А. П. Стасенко[1]

ОБЗОР ПОТОКОВЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ[2]

ВВЕДЕНИЕ

Язык программирования называется потоковым, если он подходит для описания программ для потоковых архитектур, которые на машинном уровне описывают вычисления в виде графа потока данных. Поэтому потоковый язык должен удовлетворять следующим критериям.

1. Возможность извлечения зависимостей по данным из текста программы, что достигается при соблюдении принципа локальности действия (locality of effect) или прозрачности имён ссылок (referential transparency), олицетворяющих значения, а не ячейки памяти.
2. Последовательность вычислений определяется только готовностью данных, что обеспечивается отсутствием побочных эффектов (side-effect) вычислений или их математической правильностью.

Потоковые языки программирования обычно являются представителями более широкого класса «чистых» (без побочных эффектов) функциональных языков, задающих вычисления путём описания применений функций к их аргументам. Функциональные языки, основанные на λ-исчислении Черча, обладают большей «выразительной мощностью», чем императивные языки программирования, основанные на машинных языках. Выразительная мощь функциональных языков заключается в легкости описания операций над сложными объектами, такими как сами функции, и неявно задаваемом машинно-независимом параллелизме, ограниченном лишь зависимостями по данным.

В связи с тем, что функциональные языки не фиксируют порядок вычислений, для них актуально понятие стратегии вычисления, определяющей порядок применения функций. Строгая стратегия (strict, eager, call-by-value), заключающаяся в применении функции уже к вычисленным аргументам, может приводить к зацикливанию вычислений, если зацикливается даже какая-то его часть, не влияющая на общий результат. Ленивая страте-

[1] astasenko@gmail.com

[2] Работа выполнена при финансовой поддержке Министерства образования РФ (научная программа «Университеты России», грант № УР.04.01.201).

гия (lazy, non-strict, call-by-need), откладывающая вычисления пока это возможно, свободна от этого недостатка, что позволяет включать в язык потенциально бесконечные структуры данных, но сильнее ограничивает параллелизм. Возможны и другие (lenient) смешанные стратегии вычисления.

Принцип прозрачности имён ссылок меняет семантику оператора присваивания на определение связи между именем и значением. Свойство однократного присваивания (single-assignment), не являющееся обязательным для потоковых языков, запрещает переопределение одного имени в пределах одной области. Также принцип локальности действия приводит к необходимости обрабатывать ошибочные ситуации с помощью специально выделенных для этой цели ошибочных значений.

Вначале свойства потоковых языков в значительной степени определялись особенностями существующих потоковых архитектур, первые из которых были статическими и не допускали повторное использование потокового кода, в том числе рекурсии. Данные ограничения были в дальнейшем преодолены в динамических потоковых архитектурах, которые разрешали одновременное продвижение множества наборов разметок по дугам потокового графа.

В настоящее время потоковые языки потеряли свою изначальную привязанность к потоковым архитектурам и в основном рассматриваются из-за своей выразительной мощности. Далее более подробно будут рассмотрены типичные представители потоковых языков, такие как языки Lucid, Id, Val, Post, Sisal и Пифагор.

1. LUCID

Нестрогий язык Lucid, разработанный в 1976 г. [1] для динамической потоковой архитектуры, был одним из первых потоковых языков. В своём изначальном виде язык Lucid представлял собой расширение двумя новыми операторами *next* и *fby* одного из первых функциональных языков ISWIN (If You See What I Mean). Изначально язык Lucid предназначался для верификации программ.

В языке Lucid все значения, даже константные, являются потоками, причем числовые константы обозначают потоки, составленные из их повторений. Обычные арифметические операции выполняются для потоков покомпонентно. Пусть есть потоки $X = (x_0, x_1, \ldots, x_n, \ldots)$ и $Y = (y_0, y_1, \ldots, y_n, \ldots)$, тогда *next* $X = (x_1, x_2, \ldots, x_{n+1}, \ldots)$ и X *fby* $Y = (x_0, y_0, y_1, \ldots, y_{n-1}, \ldots)$. Потоки можно определять рекурсивно. Потоки языка Lucid являются конечными,

если в качестве их хвоста использовать специальное значение *eod* (End Of Data).

