© 2012 г. В.Э. МАЛЫШКИН, д-р техн. наук
(Институт Вычислительной математики и математической геофизики СО РАН)

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ И ПРЕДСТАВЛЕНИЕ АКТИВНЫХ ЗНАНИЙ[1]

Рассматриваются идеи параллельного программирования в приложении к формированию технологии накопления, обработки и использования активных знаний и предлагаются технологические модели и средства реализации.

V.E.Malyshkin. (Institute of Computational Mathematics and Mathematical Geophysics SB RAS, pr.Nauki 6, Novosibirsk, 630090, malysh@ssd.sscc.ru).
Applications of the parallel programming basic models to the accumulation, processing and utilization of active knowledge are considered. The models and means for implementation of the active knowledge are proposed

1. Введение

Параллельное программирование – это не только техника разработки быстро работающих программ. По разным причинам модели параллельных вычислений оказались полезны для решения многих задач. Отличительной чертой таких моделей является их технологическая реализуемость, т.е. возможность реализовать модель с приемлемым качеством.

Одна из целей статьи – показать возможные применения одной из таких моделей – вычислительных моделей – для построения технологии активных знаний. Ныне имеем две основные проблемы в работе со знаниями. Первая состоит в том, что существующие системы описания, хранения и обработки знаний – это системы ***пассивного*** знания, т.е. эти описания не могут быть применены непосредственно. В основном это тексты, книги, фильмы, описанные в них знания напрямую не применимы. Чтобы применить эти знания, надо стать специалистом: необходимо прочитать значительное количество книг из нужной предметной области, понять их содержание, научиться его правильно использовать. Всего, чтобы стать специалистом в сложной предметной области (каких сейчас большинство), человек учится примерно 25 лет = 10 лет в школе + 6 лет в университете + набирает после университет опыт практической работы в течении 10 лет прежде, чем становится более-менее самостоятельным работником. Так как в наше время технологии иной раз меняются в течении 2-5 лет, процесс обучения становится постоянным и непрерывным.

Вторая проблема состоит в том, что накопление знаний идет быстрыми темпами и объем накопленных знаний к настоящему времени столь велик, что уже накопленные знания нередко не используется, о них не знают и не понимают. Вообще пассивные знания, не встроенные в общую систему, легко могут теряться. Приходится нести большие затраты на сохранения, поиск и обработку знаний.

---

[1] Работа выполнена при финансовой поддержке РФФИ, грант № 10-07-00454-а

2. О модели знаний

Понятно как преодолеть эти проблемы. Необходимо определить технологически реализуемый способ (модель) описания, накопления и, главное, применения знаний в активной форме, без детального его изучения человеком. Главный вопрос – как это сделать. Разработанные методы логического синтеза программ предлагают такой способ [1-4]. В нем есть одна проблема – этот способ технологически не реализуем. Например, в нём невозможно за приемлемое время вывести алгоритм решения задачи, сформулированной в терминах теории, и сконструировать реализующую этот алгоритм программу, которая бы обладала заданным набором свойств типа выдавать результат с заданной точностью, быть оптимальной по каким-то критериям и т.п.

Уточним для текущего рассмотрения смысл термина «знание». Не вдаваясь в детали обсуждения таких понятий как знание, инженерия знаний и т.п. в наших рассуждениях ниже в качестве единицы знаний будем рассматривать программный модуль, исполнение которого будет иметь результатом применение знания. В таком модуле реализован некоторый алгоритм, реализующий знание. Например, знание арифметики сейчас реализовано в модуле, который называется *калькулятор* и с момента создания калькулятора человек может не знать алгоритмы арифметики и устный счет, как этого не знают нынешние школьники. Множество единиц знаний ещё не образуют базу знаний, потому что модули предметной обрасти находятся в некоторых отношениях друг с другом и, чтобы отразить эти отношения, на этом множестве необходимо навести соответствующую структуру.

