобна и используется, скажем, для передачи параметров когда в сложном выражении нужно попутно запомнить какие-то промежуточные результаты.

Не могу закончить эту часть доклада, не вспомнив, что мне было довольно трудно внедрять Алгол 68 в СССР. Времена были такие, что, если не было какого-то западного аналога, то никто и говорить не хотел. Когда в США появился язык Ада, быстро ставший стандартом Министерства Обороны США, этот язык был очень далек от Фортрана и PL/1, но очень похож на Алгол 68 по мощи, по выразительной силе, по полной типизации и так далее. Появление языка Ада в Америке помогло мне с внедрением Алгола 68 в СССР. Я стал говорить: «Смотрите, а Америке появился язык с полной типизацией, вопросы надежности ПО выходят на первый план», и этот аргумент мне очень помогал при внедрении Алгола 68, особенно, в военных организациях СССР.

Специализированные языки на базе Алгола 68

Мы далеко не сразу осознали, какую мощную новую функцию имеет Алгол 68. Дело в том, что в этом языке впервые появилась возможность описывать новые типы данных и операции над ними. Идентификатор вида или операции надо было выделить жирным шрифтом или подчеркнуть.

mode точка = struct (**real** x, y);

- это описание точки, а описание прямой:

mode прямая = **struct** (**точка** a, b);

После того, как мы описали виды, ими можно пользоваться наравне с **int**, **real** и так далее. Более того, можно описать новые операции над ними. Например, операцию **Пересечение**, где операндами выступали бы прямые, а результатом – точка или исключительное значение в случае, когда прямые параллельны. Можно переписывать даже стандартные знаки операции, что очень удобно: +, -, *, /. Таким образом, можно спокойно описать сложение или умножение матриц – любые действия с любыми типами данных.

В сочетании со средствами накопления контекста [10] появилась идея делать специализированные языки. Например, мы делали специализированные языки для бухгалтеров, и они годами работали и даже не подозревали, что они работают на Алголе 68. То же самое мы делали для матфизиков, причем отдельно для «прочнистов», отдельно для «упругистов».

Одной из первых была работа моей дипломницы Наташи Витт из города Лыткарино, там находится огромный оптический завод, ориентированный в первую очередь на космос и на оборонку, но выпускает и гражданскую продукцию – линзы, зеркала, объективы. По их заказу мы разработали специализированный язык планиметрии, именно в этом языке были определены точки, прямые, окружности и другие геометрические фигуры и операции отражения и преломления. В результате, появилась возможность очень компактно и

эффективно описывать сложные структуры, состоящие из многих зеркал и линз. Насколько я знаю, этот язык очень интенсивно использовался на заводе.

Раз уж мы говорим о влиянии Алгола 68 на программирование, могу сказать, что я и сегодня занимаюсь ровно тем же самым. Мы разрабатываем метатехнологию, с помощью которой можно разрабатывать (и быстро реализовывать) графические редакторы, генераторы кода, отладчики и так далее для различных специализированных языков [11]. Сегодня мы делаем это на базе других инструментальных средств и других технологий, но идея-то та же самая! Мы уже тогда убедились, что специализированные языки намного эффективнее в использовании, чем универсальные языки, а Алгол 68 обладал такими свойствами, что можно было определить специализированный язык, но пользоваться при этом стандартным транслятором с Алгола 68. Повторю, многие пользователи годами использовали специализированный язык и не догадывались, что они пишут на Алголе 68. Мне вспоминается цитата из Мольера: «Месье Журден и не подозревал, что всю жизнь говорил прозой». Мне эта цитата нравится тем, что очень точно отображает ситуацию с использованием DSL – Domain Specific Languages.

Я бы хотел еще раз подчеркнуть, что если вы описали какой-то набор видов данных и какой-то набор операций над ними, то при этом использование (даже в раздельно транслированных процедурах) абсолютно надежно и эффективно, как-будто вы все эти описания скопировали, как это было в копибуках в Коболе, но при этом все это сохраняется в таблицах, и трансляция происходит очень быстро. Это было удачной чертой, которой мы с удовольствием пользовались. Одним из применений этого подхода (нашим коллективом) была следующая работа – мы разработали специализированный язык А68К – от слова «коммутация». Мы выделили некоторое подмножество языка, очень эффективно реализованное и необходимое для систем реального времени, и дополнили его операциями, необходимыми для описания систем реального времени, связанными с таймерами, посылкой/приемом сообщений и так далее. Этот язык более десяти лет активно применялся при разработке разных телефонных станций и других средств телекоммуникации.