В процессе развития в языке Lucid произошла смена базовых операций *fby* и *next* на операции запроса (querying) текущего индекса и операции, индексирующей (navigating) поток X потоком Y. Новые операции являются фундаментальными для содержательного (intensional) программирования, основанного на интенсиональной логике.

Современная версия языка Lucid называется Indexical Lucid и поддерживает многомерные потоки и операции над их отдельными именованными размерностями. Также язык Lucid был расширен за счет операций над самими размерностями и функциями.

В дальнейшем язык Lucid был расширен в нескольких направлениях. Разновидности языка использовались для спецификации трёхмерных электронных таблиц, систем реального времени с помощью разновидности Lustre и реагирующих (reactive) систем с помощью Lucid Synchrone, взаимодействий агентов с помощью AIPL (Agent Intensional Programming Language). Для реализации этих и других диалектов языка Lucid была разработана единая среда GIPSY (General Intensional Programming System).

Одно из интересных расширений языка Lucid, разработанное в калифорнийской компании SRI International, получило название Granular Lucid (GLU) [2] и предназначалось для описания крупнозернистого параллелизма, где последовательные части задавались на языке Си. Программа на языке GLU состоит из текста программы языка Lucid с объявлениями типов и заголовками функций и языка Си, тела которых находятся в отдельном файле. Допустимы любые объявления типов языка Си, так как их использование ограничено аргументами и результатами импортируемых функций языка Си. Тем самым от языка Lucid язык GLU, по сути, отличается более гранулированными элементарными операциями.

2. ID (IRVINE DATAFLOW)

Язык параллельного программирования общего назначения Id, первое упоминание о котором датируется 1978 г. [3], был в основном спроектирован в Массачусетском технологическом институте (MIT), где была разработана версия Id Nouveau [4]. Последняя рассматриваемая ниже ревизия языка Id 90.1 [5] датируется 1991 г.

Язык Id всё ещё остаётся исследовательским языком, развивающимся в направлении лучшего выражения недетерминированных вычислений, вво-

да-вывода и управления ресурсами. Для языка Id существует функционирующий компилятор для потоковой архитектуры с реализацией передачи маркера Массачусетского технологического института и вычислительной системы Monsoon фирмы Motorola.

Язык Id предназначается для программирования потоковых и других параллельных архитектур и содержит три синтаксически разделяемых слоя.

1. Основная часть языка, являющаяся чисто функциональным языком с нестрогой семантикой вычисления.
2. Расширение с помощью I-структур, сохраняющее детерминизм основной части, но теряющее гарантируемую прозрачность ссылок (referential transparency).
3. Расширение с помощью M-структур и ввода-вывода, которое может приводить к недетерминированным вычислениям.

Функциональная часть языка Id поддерживает функции высшего порядка, средства для явного задания отложенных вычислений, сопоставление с образцом и явно выделенные циклические выражения. Также в языке используются статическая полиморфная система типов в стиле Милнера с перегрузкой, определяемые пользователем типы, списки и массивы с развитыми средствами их генерации (comprehension).

Язык Id поддерживает переменные типов для обозначения зависимостей между компонентами полиморфных агрегатных типов. Допускается определение алгебраических типов (или несвязных объединений), которые позволяют задавать многие типы, такие как булевский тип, встроенные в других языка программирования. Языком Id также поддерживаются абстрактные типы данных.

Компоненты структуры данных языка Id могут иметь функциональную семантику, семантику I-структуры или семантику M-структуры. В случае функциональной семантики компоненты создаются и инициализируются одновременно, тогда как компоненты I-структуры и M-структуры создаются пустыми и инициализируются позже.

Значение I-структуры может инициализироваться только один раз и, в случае повторной инициализации, возникает исключение, делающее всю программу несостоятельной. Читать значение I-структуры можно произвольное число раз, и попытка прочитать значение пустой I-структуры откладывается до её инициализации.

