Написано 25.3.82.
Напечатано на 3 стр.
Отправлено в ИПИ 2.4.82.

234-38

Член-корреспондент АН СССР А.П.Ершов, Б.Н.Островский

Систематическое построение программы решения частной задачи из некоторого класса методом смешанных вычислений на примере синтаксических анализаторов

Поиск общего метода решения любой задачи, принадлежащей некоторому естественному классу, является одной из главных парадигм современной науки. В конструктивной математике эта черта находит свое выражение в построении так называемых универсальных алгоритмов. Построение такого алгоритма начинается с выделения класса задач и отображения этого класса во множество P конструктивных объектов, называемых формулировкой задачи. Универсальный алгоритм - это программа, реализующая функцию $U(f)$, такую что $U(f)$ для любого $f \in P$ дает решение задачи с формулировкой f. Обычно решение задачи f само является функцией $f(x)$ параметра x, характеризующего индивидуальный вариант задачи f. Этот параметр, естественно, становится дополнительным аргументом универсального алгоритма: $\forall f \forall x\ U(f,x)=f(x)$.

Решение задач с помощью универсального алгоритма связано с определенными издержками, поскольку алгоритм, рассчитанный на общий случай, в применении к частной задаче неизбежно оказывается избыточным по отношению к индивидуальному способу, придуманному только для данной задачи. Если частная задача решается многократно, то издержки общего подхода могут оказаться чрезмерными. Это хорошо известное противоречие между универсализацией и специализацией. Возникает принципиальная проблема систематического поиска эффективных решений частных задач, не утрачивая

знания, воплощенного в универсальных алгоритмах. В настоящем докладе сообщаются некоторые результаты исследования этой проблемы, проведенного авторами за последние четыре года.

В общей теории вычислимости ([1]) установлено существование универсальной процедуры, которая для любой программы $p(\underline{x},\underline{y})$, реализующей функцию $\varphi(\underline{x},\underline{y})$, эффективно строит программу $p_d(\underline{y})$, реализующую функцию $\varphi(d,\underline{y})$, где d – константа предметной области. В дальнейшем эта процедура несколько раз переоткрывалась специалистами по программированию (см. обзор ([2])), в том числе под названием частичных, или смешанных вычислений. Основой смешанных вычислений является обобщение семантики языков программирования, допускающее выполнение программы при неполностью заданных входных значениях. Программа $p_d(\underline{y})$ называется остаточной программой, или проекцией программы p на доступные данные d. Каждое из этих названий отражает разные содержательные свойства смешанных вычислений: первое говорит о том, что $p_d(\underline{y})$ – это тот программный текст, который остается после редукций программы, становящихся возможными для заданной информации d. Второе название трактует смешанные вычисления как освоение, ассимиляцию программой заданной информации d.

Было высказано предположение ([3]), что систематическое получение эффективных частных алгоритмов может быть осуществлено с помощью смешанных вычислений. Целью авторов было исследование продуктивности этой гипотезы в условиях вычислительного эксперимента, в котором особое внимание уделялось репрезентативности предметной области, систематичности процедуры и оценке качества частной программы.

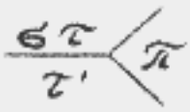
В качестве класса задач были взяты синтаксические анализаторы контекстно-свободных грамматик. Этот класс имеет хорошо развитую теорию универсальных анализаторов и носит реальный характер. В ка-

честве языка реализации использовалось содержательное подмножество языка Паскаль, имеющее все необходимое для эффективного программирования синтаксических анализаторов, но оставляющее за бортом периферийные возможности или средства, затрудняющие смешанные вычисления (отсутствуют типы: вещественные, указатели, массивы размерности выше I, вариантные записи, множества, файлы, кроме входных и выходных; отсутствуют операторы: перехода, выбора, цикла с параметром и условием окончания, присоединения; добавлены тип объединение и понятие блока).

