УДК 735.29.(32)

СОБЫТИЙНЫЙ ИНТЕРПРЕТАТОР ДЛЯ ФУНКЦИОНАЛЬНОГО ЯЗЫКА ПОТОКОВО-ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ

Матковский И.В.
Научный руководитель – д-р. техн. наук, профессор Легалов А.И.

Сибирский федеральный университет

Язык программирования Пифагор был создан для анализа способов построения, промежуточного представления, исполнения и отладки функционально-потоковых параллельных программ. Последняя версия интерпретатора языка позволяет выполнять программы в режиме обработки очереди событий.

Написанная на языке Пифагор программа обрабатывается транслятором; при отсутствии ошибок создается промежуточное представление программы – информационный граф. По информационному графу создается граф управляющий; он показывает, как именно описанные в информационном графе данные взаимодействуют между собой. Промежуточные представления графов хранятся в двоичных файлах в специально выделенном каталоге – модульной библиотеке. Модульная библиотека представляет собой иерархическую систему каталогов, каждый из которых соответствует определенному пространству имен. Принадлежность функции к пространству имен определяется её полным именем – перед именем самой функции через точку последовательно перечисляются те пространства имен, в которые она входит. Так, функция *sort.hoar.main* будет считаться элементом пространства имен *hoar*, являющегося, в свою очередь, элементом пространства имен *sort*; её информационный граф будет храниться в файле *main.ig* в директории *\sort\hoar* в каталоге модульной библиотеки. При написании программ допускается вызов функции, еще не написанных или не оттранслированных – у транслятора несуществующая функция не вызовет никаких нареканий.

Объектом интерпретации является функция; кроме информационного и управляющего графов для работы с ней может понадобиться аргумент – преобразованный с помощью транслятора в понятный для интерпретатора формат. Вычисляемое выражение – частный случай функции; от обычной функции оно отличается тем, что не принимает аргументов. Ввиду того, что операции ввода информации в ходе выполнения программы в языке пока не определены, такие выражения всегда будут выдавать один и тот же результат – в полном соответствии с определением чисто функциональных (purely functional) языков. Следует отметить, что чистая функциональность для языка Пифагор основной целью не является, и в дальнейшем язык может пополниться элементами, чистоту функциональности нарушающими. Значения вычисляемых выражений можно сохранять в модульной библиотеке – аннулируя при изменении исходного кода функции. Если вычисляемое выражение для своего исполнения не требует никаких внешних функций и работает лишь с примитивами языка Пифагор – оно может быть рассчитано еще на этапе трансляции и сведено к возвращению рассчитанного значения.

После начала интерпретации анализируется управляющий граф функции и производится выделение уже готовых к выполнению задач. Эти задачи помещаются в специальную очередь, откуда в дальнейшем извлекаются самим интерпретатором. По мере интерпретации имеющихся событий порождаются события новые; процесс продолжается до тех пор, пока в очереди присутствуют задачи.

Как и во многих других функциональных языках – Haskell, LISP, Python – в Пифагоре активно используются списки данных. Наиболее простой концепцией являются обычные списки данных (data list) – они представляют собой обычные массивы с рядом обычных для функциональных языков ограничений по изменению.

Идея параллельных списков (parallel list) несколько сложнее; такой список представляет собой не столько обычный массив данных, сколько ряд совместно передаваемых элементов. Параллельный список может рассматриваться и на правах списка обыкновенного; при интерпретации, однако, он не как массив, но в как ряд отдельных составляющих – для каждого из которых операция интерпретации производится независимо и параллельно с остальными.

Задержанные списки (delayed list) представляют собой выделенную часть программного кода – и информационного графа – чье выполнение производится лишь по отдельному распоряжению. До раскрытия с точки зрения программы задержанный список представляет собой элемент, похожий на обычную переменную; впрочем, термин “переменная” применительно к Пифагору некорректен, поскольку язык построен на принципе единственного присваивания (single assignment). Задержанные списки предоставляют программисту своеобразную альтернативу распространенным в других функциональных языках ленивым вычислениям (lazy computations); на их же основе в дальнейшем планируется введение обработки функций высшего порядка в качестве полноправных объектов первого уровня (first class object).

Из всех четырех представленных в семантике языка видов списков асинхронный (asynchronous list) является наиболее новым. При обработке списка любого другого типа необходимо сначала дождаться его окончательного формирования – и лишь потом начать необходимые расчеты. Благодаря механизму асинхронных списков нет никакой необходимости дожидаться полного формирования объекта – можно проводить обработку массива по мере поступления элементов.

Интерпретатор осуществляет поиск функций в модульной библиотеке по их полному имени; именно на этапе интерпретации будет выдана ошибка в том случае, если одной из используемых функций не будет обнаружено в библиотеке. Проверка наличия вызываемых функций производится перед началом самой интерпретации – сначала проверяется существование всех функций, вызываемых из обрабатываемой, затем – функций, вызываемых из каждой проверенной на предыдущем шаге функции и так далее.

Для обработки внешних функций создается отдельный экземпляр интерпретатора; у него, однако, сохраняется доступ к общему и единственному каталогу уже сохраненных в памяти промежуточных представлений элементов. Создание отдельных интерпретаторов позволяет обеспечить большую гибкость и максимальный параллелизм обработки функций.