Значение M-структуры может меняться произвольное число раз, но изменение возможно только, если M-структура является пустой. Попытка изменения непустой M-структуры приводит к исключению, делающему всю программу несостоятельной. Чтение непустого значения M-структуры

делает её снова пустой, откладывая все другие, возможно, параллельные чтения этого значения, произвольное из которых будет вновь возобновлено при последующем присваивании значения этой М-структуры.

Типичное применение М-структур заключается в использовании несколькими параллельными вычислениями одного неразделяемого ресурса. Каждое вычисление читает значение ресурса, задаваемого М-структурой, работает с ним и записывает его обратно. Семантика доступа М-структуры гарантирует эксклюзивный доступ к ресурсу.

В целях машинно-зависимых оптимизаций в языке Id можно ограничивать степень параллелизма циклов и других конструкций языка, исполняемых параллельно, явным образом, путем указания максимального количества одновременно исполняемых итераций. Эти ограничения могут стать причиной зависания вычислений с обратными (циклическими) зависимостями, которые не запрещаются языком.

3. VAL

Язык Val (Value Algorithmic Language) был разработан в 1979 г. [6] в Массачусетском технологическом институте (MIT) и предназначался для статической потоковой архитектуры, что привело к отсутствию типичных для динамической потоковой архитектуры возможностей, таких как рекурсия. В то же время язык не содержал машинно-зависимых элементов, включая встроенный ввод и выход. Являясь небольшим исследовательским языком, язык Val демонстрирует возможность использования языка программирования, принадлежащего высокому уровню, в качестве потокового.

В отличие от большинства других функциональных языков, язык статически типизирован (применяется структурная эквивалентность) разветвленной системой типов и синтаксически напоминает язык программирования Pascal. В частности, в языке содержится «оператор присваивания», который семантически является однократным связыванием (single-assignment) значения с именем. Также язык Val в явном виде содержит несколько форм циклических выражений, одна из которых может использоваться для задания явного параллелизма. Функции языка Val, как и другие выражения, могут возвращать несколько значений.

Каждый тип языка Val дополнительно содержит одно выделенное ошибочное значение, что гарантирует независимость результата от порядка вычисления. Стратегия вычислений языка Val в первую очередь определяется соображениями эффективности и простоты, сочетая ленивые и строгие

вычисления. Ввиду отсутствия в языке побочных эффектов и наличия выделенных ошибочных значений, язык Val хорошо подходит для строгих вычислений. Однако в некоторых конструкциях, таких как выражения выбора и циклы с условием, где сначала вычисляется управляющее выражение, используется ленивая стратегия для уменьшения ненужных вычислений. Параметры функции в языке Val всегда вычисляются до её вызова.

Смешанная стратегия вычислений нарушает эквивалентность отдельных преобразований. Например, в общем случае нельзя заменить выражение выбора эквивалентным вызовом функции, так как выражению выбора может потребоваться только часть используемых в нём значений, а вызов функции должен вычислить их все. Также возможность языка Val обнаруживать и исправлять ошибочные значения, в отличие, например, от функционального языка FP, сохраняющего $\perp$ (ошибочное состояние), приводит к некорректности некоторых преобразований программ, свойственных функциональным языкам, таких как перенос функционального вызова на ветви условного выражения.

4. POST

Экспериментальный язык Post появился в 1981 г. в качестве одного из результатов кандидатской диссертации [7]. Язык Post более точно, чем языки Val и Sisal, отражает особенности потоковых архитектур, но его программы сложнее для понимания. В языке Post предложено несколько новых идей развития потоковых языков:

- возможность влиять на используемую стратегию вычисления;
- смешанно-синхронные структуры данных, являющиеся частично синхронными, а частично — асинхронными;
- взаимодействие между вычислениями для прекращения ненужных вычислений, что нехарактерно для чистых функциональных языков.

Язык Post не был полностью реализован за исключением прообраза компилятора, преобразующего программный текст в потоковый граф и, затем в инструкции динамической потоковой машины со специфическими особенностями, которая тоже так и не была создана.