Будем рассматривать программы (программные модули) как реализации знаний. Любые материальные реализации знаний не рассматриваются. Основными объектами манипуляции знаниями являются в нашем случае алгоритмы и программы.

Так как логический синтез программ в чистом виде оказался технологически не реализуем, то для решения проблем создания системы обработки активных знаний необходимо использовать другой подход. В качестве такового предлагается использовать метод структурного синтеза программ.

3. Логический синтез и структурный синтез программ

Известны два, в понятном смысле диаметрально противоположных, подхода к синтезу программ. Логический синтез программ базируется на достаточно полном описании предметной области (ПО), в котором можно сформулировать все необходимые утверждения, т.е. требуется наличие достаточно полной аксиоматической теории.

Трудности реализации логического синтеза программ заставляют пользоваться при описании ПО менее богатыми, но более удобными для машинного восприятия средствами. Одно из них заключается в наведении на ПО некоторой структуры, отражающей ассоциативные связи между понятиями ПО. Явное выражение этих связей позволяет свести случайный поиск к ассоциативному, что решающим образом сказывается на оценках трудоемкости процесса планирования. Кроме того, структурное описание имеет наглядное графическое представление. Описанный подход к синтезу программ получил название *структурного синтеза* программ (синтез программ на вычислительных моделях) [5-14].

В методе структурного синтеза параллельных программ о полноте теории вообще не заботятся. В нем принципиально в описание предметной области включается лишь то, что уже существует, что уже реализовано с приемлемым качеством на практике, т.е. ме-

тод базируется на тщательно сконструированном частичном описании предметной области, на частично определенной аксиоматической теории.

Задача ставится следующим образом:

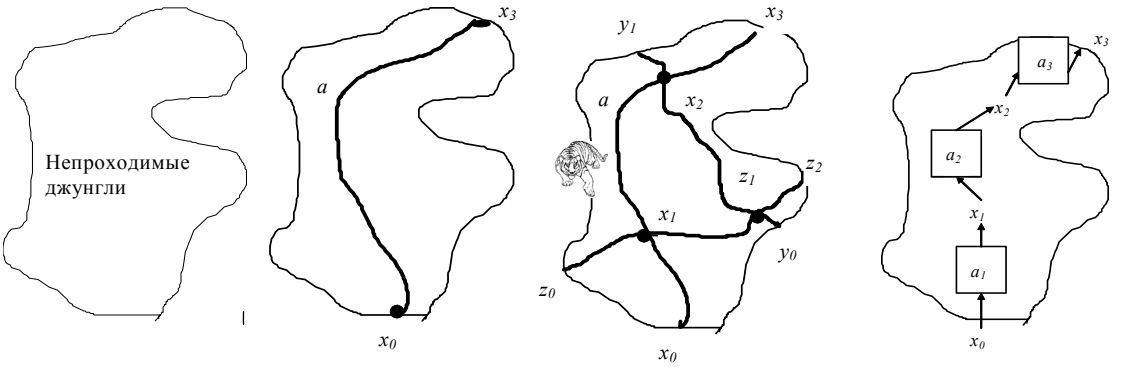
Пусть даны:

- класс S спецификаций входных заданий;
- класс P результирующих выходных программ;
- отношение эквивалентности ~ на P;
- отношение качества > на P, удовлетворяющее аксиомам частичного порядка.

Требуется найти алгоритм $A{:}S{\to}P$ такой, что результат синтеза - программа $p{=}A(s)$, $p{\in}P$, - удовлетворяет спецификации $s{\in}S$ и является наилучшей в смысле > в классе программ $\{p|p{\sim}\mathrm{A(s)}\}$.

Таким образом, проблема синтеза заключается в разработке алгоритма, конструирующего по каждому элементу из S некоторый элемент из P, наилучший среди всех программ, решающих специфицированную задачу.