Внедрение в промышленность

Когда мы разработали транслятор с Алгола 68, естественно встал вопрос, как его внедрять в промышленность. Университетские люди (из разных университетов) использовали Алгол 68 очень охотно – новый язык, новые возможности. Но мы понимали, что очень важно внедрить его в реальную промышленность. И вновь так совпало, что в начале 80-ых годов (так же как и совпадение начала работ по реализации Алгола 68 с разработкой ЕС ЭВМ), нас почти силой заставили заниматься военными разработками. В начале 80-ых годов военные испытывали довольно много проблем при переходе к цифровой вычислительной технике (до этого использовались либо аналоговые машины, либо вообще не было никакой электроники). При разработке информационных систем, систем управления, основных военных приложений ЭВМ, оказалось, что никакими приказами, драконовскими порядками не заставить людей разработать и довести до промышленной эксплуатации и сопровождения систему, в которой, скажем, миллион строк кода.

Мы стали внедрять Алгол 68 как некую панацею, поскольку мы искренне думали, что этот язык очень хорошо защищен от ошибок, что если ты попробуешь сложить, грубо говоря, яблоки с коровами, то эта ошибка будет найдена или, если ты передашь неправильное число параметров в процедуру, это тоже будет обнаружено, или, если ты напишешь A[i], и i выйдет за границы массива, это также будет обнаружено и так далее.

Жизнь показала, что мы начали не совсем с того, что было нужно. Мы начали с каких-то простых общетехнологических принципов (документация на машинных носителях, кросс-трансляторы, редакторы связей и так далее). Мы сделали около двух десятков различных кросс-трансляторов для различных диалектов языка Алгол 68, для разных целевых, часто, секретных ЭВМ. Мы разработали язык А68К. В каком-то смысле, с точки зрения

университетского специалиста, это был шаг назад (язык был существенно проще Алгола 68). Но с другой стороны, это был узнаваемый Алгол 68 со всеми его основными особенностями, контролем, с условными выражениями, последовательными выражениями, с операциями накопления и так далее, с очень хорошей защищенностью от ошибок пользователя. Но при этом, за счет того, что мы отказались от некоторых дорогостоящих конструкций типа гибких массивов, которые могут менять свои размеры прямо во время счета, мы добились весьма эффективной реализации, ввели специальные типы данных и операции, например таймер или операторы посылки сообщений.

Этот А68К стал активно использоваться не только нашим коллективом. Нам удалось внедрить это средство в самые разные организации (в основном, военного толка). Мы получили огромную пользу от этого, поскольку была широкая обратная связь: люди находили какие-то ошибки, делали предложения по улучшениям, в том числе, пользовательского интерфейса. Много нареканий вызывала система сигнализации об ошибках, мы потратили много сил, чтобы ее улучшить. В это самое время как раз появились первые персональные ЭВМ, и мы методами раскрутки перенесли А68ЛГУ на них. Чуть позже перенесли А68ЛГУ на DEC-овскую архитектуру (СМ3, СМ4). Мы уже привыкли работать на достаточно больших машинах, каковыми были ЕС ЭВМ, и, когда мы фактически сделали шаг назад и вернулись к работе на миниЭВМ, то оказалось, что то, что хорошо выглядит на ЕС ЭВМ, плохо выглядит на СМ4.

Приведу пример. Синтаксический анализ на СМ4 занимал для обычной программы 5 минут, а мы уже привыкли, что он занимает какие-то доли секунды. Мы стали исследовать этот вопрос. Оказалось, что когда-то мы применили автоматную модель для сравнения видов Алгола 68, а поскольку есть рекурсивные виды (цепные списки и юнионы), то структура видов довольно сложная. Когда мы нашли статью Ярослава Крала из Пражского Карлового университета, в которой говорилось, что можно вместо деревьев, где все алгоритмы экспоненциальны, использовать автоматные модели, где сложность сравнения автоматов или минимизации порядка n*log n, где n – количество состояний, мы обрадовались и перешли на автоматы. Но оказалось, что этот метод не очень-то и хороший – он требует сравнения всех описаний, какие только есть, включая и операции из стандартного вступления, в котором порядка тысячи описаний. Именно это и замедляло работу. Тогда мы сделали ужасный поступок: вернулись к алгоритмам, которые были реализованы еще в моей дипломной работе. Они были с экспоненциальной оценкой сложности, но эта экспонента реально возникала, только когда использовались сложные вложенные юнионы, что на практике очень редко, фактически никогда, не встречалось. Когда мы вернулись к потенциально экспоненциально сложному алгоритму, программы стали работать намного быстрее.