Содержательная проблема извлечения эффективного алгоритма решения частной задачи $f(\underline{x})$ из универсального алгоритма $U(\underline{f},\underline{x})$ формализуется как получение оптимальной по объему и быстродействию остаточной программы $U_p(\underline{x})$ при смешанном вычислении алгоритма U при заданной формулировке f частной задачи. Смешанные вычисления трактуются ([4]) как направленный процесс применения пошаговых базовых трансформаций t программного текста p и данных d: $t(p,d)=(p',d')$. Корректность базовых трансформаций выражается в наличии некоторого инварианта $I(\underline{p},\underline{d})$, сохраняемого при выполнении разового преобразования, при этом $I(p,d)=I(p',d') \Rightarrow p(d)=p'(d')$.

Анализ смешанных вычислений показывает, что главным источником эффективности остаточной программы являются те редукции, которые становятся возможными благодаря доступной входной информации. Эти редукции носят универсальный характер, отражающий природу вычислений в языке реализации. Другим универсальным источником эффективизации является комбинаторная оптимизация остаточной программы, которая тоже формализуется в виде базовых трансформаций.

Содержательно, базовые трансформации описываются согласно ([5]) в виде трехчленного правила



где σ - формальный символ преобразования, τ - терм, подлежащий изменению, τ' - результирующий терм и π - условие применимости: $\sigma(\tau) =$ если π то τ' иначе τ. Символы преобразований могут входить в термы и тем самым инициировать дальнейшее выполнение преобразований. Смешанные вычисления завершаются, когда пара (p,d) не содержит символов трансформаций, что, в свою очередь, является следствием неприменимости преобразований. Формально, базовые трансформации выглядят как система рекурсивных определений над информационной структурой, описывающей преобразуемую программу и ее данные (p,d). Символы трансформаций становятся при этом именами рекурсивных процедур, условия применимости играют роль предикатов, а остаточная программа оказывается неподвижной точкой рекурсивной системы. Структура программы описывается в терминах абстрактного синтаксиса языка реализации, структура данных представляет собой иерархию векторов состояний, отражающую рекурсивный характер исполнения.

Для выбранного языка реализации была взята следующая (излагаемая упрощенно) номенклатура редукций. Группа описаний: редукция описания переменной, вносящая переменную в вектор состояний; редукция описания константы, связывающая ее имя и значение; редукция описания типа, раскрывающая структуру объекта; редукция неиспользуемых описаний. Группа выражений: редукция переменной, заменяющая ее имя на константу-значение, взятое из вектора состояний; редукция константного терма. Группа операторов: редукция оператора присваивания с возможным изменением вектора состояний; редукция вызова процедуры путем открытой подстановки; редукция чтения из входного файла с изменением вектора состояний; редукция записи в выходной файл; редукция условного оператора путем устранения невыполняемой ветви; редукция цикла путем однократного копирования

тела цикла; редукция блока, состоящая в глобализации его локальных переменных. В зависимости от степени доступности данных или способа вхождения языковой конструкции в программу имеет место один из четырех вариантов редукций: "чистое" выполнение с изменением вектора состояний или сверткой константного терма; "символическое" выполнение типа подстановки; задержка, когда процесс редукции транслируется с языковой конструкции на ее конституенты; "чистая" редукция, т.е. устранение неиспользуемой конструкции из программы.

При комбинаторной оптимизации остаточной программы использовались следующие схемные трансформации: вынесение одинаковых операторов из ветвей условного оператора (и обратно), запроцедуривание, отождествление процедур без параметров с одинаковыми телами, упрощение тел процедур с правосторонней рекурсией, зацикливание, синхронная подстановка тел процедур на места нескольких вызовов.