Образно, идея структурного синтеза программ может быть продемонстрирована на следующем несерьезном примере. Представим себе непроходимые джунгли (рис.1.а). Для “автоматического” пересечения джунглей из пункта x_0 в пункт x_3 необходимо тщательное, во всех мыслимых деталях, описание джунглей (нужна достаточно полная теория в логическом программировании). На практике рассчитывать на наличие такого знания изначально невозможно, никто его и не пытается создать. Поэтому, если понадобится ходить из пункта x_0 в пункт x_3, то вначале первопроходцы, не заботясь о детальном изучении джунглей, проложат тропинку a из x_0 в x_3 (рис.1.б) на основе имеющихся знаний. При совершении этого подвига некоторые из них, возможно, будут съедены дикими зверями или утонут в трясинах, но после них и все прочие люди смогут пользоваться тропинкой, имея лишь минимально необходимые знания о джунглях. Затем таким же образом могут появиться тропинки из y_0 в y_1 и из z_0 в z_2 (рис.1.в). Но одновременно появился и путь $x_0x_1z_1z_2$. А из x_0 в x_3 ведут теперь два пути: $x_0x_1x_2x_3$ и $x_0x_1z_1x_2x_3$. И второй, более длинный, путь вполне может оказаться предпочтительнее первого, более короткого, например, в случае, когда на участке x_1x_2 путника поджидает проголодавшийся тигр.



а) б) в) г)

Рис.1

Тропинки служат здесь образами алгоритмов. Тропинке a соответствует последовательность шагов (операций) алгоритма (рис.1.г). Операция a_1 вычисляет значение переменной

x_1 из значения переменной x_0 и т. д. Операция может реализоваться в программе процедурой или фрагментом кода. Тогда переменная x_3 вычислится из x_0 последовательностью операций $x_0 \to a_1 \to x_1 \to a_2 \to x_2 \to a_3 \to x_3$. Пусть аналогично, значение переменной z_2 из значения z_0 (тропинка $z_0x_1z_1z_2$) вычисляется последовательностью операций $z_0 \to b_1 \to x_1 \to b_2 \to z_1 \to b_3 \to z_2$, а значение переменной y_1 из значения переменной y_0 (тропинка $y_0z_1x_2y_1$) — последовательностью $y_0 \to c_1 \to z_1 \to c_2 \to x_2 \to c_3 \to y_1$. Тогда переменная $x3$ может быть вычислена двумя разными алгоритмами, соответствующим двум возможным путям $x_0x_1x_2x_3$ и $x_0x_1z_1x_2x_3$, т.е. алгоритмами $x_0 \to a_1 \to x_1 \to b_2 \to z_1 \to c_2 \to x_2 \to a_3 \to x_3$ и $x_0 \to a_1 \to x_1 \to a_2 \to x_2 \to a_3 \to x_3$. На этом выборе основано конструирование оптимального алгоритма.

Вычислительная модель является базой знаний, которая содержит множество алгоритмов/программ, причем хороших алгоритмов (как тропинки в джунглях не прокладываются плохо, так и в вычислительных моделях накапливаются только хорошие алгоритмы). И комбинации хороших алгоритмов (путь $x_0x_1z_1x_2x_3$ в джунглях) тоже могут быть хороши. Они хотя и не обязательно оптимальны, но и не самые худшие. Задача вывода приемлемого алгоритма становиться простой и сводится к ограниченному управляемому перебору на графе.

В дополнение к этому, так же как массив джунглей разбиваются тропинками на фрагменты, так и описание предметной области разбивается в вычислительных моделях на множество меньших предметных областей, для которых построение более или менее полных теорий (для каждой подобласти своей) более вероятно. Таким образом, в вычислительных моделях формальное описание предметной области строится как система теорий, связанных соотношениями модели.