Об этом я часто рассказываю студентам, как о примере того, что не всегда асимптотическая оценка играет такую практически важную роль в реальной жизни. Кстати, с тех пор я встретился с еще несколькими примерами того же сорта. Но этот мне запомнился лучше всего – поскольку он был первым и, главное, казалось, что мы делаем абсурдную вещь: переходим от оценки сложности n*log n к экспоненциальной сложности, а все начинает работать быстрее. Понятно, что здесь никакого нарушения теории нет, просто эти оценки даются с точностью до некоторой константы, и когда эту константу удается, например, в сто раз уменьшить, то во многих практических случаях это оказывает решающую роль, даже большую, чем основная оценка.

Внедрение в промышленность шло болезненно, но все-таки шло. Очень помогала идея специализированных языков. Как только мы пытались объяснять людям с крупных промышленных предприятий Алгол 68 в оригинальном виде – мы встречали полное непонимание. Когда же мы забыли об Алголе 68 и стали рассказывать только о А68К, хотя всегда при этом мы пользовались нашим стандартным транслятором с полного Алгола 68, то это намного легче воспринималось. Это дало мне пищу для размышлений, и я до сих пор считаю, что широкие массы программистов требуют выразительных, но относительно простых средств, а академические изыски им ни к чему.

Аппаратная реализация

Мы сделали больше десятка трансляторов и кросс-трансляторов для специализированных военных ЭВМ, каждая из которых была хуже другой. Дело в том, что для всех этих специализированных ЭВМ, которые проектировались инженерами и для инженеров, формулировались требования типа малого энергопотребления, надежности, возможности работы в кунге, едущем по бездорожью. При этом разработчики забывали такое простое требование, что для этих машин придется еще и программировать. В конце концов мы решили, что спасение утопающих – дело рук самих утопающих и решили сделать свою машину. Кое-что мы на эту тему уже знали: надо было разработать машину, ориентированную на языки высокого уровня – HLL-компьютер (high level language). Мы уже знали про не очень успешный опыт машины Мир с языком Аналитик, где аппаратная интерпретация стоила довольно дорого и препятствовала массовому использованию. Знали о разработке компании Intel - iAPX432, которая по тактовой частоте и другим технологическим возможностям должна была давать порядка 10 миллионов операций в секунду, а давала только 100 тысяч операций в секунду, поскольку там было 7 типов адресации, и при каждом действии динамически проверялись все эти типы адресации. Знали мы и о машине Эльбрус с ее аппаратной реализацией алгоритмических языков.

Не всем нам нравился тот факт, что Эльбрус был большой и с водяным охлаждением. Понимаете, у Эльбруса была своя цель, эта машина использовалась для сил ПВО страны или чего-то еще в этом роде. Перед разработчиками Эльбруса стояла одна задача – сделать советскую суперЭВМ, превосходящую по возможностям суперЭВМ потенциальных противников. Когда мы в награду за многочисленные разработки для Эльбруса получили в фондах ЦК КПСС направление на Эльбрус, т.е. Эльбрус должны были установить на мат-мехе ЛГУ в Петергофе, это была бы точно единственная такая машина в открытой организации. Мы очень радовались и гордились этим фактом. Но когда наши мат-меховские инженеры поездили на завод счетно-аналитических машин (САМ) в г. Москве, познакомились поближе с Эльбрусом, узнали требования по влажности воздуха, теплоотводу, энергопотреблению, наличию градирни (специальный бассейн для охлаждения воды), они сказали, что сопровождать Эльбрус в условиях университета они не смогут. Если у тебя есть полк солдат, который следит за всеми приборами, наводит чистоту, то машина сможет выдавать свои великолепные характеристики. Если полка солдат нет, то машина работать не будет. Это произвело на меня удручающее впечатление, поскольку мне очень нравился Эльбрус и совсем не нравилась «цельносоздранная» (по выражению С.С.Лаврова) машина ЕС 1060. Но что делать? От Эльбруса пришлось отказаться, хотя нам уже пришло оборудование примерно на миллион рублей, мы уже подготовили зал и построили ту самую градирню.

