Разделяемые структуры данных в системе автоматического доказательства теорем КВАНТ/3*

Е. А. Черкашин
Институт динамики систем и теории управления СО РАН, Иркутск, Россия
e-mail: eugeneai@icc.ru

A problem of development of new data structures for a new version of the automatic theorem prover (ATP) in positively constructed formulae calculus for QUANT/3 is considered. The aim of the development is to increase the prover's productivity by sharing the computer memory by most of the dynamic data structures, supporting the data structures enhancement with specific structural elements, and, in turn, allowing one to define strategies of the inference search.

Введение

Задачей исследования является разработка программной системы автоматического доказательства теорем (АДТ) для исчислений позитивно-образованных формул (ПО-формул) [1 – 3]. Исчисление ПО-формул и разработанные методы АДТ, основанные на этом исчислении, обладают существенно более широкой выразительностью в сравнении, например, с методом резолюций [4, 5].

В [3] определяется язык L ПО-формул. Методы АДТ в этом языке, построенные на основе ПО-исчислений, позволяют представлять в процессе поиска логических выводов (ЛВ) специализированные эвристики поиска ЛВ, поскольку ПО-формулы достаточно однородны и в то же время хорошо структурированы. Язык L — полный язык первого порядка, формулы которого представляются в виде деревьев: каждый узел — это позитивный квантор $\forall\bar{x}\colon A(\bar{x}) \stackrel{\text{df}}{=} \forall\bar{x}\,(A(\bar{x}) \to \ldots)$ или $\exists\bar{x}\colon A(\bar{x}) \stackrel{\text{df}}{=} \exists\bar{x}\,(A(\bar{x})\ \&\ \ldots)$ с условием на кванторную переменную в виде конъюнкции атомов или тождественно истинного предиката True; вдоль ветвей дерева структуры формулы типовые кванторы всеобщности и существования чередуются. В языке L построено дескриптивное исчисление J с единственным правилом вывода. Исчисление J корректно и полно относительно выводимости в исчислении предикатов первого порядка. На основе исчисления J разработан ряд других исчислений, в том числе конструктивных.

Существующие реализации системы АДТ в исчислениях ПО-формул, например система КВАНТ/1 [6], построены на алгоритмах, использующих копирование структур данных на каждом шаге вывода. Несмотря на то что процедуры копирования деревьев при выполнении шага ЛВ имеют линейную сложность, выполнение этой процедуры требует значительного процессорного времени и ресурсов памяти. Реализованная

*Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (гранты № 05-07-97201-р_байкал_в, № 05-07-97204-р_байкал_в).

в [6] программная подсистема АДТ обладает возможностью управления (модификации) базовой стратегии поиска ЛВ (default strategy). Эта стратегия последовательно, без пропусков, перебирает подстановки, обеспечивая процессу поиска логических выводов свойство полноты. Модификации стратегии изменяют порядок перебора базовых подформул и вопросов, настраивая ЛВ на свойства решаемой задачи. Ряд таких модификаторов реализован, однако программы этих модификаторов представляют собой модули языка программирования С, что не обеспечивает должного уровня абстракции при программировании модификаторов.

В данной работе рассматривается реализация структур данных новой версии системы АДТ КВАНТ/3, выполненной в среде функционального языка программирования OCaml [7]. Язык OCaml является переемником базовых свойств языка ML [8], первоначально предназначенного для разработки систем АДТ. Языки класса ML прежде всего являются типизированными, т. е. обеспечивают проверку типа данных на этапе компиляции исходного кода программы. Типы данных и модули в языках поддерживают полиморфизм, т. е. структура типа может быть до конца не определена в одном из модулей программы и доопределена в другом модуле программы. Один из подклассов ML-языков поддерживает структуры данных с изменяемыми элементами (mutable fields), и, как следствие, автоматическую сборку мусора (garbage collection). Язык Ocaml принадлежит этому подклассу и, кроме того, поддерживает смешанную функционально-последовательную (императивную) модель исполнения программы, т. е. хотя язык в целом функиональный, некоторые подпрограммы могут быть реализованы в виде последовательности функций, операторов цикла и других императивных синтаксических конструкций. Компилятор языка OCaml порождает как байт-код виртуальной машины, используемый на этапе отладки программы, так и оптимизированный код процессора компьютера, сравнимый по производительности с программным кодом, порождаемым современными компиляторами языка программирования С. Перечисленные свойства делают привлекательным использование именно этого языка программирования для разработки новых версий систем АДТ, ориентированных на решение двух основных задач в исчислениях ПО-формул: повышение производительности систем поиска логических выводов и создание инструментальных средств для поддержки научных исследований в этой области.