5. SISAL

Язык Sisal (Streams and Iterations in a Single Assignment Language) является развитием языка Val, по большей части разработанным в Ливерморской национальной лаборатории имени Лоренца (LLNL). Первая основная версия языка Sisal 1.2 была зафиксирована в 1985 г. [8]. Для этой версии разработаны оптимизирующие компиляторы под различные машинные архитектуры. Последующие «официальные» версии языка Sisal 2.0 [9] и Sisal 90 [10, 11], разработанные в 1991 и 1995 гг. соответственно, так и не были реализованы. Для языка Sisal 90 не было создано даже строгой спецификации. Языки Sisal 1.2, Sisal 2.0 и Sisal 90 синтаксически не совместимы между собой. В то же время язык Sisal 90 находится значительно ближе, чем язык Sisal 2.0, к языку Sisal 1.2.

Язык Sisal является универсальным языком общего назначения, но с ярко выраженной научной направленностью, позиционирующийся как замена языка Fortran. Синтаксически язык Sisal, как и язык Val, напоминает язык Pascal. Язык Sisal, в отличие от языка Val, разрешает использование рекурсии, так как был разработан для более прогрессивных динамических потоковых архитектур.

Уже в версии Sisal 1.2 в язык были введены тип потока для облегчения организации машинно-независимого ввода-вывода и поддержка модульности на уровне импорта-экспорта отдельных функций. Версия Sisal 2.0 расширяла язык функциями высшего порядка, множествами типов, более полной поддержкой модульности, алгебраическими матрицами и покомпонентными операциями над массивами и потоками.

Версия Sisal 90 содержала все нововведения версии Sisal 2.0 за исключением типа алгебраических матриц. Покомпонентные операции над массивами и потоками в языке Sisal 90 были позаимствованы из языка Fortran 90. В языке Sisal 90 появилась возможность задавать пользовательские редукции и константы времени исполнения, не нарушающие принцип математической чистоты функций в пределах одного запуска программы.

Следующий концептуальный виток развития языка Sisal 90 был совершен в версии Sisal 3.0 [12], разработанной в Институте систем информатики (ИСИ) имени А. П. Ершова СО РАН. Нововведения языка Sisal 3.0 заключаются в возможности задавать отдельные части программы на императивном языке Си, расширенной поддержке модульности программ, возможности их препроцессинга и аннотирования для упрощения оптимизирующих преобразований.

Для последней версии языка Sisal 3.1, находящейся в разработке в ИСИ, впервые со времён языка Sisal 90 был строго формализован синтаксис его функциональной части. Функциональная часть языка была усилена за счёт возможности переопределения операций и перегрузки вызовов функций и редукций. Синтаксис и семантика некоторых конструкций языка Sisal 90 были изменены для повышения читаемости и упрощения разбора в существующем трансляторе переднего плана языка Sisal 3.1.

6. ПИФАГОР

Язык программирования Пифагор [13], разработанный в Красноярском Государственном Техническом Университете (КГТУ) в 1995 г., предназначен для разработки переносимых параллельных программ, управление вычислениями в которых осуществляется по готовности данных. Язык Пифагор ориентирован на непосредственное описание информационного графа, что привело к синтаксису языка, несколько отличающемуся от общепринятого синтаксиса. В частности, не поддерживаются инфиксные бинарные операции, что снижает читаемость программы. Ввиду экспериментальной природы языка Пифагор, в нем отсутствуют конструкции, повышающие технологичность разработки программ, такие как модульное построение программ, раздельная трансляция и согласование со стандартными библиотеками.

Существующие на данный момент версии языка Пифагор и исполнительной системы поддерживают только динамическую типизацию. Язык поддерживает простые типы и три вида списков, которые могут быть вложены друг в друга. Списки бывают последовательными, параллельными и задержанными. Каждый тип языка дополнительно содержит несколько различных значений, обозначающих разные виды ошибок вычисления. Альтернативные вычисления реализуются через механизм задержанных списков. Особенности модели вычислений и синтаксиса языка Пифагор накладывают отпечаток на методы и стиль программирования. Отсутствие операторов цикла позволяет писать потоковые программы без синхронизации перед входом в циклический фрагмент, но в то же время приводит к необходимости использования рекурсии.