Когда мы решили, что нам хватит делать кросс-трансляторы для совершенно дурацких специализированых вычислительных машин, мы начали думать, что же надо предпринять, чтобы сделать HLL-компьютер, но не такой дорогой как Эльбрус (пусть и не такой быстрый), при этом более совершенный и эффективный чем Intel APX 432. Как и во многих других случаях, решение подсказал Алгол 68. После стольких лет работы над разными трансляторами с Алгола 68 для разных ЭВМ мы уже четко понимали, что главное преимущество Алгола 68 в его полном видовом контроле периода компиляции. В каждой точке программы компилятор точно знает виды обрабатываемых значений. Именно эту идею мы использовали при разработке своей вычислительной машины, которую мы назвали Самсон [12], разработали её, получили на нее три авторских свидетельства. Самсон при тактовой частоте 3.25 МГц выигрывал по скорости счета на реальных программах не менее чем в 3 раза(!) у IBM PC AT с тактовой частотой 16 МГц. Таким образом, архитектурный выигрыш был не менее чем 15.

Главным ограничением УВК Самсон (Управляющий Вычислительный Комплекс, это была не универсальная машина, а ЭВМ для специальных применений, прежде всего для систем управления) было обязательное требование, что входные программы пишутся только на статических языках высокого уровня типа Алгол 68, Модула 2, Ада, Chill (был такой стандартный язык международного комитета по телеграфии и телефонии), но не на языках типа Фортрана, PL/1 или даже С, в котором в те времена соответствие параметров (не только их тип, но и даже количество) никак не проверялось.

На этой идее, что если абсолютное большинство проверок можно вынести на этап компиляции, а во время счета все, что проверил транслятор, уже не перепроверять, удалось сделать малогабаритную машину с высокой эффективностью. Позже наши военные заказчики потребовали сделать также и транслятор с языка С, чтобы обеспечить совместимость с теми программами, с которыми они работали раньше. Мы его, конечно, реализовали, но взяли подписку с заказчиков, что к нам претензий в случае аварийных ситуаций никто предъявлять не будет. Как говорится, кто предупрежден – тот вооружен. Наши заказчики это осознали и довольно часто проектирование программ велось на Алголе 68, а если какие-то подпрограммы использовались, ранее написанные на языке С, то при этом предпринимались все мыслимые методы и способы защиты: проверки аргументов, проверки результатов и так далее, чтобы уберечься от ненадежности языка С.

Первую реализацию УВК Самсон мы провели в основном за деньги наших болгарских друзей из города Пловдив, это готовилось как подарок к 70-летию советской власти. 5 ноября 1987 года состоялась демонстрация, на которой присутствовали члены Политбюро ЦК КПСС и Компартии Болгарии. За полчаса до начала визита высоких гостей еще ничего не работало, но как известно, я везунчик. К двум часам дня машина заработала. Фактически, она просто прошла несколько тестов, но члены Политбюро ничего другого и не спрашивали. Это был запоминающийся момент, после этого мы стали массово гонять разнообразные тесты, улучшать архитектуру, изменять систему команд с целью ее оптимизации и так далее – это уже длинная и скучная история. Начальная же идея – опыт Алгола 68 помог нам сделать ЭВМ, отличающуюся, причем довольно сильно, от других своей архитектурой.

В 1992 году УВК Самсон была принята на вооружение. После этого я на долгий срок потерял эту машину из виду, тем более я и сам не хотел влезать в какие-то особо секретные работы. Иногда от моих бывших учеников или бывших сотрудников доходила какая-то информация об использовании Самсона. Совсем недавно я узнал, что Самсон до сих пор выпускается, причем в довольно большом количестве экземпляров.