1. Структуры данных системы КВАНТ/3

Основными требованиями к структурам данных в разработанной системе являются:

— разделение общих ресурсов, что позволяет уменьшить накладные расходы процессорного времени на копирование;

— возможность включать новые элементы, что позволит разрабатывать производные стратегии поиска ЛВ.

ПО-формула — это деревовидная структура; вершины трех первых уровней этого дерева имеют свои особенности. *Корень* дерева $\forall\,\mathrm{True}$ — первый уровень — всегда один и тот же для любой ПО-формулы; ему соответствует массив ссылок на вершины второго уровня. Вершины второго уровня — *базы* $\exists \bar{X}_i : A_i$, $i = \overline{1,n}$, $n \in N^+$, $N^+ = N \setminus \emptyset$, — в процессе построения ЛВ либо “накапливают переменные или атомы”, либо “множатся”, т. е. из одной базы, называемой далее *родительской*, на очередном шаге порождаются две и более дочерние базы. Порожденные базы обладают общими частями, унаследованными от исходной родительской базы. Вершины третьего уровня — *вопросы* — на

очередном шаге поиска ЛВ могут быть исключены из формулы. Кроме того, в вопросах, как правило, осуществляется хранение состояния перебора множества *ответных подстановок* [6]. К остальным поддеревьям ПО-формулы может быть применена ответная подстановка, что порождает новый частный случай этого поддерева. Новое поддерево в общем случае содержит оригинальные части исходного дерева и даже может с ним полностью совпадать.

В процессе поиска ЛВ существует необходимость менять порядок следования вершин-преемников в узлах первых двух уровней ПО-формулы, организовывать очереди и другие последовательности из этих вершин-преемников (поддеревьев). Для этого ветвление в этих вершинах реализовано в виде массивов с изменяющейся длиной. Для остальных поддеревьев, т. е. в узлах дерева четвертого и более уровня, рекурсивная списковая структура обеспечивает все необходимые свойства.

В таблице приведены основные структуры данных системы КВАНТ/3 в виде определений типов на языке OCaml. При помощи идентификаторов `'b` и `'q` обозначены элементы полиморфных типов данных, которые доопределяются при разработке производных систем АДТ на основе программных модулей КВАНТ/3. Эти структуры доопределяются на этапе конструирования ПО-формулы из отдельных компонентов.

Для представления списка переменных и типового условия используется общая структура — массив, называемый *пулом*. Элементом этого массива может быть константа, переменная, список или структура — сложный терм или атом. Имена констант, переменных, функций (функциональных символов) и атомов (предикатных символов), а также арность функциональных и предикатных символов хранятся в таблице символов. В ПО-формуле для указания имен элементов используются целочисленные ссылки в таблицу символов. Использование ссылок на таблицу символов повышает производительность операции сравнения элементов пула на этапе поиска подстановки. Аргументы сложных структур (функций или атомов) представляются при помощи целочисленных индексных ссылок на элементы пулов, из которых состоят эти стуктуры. Пример представления атома внутри пула приведен на рис. 1.

Массивы (таблица дескрипторов и пул) представлены в виде набора боксов с общими гранями, а списковые (рекурсивные) структуры — боксами, соединенными черточками. Ссылки на объекты показываются стрелками, начинающимися с жирных точек. Первый столбец таблицы дескрипторов содержит имена элементов, второй — типы, третий — арность. Аргументы структур данных в общем случае ссылаются как на пул, в котором находится структура данных, так и на другие пулы, например, предикат

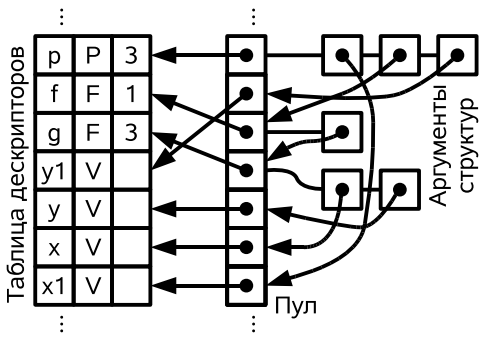

Рис. 1. Представление $p(x1, f(g(x, y)), y1)$ в виде пула

Основные структуры данных системы АДТ КВАНТ/3

Сущность	Представление в OCaml	Комментарий

Константное значение	`type const =`
	`  CI of int`
	`| CF of float`
	`| CS of string`
	`| CC of char`
		Константное значение — это объект OCaml, используемый системой АДТ в качестве константы