Динамическая типизация позволяет получать интересные эффекты, расширяющие возможности языка. В частности, сочетание механизма перегрузки функций и динамических типов, определяемых пользователем, обеспечивает написание эволюционно расширяемых параллельных программ. В

языке Пифагор реализован механизм перегрузки функций, позволяющий гибко и безболезненно расширять уже разработанную программу. Так как в языке отсутствует строгая типизация, все функции с одинаковыми именами становятся неразличимы (имеют одинаковую сигнатуру). Поэтому вместо выбора одной из перегруженных функций осуществляется их одновременное выполнение. Результат возвращается в виде параллельного списка.

ЗАКЛЮЧЕНИЕ

Рассмотрены общие определяющие качества потоковых языков программирования и характерные особенности некоторых распространенных и экспериментальных потоковых языков.

СПИСОК ЛИТЕРАТУРЫ

1. **Ashcroft E. A. and Wadge W. W.** Lucid: A formal system for writing and proving programs // SIAM J. on Computing. — 1976. — Vol. 5, No. 3. — P. 336–354.
2. **Jagannathan R. and Faustini A. A.** The GLU programming language. — Menlo Park, CA, 1990. — (Tech. Rep. / SRI International, Computer Science Laboratory; SRI-CSL-90-11).
3. **Arvind, Gostelow K. P. and Plouffe W.** An asynchronous programming language and computing machine. — Irvine, CA, 1978. — (Tech. Rep. / Univ. of California, Department of Information and Computer Science; 114a).
4. **Arvind, Nikhil R. S. and Pingali K. K.** Id Nouveau reference manual. — Cambridge, MA, 1986. — 64 p. — (Tech. Rep. / Massachusetts Institute of Technology, Laboratory for Computer Science, Computation Structures Group; Memo-265).
5. **Nikhil R. S.** Id language reference manual (version 90.1). — Cambridge, MA, 1991. — 54 p. — (Tech. Rep. / Massachusetts Institute of Technology, Laboratory for Computer Science, Computation Structures Group; Memo-284-2).
6. **McGraw J. R.** Val language, description and analysis. — Livermore, CA, 1980. — 51 p. — (Tech. Rep. / Lawrence Livermore National Laboratory; UCRL-83251, Rev. 1).
7. **Ravishankar C. V.** Post: a language for dataflow programming. — Ph.D. thesis. — Madison: University of Wisconsin, Computer Sciences Department, 1987. — 213 p.
8. **McGraw J. R.** Sisal: Streams and iterations in a single assignment language, Language Reference Manual, Version 1.2. / McGraw J. R., Skedzielewski S. K., Allan S. J., Oldehoeft R. R., Glauert J., Kirkham C., Noyce B. and Thomas R. — Livermore, CA, 1985. — (Tech. Rep. / Lawrence Livermore National Laboratory; M-146, Rev. 1).

9. **Böhm A. P. W.** The Sisal 2.0 reference manual / Böhm A. P. W., Cann D. C., Feo J. T. and Oldehoeft R. R. — Livermore, CA, 1991. — 128 p. — (Tech. Rep. / Lawrence Livermore National Laboratory; UCRL-MA-109098).
10. **Feo J. T.** Sisal 90 user's guide / Feo J. T., Miller P. J., Skedzielewski S. K. and Denton S. M. — Livermore, CA: Lawrence Livermore National Laboratory, Draft 0.96, 1995. — 80 p.
11. **Бирюкова Ю. В**. Sisal 90: Руководство для пользователя. — Новосибирск, 2000. — 84 с. — (Препр. / РАН. Сиб. Отд-е. ИСИ; № 72).
12. **Касьянов В. Н., Бирюкова Ю. В., Евстигнеев В. А.** Функциональный язык Sisal // Поддержка супервычислений и интернет-ориентированные технологии. — Новосибирск: ИСИ СО РАН, 2001. — С. 54–67.
13. **Legalov A. I., Kazakov F. A., Kuzmin D. A.** Description of parallel-functional programming language // Advances in Modeling & Analysis, Series A. — Brno, Czech Republic: AMSE Press, 1995. — Vol. 28, No. 3. — P. 1–17.