Техники трансляции Алгола 68, используемые до сих пор

В 1976 году, когда мы завершили работу над первой версией транслятора с Алгола 68 на ЕС ЭВМ, мы решили, что наши находки (а их было довольно много) надо собрать в одном месте. По инициативе Г.С. Цейтина мы написали монографию «Алгол 68. Методы реализации» [13]. У этой монографии довольно много авторов, но в предисловии аккуратно расписано, кто что писал, и из этого предисловия видно, что моя фамилия встречается там чаще других.

Не все из того, что мы тогда написали в этой монографии, выдержало проверку временем. Методы синтаксического анализа, которые там были предложены Мартыненко и Гиндышем, мы сейчас не используем. Сами эти методы хорошие, но там была довольно неудачно реализованная сигнализация об ошибках. В результате, после нареканий первых пользователей, мы довольно существенно переписали видонезависимый анализ, с тем, чтобы обеспечить более разумную сигнализацию об ошибках пользователей. А вот методы видозависимого анализа, идентификации мы применяем и до сих пор. В частности, именно потому, что ЭВМ того времени не были такими быстрыми и мощными, как сейчас, и их память была в тысячи раз меньше, чем сейчас, мы применяли различные оптимизации. Для того чтобы при поиске описания по применению не перебирать все строчки таблицы

идентификаторов, мы ввели такую технику, что в таблице представлений у нас всегда была ссылка на текущее описание. Описание находилось буквально за одно обращение к памяти. В начале и конце блока все эти ссылки разом подменялись, что существенно ускоряло работу этого механизма (так как описаний во много раз меньше, чем применений) по сравнению с традиционным перебором.

Или, например, метод динамического распределения памяти. Моя дипломница Людмила Григорьева в своей дипломной работе сравнила различные методы динамического распределения памяти (первый подходящий, наиболее подходящий, метод граничных признаков, метод близнецов) и провела моделирование на датчике случайных чисел. Оказалось, что лучшим является метод с граничными признаками, поскольку в этом методе память не только дробится, но и укрупняется за счет склеивания рядом стоящих участков. При довольно разумных ограничениях этот метод работает практически бесконечно, не требуя сборки мусора. Теперь во всех разработках, которые мы делаем, мы используем именно этот метод, настолько удачным было это решение.

Мы практически не используем тех методов отладки, которые были описаны в монографии «Алгол 68. Методы реализации», зато практически полностью сохранили технику синтеза, в частности, мне удалось придумать понятие сменяемой переменной, которую мы широко использовали вместо стеков в виде массива. Многие мои коллеги говорили, что это «обман трудящихся», что стек не может быть без массива, так как мы никогда не можем оценить его глубину. Каждый раз приходилось объяснять, что стек существует, но это стек рекурсивных вызовов процедур синтеза, который создается не для этой специальной цели, а просто так устроен синтез, что для каждой конструкции вызывается процедура генерации кода этой конструкции, причем, поскольку конструкции могут быть глубоко вложены, то процедуры, естественно, получаются рекурсивными. Сменяемые переменные давали нам возможность использовать верхушку стека как простую переменную, а не как вырезку элементов массива, что, конечно, было существенно эффективней, да и текстуально читабельней и наглядней.

Г.С. Цейтин предложил технику запросов и ответов (позже я нашел статьи Бронкара и Леви, где похожая техника также была описана, но в худшем виде), когда внешняя конструкция заказывает, как разместить значения внутренней конструкции (свободный запрос, на регистр, в определенное место памяти, уничтожить значение, если это просто оператор). Эта техника оказалась чрезвычайно удобной, и мы пользуемся ей и сейчас.

Конечно, инструменты раздельной трансляции с накоплением контекста в виде таблиц мы используем и сейчас во всех похожих случаях.