Дескриптор элемента типового условия; таблица символов	`type descr =  {`
	`kind: symbol_kind;`
	`sym: symbol;`
	`impl: impl_t; }`
	`and symbol_table= {`
	`hash: ... Hashtbl.t;`
	`descrs: descr array;}`
		Дескриптор определяет тип и название элемента типового условия `kind` ∈ {`None`, `Const`, `Var`, `Pred`, `Func`, `Def`}, `sym` — название элемента и его арность, `impl` — задаваемое ПО-поддерево, если `kind` = `Def`

Элемент типового условия: термы и атомы; массив термов (пул) `pool`	`type term =`
	`  DummyTerm`
	`| V of int`
	`| C of const`
	`| L of term list`
	`| F of int *`
	`  (int * pool) llist`
	`and pool = term array`
		Термы и атомы представляются одинаково, `DummyTerm` используется в процессе инициализации структур, использующих термы. Функциональные и предикатные символы `F`, константы `C` и переменные `V` идентифицируются индексом в таблице дескрипторов

Узел ПО-формулы	`type pcf = {`
	`tq: pool;`
	`fs: pcf list; }`
		Структуры представляют узлы дерева ПО-формулы четвертого и более уровня. Здесь `tq` — пул типового условия, `fs` — пул подформулы

База ПО-формулы	`type`
	`('b, 'q) b_struct = {`
	`idx: int;`
	`mutable btq: pool;`
	`pb: ('b, 'q) base;`
	`mutable qs:`
	`  'q questions;`
	`mutable bdata:`
	`  'b bdata_t; }`
	`and ('b, 'q) base =`
	`  Root`
	`| B of ('b, 'q)`
	`  b_struct`
		База ПО-формулы состоит из изменяющейся локальной части пула `btq`, ссылки на исходную базу `pb`, массива вопросов `qs` и данных пользователя `bdata`

Вопрос	`type 'q q_struct = {`
	`qtq: pool;`
	`mutable qdata:`
	`  'q qdata_t;`
	`qfs: pcf list; }`
		Вопрос включает в себя пул `qtq`, список ПО-подформул четвертого уровня и данные пользователя `qdata`

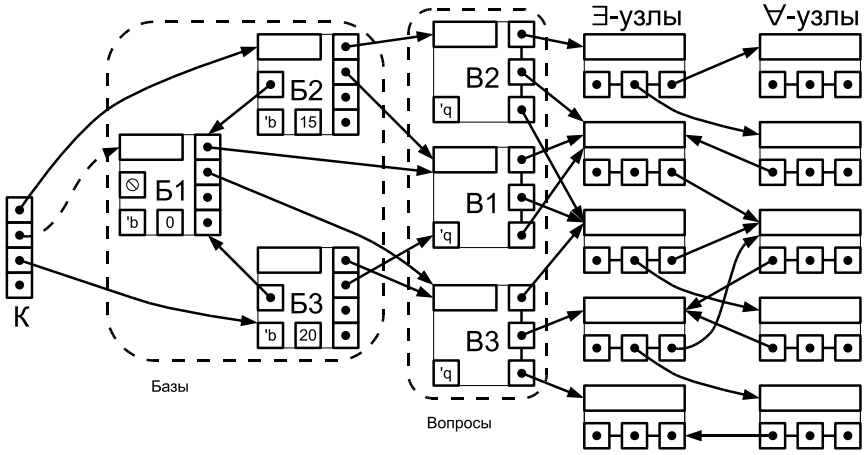

Рис. 2. Представление дерева ПО-формулы

из пула вершины-вопроса может использовать переменные, определенные в пуле базы. Если две структуры имеют в своем составе идентичные термы, то достаточно, чтобы структуры ссылались на одно и то же представление терма в пуле.