Существенным моментом является включение в смешанные вычисления некоторых действий, отражающих наше специальное знание класса задач и, прежде всего, универсального алгоритма. Например, универсальный анализатор рассчитан на произвольное число символов грамматики, в то время как для каждого частного анализатора это число является константой. В результате некоторые циклы универсального анализатора могут быть полностью раскрыты. Для этого в ходе смешанных вычислений их надо будет специальным образом запроцедурить, чтобы потом раскрыть с помощью открытых подстановок, которых, как мы знаем заранее, будет конечное число. Для этой цели универсальный анализатор подвергается предварительной модификации ("расширению"), состоящей во включении в его программу заготовок некоторых рекурсивных процедур, а система базовых трансформаций дополняется правилами, собирающими и внедряющую в остаточную программу эти процедуры в нужные места.

Грубо, построение частного анализатора складывается из трех рекурсивно сменяющих друг друга этапов: получение "сырой" остаточной программы, уже ассимилировавшей в своем тексте таблицу грамматики; затем схемные трансформации; наконец, дополнительные редукции, становящиеся возможными в результате перестройки текста программы.

В качестве универсального анализатора был взят таблично-управляемый МП-автомат для SLR(1)-грамматик ([6]). Вычислительный эксперимент предусматривал в качестве частной задачи построение анализатора для языка Паскаль. Как известно, LL(1)- и R- (регулярные) грамматики допускают более простые схемы анализа. Чтобы поставить универсальный анализатор в максимально выгодные условия, в Паскаль-грамматике были выделены LL(1)- и R-фрагменты и, с использованием специально разработанных приемов, построена комбинированная таблица грамматики. Кроме этого, таблица была подвергнута серии оптимизаций ([7-9]), направленных на ликвидацию холостых действий анализатора и сокращение таблиц.

Для верхней оценки качества частного анализатора был построен опытным программистом, но не знакомым с техникой смешанных вычислений, "индивидуальный" Паскаль-анализатор. Систематическое построение Паскаль-анализатора было проведено ручными вычислениями строго по рекурсивным описаниям базовых трансформаций, примененным к универсальному анализатору и таблице Паскаль-грамматики. Длины программ универсального (включая таблицу грамматики), систематически построенного и индивидуального анализаторов оказались соответственно 6314 , 6080 и 8050 команд объектного кода (БЭСМ-6). Наконец, на вход всех трех анализаторов были поданы разнообразные Паскаль-программы для выполнения фактического анализа. Анализаторы

234-44

выдают идентичные результаты, а именно, номера правил Паскаль-грамматики, образующих дерево разбора в некотором упорядочении.

Таблица 1.

Номер теста	Длина в лексемах	Чистое время анализа (в мс работы процессора)		
		Универсальный	Систематически построенный	Индивидуальный
I	287	82	I8	I8
2	I784	530	I08	I08
3	II59	396	78	68
4	779	246	46	42
5	I585	478	94	92
6	2668	794	I76	I64
7	5I7	I36	28	24
8	8I3	240	48	44

Результаты эксперимента, приведенные в Табл. I, позволяют сделать вывод о практичности систематического получения частных алгоритмов из универсальных с помощью смешанных вычислений.

Вычислительный центр СО АН СССР, Новосибирск
Алтайский политехнический институт, Барнаул

ЛИТЕРАТУРА

[1] Клини С.К., Введение в метаматематику, М., 1957, с. 305. [2] Ершов А.П., Программирование, № 5, с.21(1977). [3] Ершов А.П., ДАН, т.233, № 2, с.272(1977). [4] Ershov A.P., Lect. Notes in Comp. Sci., Heidelberg, v.118, p.16 (1981). [5] Pepper P., Lect. Notes in Comp.Sci Heidelberg, v.69, p.322 (1979). [6] Ахо А., Ульман, Дж., Теория синтаксического анализа перевода и компиляции, т.I, т.2, М., 1978. [7] Cohen J. et al., IEEE Trans. Software Eng., v.SE-5, N 5, p.472(1979). [8] Зонис В.С., Шумей А.С. Программирование, № 2, 3(1976). [9] Anderson T. et al., Acta Informatica, v.2, N 1, p.12 (1973).