Для эффективной генерации кода Цейтин предложил разработать механизм работы с предсказателями (это стало темой моей диссертационной работы). Предположим, что у вас возникло некоторое анонимное, не имеющее идентификатора, рабочее значение на регистре. Если это значение долежит на регистре до момента использования – все хорошо, это самый эффективный случай, но если регистров не хватит, придется выгружать это значение в память. Но ведь обнаружиться нехватка регистров может в какой-нибудь внутренней конструкции, в том числе – во внутреннем цикле. Поэтому, если вы будете прямо там выгружать регистр в память, вы потеряете эффективность, поскольку в цикле эта выгрузка произойдет много раз. Лучше было бы выгрузить значение сразу во время возникновения, но кто знает – долежит оно или нет. Цейтин предложил мне использовать некую технику, которую он придумал для естественных языков еще в 50-е годы, но, разумеется, тогда на машинах типа УРАЛ-1 у него не началось даже сколько-нибудь серьезных экспериментов. Я разработал такую технику, и мы применили её в нашем трансляторе. Но тогда памяти ЭВМ были очень маленькие, а вся техника основывалась на том, что варианты перебирались не с помощью известного уже даже тогда бэктреккинга (экспоненциально сложного), а с помощью представления в виде очень сложных деревьев, где варианты использования переменных представлены в листьях этого дерева. Удалось разработать набор действий (сложение, сравнение деревьев, даже циклы по деревьям), были проведены эксперименты,

но, к сожалению, эта техника в реальный транслятор не вошла – все-таки нужно много памяти. Но оказалось, что эту технику можно очень хорошо применять в другой задаче - задаче поиска методом-в-ширину.

Меня на эту идею подтолкнул Джекоб Шварц, знаменитый ученый, по книгам которого я учился в студенческие годы (Данфордн, Шварц, «Линейные операторы» – это основы функционального анализа). Позже Шварц перешел в программисты и, как и положено математику, изобрел интересный язык сверхвысокого уровня SETL. Кстати, одна из первых реализаций SETL была сделана в Новосибирске. На одной из международных конференций в 1976 году Шварц стал меня спрашивать, как этот метод синтеза, о котором я рассказывал в докладе, соотносится с недетерминированными языками искусственного интеллекта. Я ничего даже не слышал об этом. Мой научный руководитель Г.С. Цейтин сидел в зале, и я переадресовал этот вопрос ему, но он тоже не знал. Понимаете, «железный занавес» был для нас не пустым звуком, мы очень редко могли встретиться с иностранными учеными на конференциях у нас в стране и уж совершенно редко можно было поехать на конференцию за рубеж. Об этом и думать было смешно. После вопроса Шварца я пережил несколько неприятных мгновений, ощущая себя человеком, повторно изобретшим велосипед. Я боялся, что на Западе обо всем этом уже знают, но Шварц оказался очень хорошим человеком и прислал мне обычной почтой бандероль с копиями статей о языках AI (Artificial Intelligence, искусственный интеллект). Там была записка: «Андрей, не меняй местами статьи, они сложены в определенном порядке, в этом порядке их и читай».

Первой была статья Боброва и, кажется, Рафаэля с обзором языков AI, а потом было несколько специализированных статей (Карла Хьюита, автора языка Planner, Джеральда Сусмана, автора языка Коннайнвер и еще какие-то статьи). Статьи из этой бандероли я хранил как зеницу ока, но давал почитать моим коллегам. Многие ученые с мат-меха прочитали эти статьи. Это было первое наше окно в мир искусственного интеллекта, тогда у нас, кажется, никто ничего об этом не знал. За счет того, что все языки AI того времени были основаны на методе бэктреккинга - поиска-в-глубину, а я со своим научным руководителем предлагал в качестве метода решения поиск-в-ширину со специализированными структурами хранения промежуточных результатов, то оказалось, что это совсем новый взгляд на решение задач в этой новой для нас области. К тому времени я занимался уже многими параллельными делами: теми же военными разработками, стал большим начальником, директором предприятия Терком, да и вообще 90-ые годы были очень тяжелыми, и серьезно заниматься наукой было непросто. Только через 30 лет мой аспирант Дмитрий Иванов реализовал предложенную технику на совершенно новых инструментальных средствах, основываясь на BDD-деревьях (Binary Decision Diagrams), и провел серию экспериментов по сравнению традиционного подхода методом «сначала поиска-в-глубину» с методом «сначала-в-ширину», реализованным на основе наших «деревяшек». Он получил довольно много интересных результатов, которые оказались выигрышными для нас. Мне было очень интересно наблюдать, как идея, придуманная Цейтиным в 50-ые годы, была впервые реализована мной в 70-ые годы, а уже в XXI веке Дмитрий Иванов довел ее до практической реализации и получил сравнительные результаты.