Индексы элементов пула используются для кодирования ответной подстановки [3, 6]. Подстановка — это отображение индекса пула вопроса на индекс элемента пула базы. Массив таких индексов кодирует подстановку элементов базы вместо соответствующих элементов вопроса. Ответная подстановка хранится в вопросе в поле `bdata`. Следует заметить, что структуры данных, приведенные в статье, не требуют от исследователя использовать именно этот вариант кодирования ответной подстановки.

На рис. 2 изображен пример представления дерева ПО-формулы. Корень дерева `K` — массив ссылок на базы; точки, из которых не выходят стрелки, соответствуют ссылкам на какие-либо другие части структуры ПО-формулы, не отображенные на рисунке. В структуру базы входят пул (обозначенный горизонтальным прямоугольником), индекс первого элемента пула (число в боксе) относительно родительской базы, ссылка на родительскую базу, данные исследователя `'b` и массив ссылок на вопросы. База `Б1` не имеет родительской базы, поэтому ее поле `idx` = `0`, а ссылка на родительскую базу пустая; эта ссылка обозначена перечеркнутой окружностью. Вопросы и остальные вершины дерева ПО-формулы состоят из пула и рекурсивного списка вершин-преемников. Вопрос `В1` унаследован базами `Б2` и `Б3` от родительской базы `Б1`, состояние перебора ответной подстановки в этом вопросе должно храниться отдельно для каждой ссылающейся на него базы.

Предлагаемый механизм наследования части пула между базами дает возможность использования следующей эвристики. Если в момент порождения дочерних баз не уничтожать ссылку из корня дерева на родительскую базу[1], как этого требует правило вывода, и, кроме того, построить опровержение (ЛВ) этой базы, то все ее дочерние базы тоже могут быть опровергнуты и удалены из корня дерева. Доказательство этого утверждения очевидно — опровержение родительской базы построено с пропуском всех шагов ветвления. Структуры данных из [6] делали невозможным реализацию такой эвристики поиска логических выводов.

[1]На рис. 2 эта связь изображена штриховой стрелкой.

Заключение

В рамках исследований новых машинно-ориентированных методов поиска логических выводов в исчислениях позитивно-образованных формул разработаны структуры данных, ориентированные на активное использование принципов разделения общих участков памяти компьютера. Предложена реализация основных структур данных для представления элементов ПО-формул.

На основе полученных структур реализовано разделение памяти между следующими элементами древовидного представления ПО-формулы:

— элементами типового условия: одинаковые термы разделяются использующими их атомами и термами;

— различными узлами дерева общих поддеревьев; вершина поддерева является разделяемой, если на нее ссылаются по крайней мере две вершины одного из предыдущих уровней ПО-дерева;

— пулами дочерних баз: часть пула родительской базы разделяется дочерними базами, при этом предлагается эвристика, повышающая производительность поиска ЛВ.

Другим аспектом исследования является предоставление исследователю возможности ввода новых элементов в структуры данных, что дает возможность разрабатывать новые стратегии поиска ЛВ, а также специализированные на конкретную задачу новые исчисления.

Использование предлагаемых структур данных в конечном счете должно привести к повышению быстродействия систем автоматического доказательства теорем в позитивно-образованных исчислениях, в частности, за счет снижения накладных расходов на копирование поддеревьев ПО-формулы по сравнению с ранее предложенными в [6].

Список литературы

[1] Vassilyev S.N. Machine synthesis of mathematical theorems // Logic Programming. 1990. Vol. 9, N 2–3. P. 235–266.

[2] Васильев С.Н., Жерлов А.К. Об исчислениях типовоквантворных формул // Докл. РАН. 1995. Т. 343, № 5. С. 583–585.

[3] Интеллектное управление динамическими системами / С.Н. Васильев, А.К. Жерлов, Е.А. Федосов, Б.Е. Федунов. М.: Физматлит, 2000.

[4] Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. М.: Наука, 1983.

[5] Robinson J.A. A machine-oriented logic based on the resolution principle // ACM J. 1965. N 12(1). P. 23–49.

[6] Черкашин Е.А. Программная система “Квант/1” для автоматического доказательства теорем: Дис. ... канд. техн. наук. Иркутск, 1999.

[7] Smith J.B. Practical OCaml. APress, 2006.

[8] Paulson L.C. ML for the Working Programmer. Cambridge Univ. Press, 1996.

Поступила в редакцию 25 января 2008 г.