Алгол 68 в образовании и научных исследованиях

Алгол 68 постепенно стал языком научных публикаций в нашей области. Хочется упомянуть хотя бы два примера. Профессор кафедры исследования операций нашего мат-меха Иосиф Владимирович Романовский написал книгу по исследованию операций [14], где все алгоритмы были написаны на Алголе 68, справедливо полагая, что так они выглядят наиболее компактно, ярко и выразительно.

Еще один замечательный пример. Мой близкий приятель Николай Непейвода, профессор из Удмуртского университета (г. Ижевск) около 20 лет назад опубликовал статью [15], как по доказательству в интуиционистской логике (то есть без закона исключенного

третьего) можно сгенерировать алгоритм работы программы, т.е. сгенерировать действующую программу по доказательству ее корректности. В этой статье тоже органичным образом были вставлены алгоритмы на Алголе 68, так как структурами и юнионами этого языка было очень удобно описать логические преобразования.

Из более приземленных и практических вопросов образования хотелось бы вспомнить, что мы начали внедрять примерно в 1978-79 гг. Алгол 68 на первом курсе мат-меха в качестве базового языка обучения, и в течение многих лет именно так и было. Оказалось, что работа в дисплейных классах на центральной ЭВМ занимает слишком много ресурсов. Когда работает профессиональный программист, то его больше интересует время счета, а когда работает студент или начинающий пользователь, то он совершает столько ошибок, что до счета доходит весьма малый процент программ, а основное время уходит на синтаксический и видовой анализ и сигнализацию об ошибках. В результате, центральная ЭВМ, на которой работало несколько десятков студентов одновременно, полностью была занята синтаксическим анализом.

Начитавшись статей Вирта про P-коды (виртуальные машины), которые он использовал для создания переносимых трансляторов с Паскаля, мы решили сделать что-нибудь в этом роде. Мы придумали свою виртуальную машину, она называлась Айвик - я даже уже и не помню почему. Наталья Вояковская реализовала компактный анализатор с Алгола 68, и этот анализатор с помощью кросс-транслятора из Алгола 68 в коды Айвека мы поместили в дисплей 7063, в котором был однобайтовый процессор Intel8080. Там было всего 64 килобайта памяти, тем не менее, через компактную виртуальную машину нам удалось уместить анализатор с Алгола 68 в эту маленькую память, в результате студенты работали в дисплейных классах и, когда они запускали трансляцию, то незаметно для них запускался анализатор, который сидел прямо в этом дисплее. В случае ошибок именно этот анализатор выдавал ошибку, и на центральную ЭВМ программа даже не отправлялась. Если же ошибок не было, программа шла на центральную ЭВМ. В результате, нагрузка на последнюю упала в десятки раз. Жизнь студентов, работающих за дисплеями, стала более комфортная, так как время ожидания сократилось на порядок.

Похожий прием мы использовали позже, когда работали над созданием телефонных станций специального назначения. Там тоже была центральная ЭВМ и куча периферийного оборудования, в котором были очень примитивные вычислители (скажем, с 4-битовым кодом операции или что-то в этом роде), а требования по реактивности в системах реального времени уже тогда были весьма жесткими. Мы провели инструментирование исполняемых команд на интерпретаторах и выяснили «бутылочные горлышки» – времяпожирающие участки программ, которые требовалось делать максимально эффективно. Как всегда, оказалось, что такие места занимают не более 5% общего объема программ, а 95% занимает всякое техобсулуживание, работа с оператором, то есть такие фрагменты, которые не требуют реального времени. Тогда мы на эти примитивные вычислители поставили интерпретатор виртуальной машины, который можно было сделать даже в таких простых условиях. 95% исходных программ на Алголе 68 мы транслировали в коды виртуальной машины Айвек, а 5% писали в виде микропрограмм прямо в кодах этого периферийного устройства управления. Оказалось, что мы уместились и в памяти, и уложились во временные ограничения, но при этом затраты на программирование уменьшили во много раз.

На самом деле, первый опыт виртуальных машин у нас был связан с именем Юрия Матиясевича, известного математика, автора решения десятой проблемы Гильберта. Он уже в 70-ые годы был всемирно известным ученым и мог выезжать за границу. Именно он привозил первые портативные ЭВМ Синклер, еще какие-то, размером с небольшую мыльницу с процессором, который чаще всего был однобайтовым. Юра хотел использовать компьютеры для своих экспериментов, и ему очень нужна была хотя бы 16-разрядная арифметика. Именно тогда мы создали для него Айвек (16-разрядная виртуальная машина) и написали массу интерпретаторов Айвека для всех машин, которые Юра привозил из-за

границы. У нас на эту тему есть публикации, в частности в журнале «Автоматика и телемеханика» [16], который уже тогда переводился на английский, французский и немецкий языки. Если смотреть по датам, то это примерно на 15 лет раньше, чем вал публикаций по Java и Java-байткоду. Понятно, что тогда было очень трудно конкурировать и бороться за мировое господство, но, во всяком случае, то, что мы умели это делать, подтверждено публикациями в солидных журналах.

Список литературы

[1] Report on the Algorithmic Language Algol 60, P. Nauer, Editor, http://web.eecs.umich.edu/~bchandra/courses/papers/Naure_Algol60.pdf

[2] Revised Report on the Algorithmic Language Algol 60, Edited by Peter Naur, http://www.masswerk.at/algol60/report.htm

[3] Dec. 1968: Report on the Algorithmic Language ALGOL 68 - Offprint from Numerische Mathematik, 14, 79-218 (1969); Springer-Verlag.[12] - Edited by: A. van Wijngaarden, B.J. Mailloux, J.E.L. Peck and C.H.A. Koster

[4] Sep 1973: Revised Report on the Algorithmic Language Algol 68 - Springer-Verlag 1976[18] - Edited by: A. van Wijngaarden, B.J. Mailloux, J.E.L. Peck, C.H.A. Koster, M. Sintzoff, C.H. Lindsey, L.G.L.T. Meertens and R.G. Fisker

[5] Ван Вейнгаарден А. [и др.], Сообщение об алгоритмическом языке АЛГОЛ-68, «Кибернетика», 1969, №6; 1970, №1

[6] Yacc: Yet Another Compiler-Compiler, Stephen C. Johnson, http://dinosaur.compilertools.net/yacc/

[7] Wulf, W. A., Johnson, R., Weinstock, C., Hobbs, S., and Geschke, C., The Design of an Optimizing Compiler American Elsevier Publishing Company, Inc., New York, 1975.

[8] Терехов А.Н., “Процессы идентификации и структура компилятора с языка Алгол 68”, “Программирование”, N2, 1975, http://www.math.spbu.ru/user/ant/all_articles/003_Terekhov_Identification.pdf

[9] Паскаль. Руководство для пользователя / Пер с англ и предисл. Д. Б. Подшивалова. М. Финансы и статистика, 1989. 255 с.

[10] Терехов А.Н., “Библиотечные вступления и отдельная трансляция процедур в трансляторе с Алгола 68” (тезисы), Тезисы докладов и сообщений на Всесоюзной конф., ч.2, Вильнюс, 1980, http://www.math.spbu.ru/user/ant/all_articles/010_Terekhov_Bibliotechn_Algol68.pdf

[11] Терехов А.Н., Брыксин Т.А., Литвинов Ю.В. QReal: платформа визуального предметно-ориентированного моделирования. // Программная инженерия, 2013, № 6, С. 11-19

[12] «УВК «Самсон» - базовая ЭВМ РВСН», Труды SORUCOM-2011 (Вторая международная конференция «Развитие вычислительной техники и ее программного обеспечения в России и странах бывшего СССР»), 2011, http://www.math.spbu.ru/user/ant/all_articles/092_Terekhov_history_Samson.pdf

[13] “Алгол 68. Методы реализации“ под редакцией Г.С.Цейтина, Изд.ЛГУ, 1976, авторы - Балуев А.Н., Братчиков И.Л., Гиндыш И.Б., Крупко Н.А., Терехов, А.Н., Цейтин Г.С. и др. (всего 12 человек), кол-во страниц – 224

[14] И.В. Романовский, Алгоритмы решения экстремальных задач. М.: Наука, 1977. 352 с

[15] Непейвода Н. Н., Применение теории доказательств к задаче построения правильных программ, Кибернетика, 1979 г., №2, сс. 43-48

[16] Матиясевич Ю.В., Терехов А.Н., Федотов Б.А.»Унификация программного обеспечения микроЭВМ на базе виртуальной машины”, “Автоматика и телемеханика”, N5, Москва, 1990, кол-во страниц -7