4. Вычислительные модели

Посмотрим, как строится вычислительная модель (ВМ)

Пусть предметная область есть геометрия и необходимо описать понятие треугольник (рис.2). Напомним, что ключевая идея ВМ состоит в накоплении множества различных алгоритмов предметной области и их взаимосвязей. Основными величинами, характеризующими треугольник, являются: x, y, z - стороны; p - полупериметр; α, β, γ - углы; h_x, h_y, h_z - высоты треугольника, опущенные на стороны x, y, z; s - площадь. Треугольник характеризуют и многие другие переменные и их взаимосвязи, но на текущий момент, к примеру, они не используются и потому не включаются в ВМ.

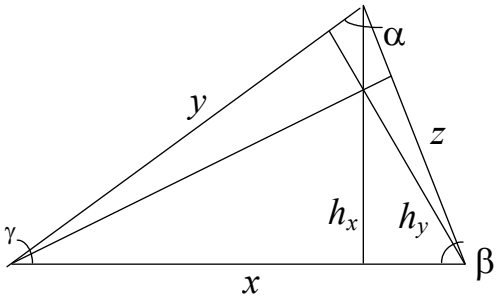


Рис. 2

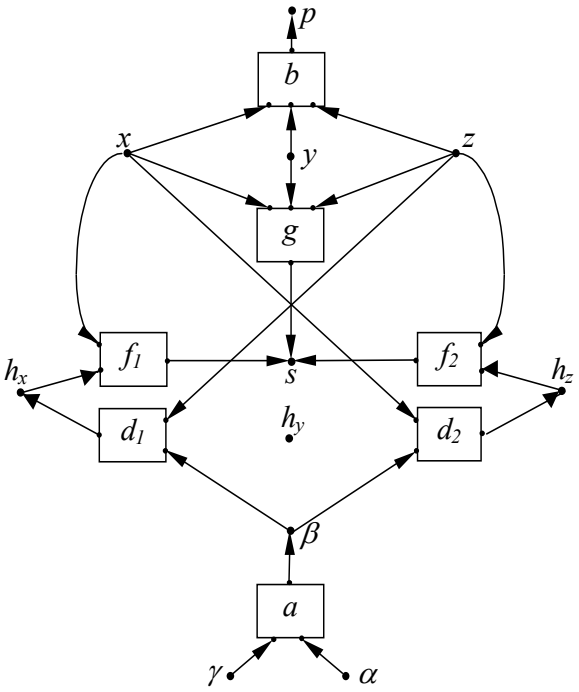


Рис. 3

Проанализируем ряд основных закономерностей, связывающих эти величины. Из формулы $\alpha+\beta+\gamma=180^\circ$, разрешая один из аргументов относительно двух других и вводя обозначение $a(X,Y)=180^\circ-X-Y$, получим

$$\alpha=a(\beta,\gamma),\ \beta=a(\alpha,\gamma),\ \gamma=a(\alpha,\beta).$$

Аналогично, используя формулу $p=(x+y+z)/2$, имеем

$$p=(x+y+z)/2=b(x,y,z);$$
$$x=2p-y-z=c(p,y,z);$$
$$y=2p-x-z=c(p,x,z);$$
$$z=2p-x-y=c(p,x,y),$$

где $b(X,Y,Z)$ обозначает функцию $(X+Y+Z)/2$, а $c(X,Y,Z)$ - функцию $2X-Y-Z$.

Выражение высот через стороны приводит к следующим соотношениям:

$$h_x=y\cdot\sin\gamma=d(y,\gamma)=z\cdot\sin\beta=d(z,\beta);$$
$$h_y=x\cdot\sin\gamma=d(x,\gamma)=z\cdot\sin\alpha=d(z,\alpha);$$
$$h_z=x\cdot\sin\beta=d(x,\beta)=y\cdot\sin\alpha=d(y,\alpha).$$

К ним следует добавить все обратные соотношения вида $y=h_x/\sin\gamma$; $\gamma=\arcsin(h_x/y)$ и т. д.

Наконец, площадь может быть получена по формулам

$$s=xh_x/2=f(x,h_x)=yh_y/2=f(y,h_y)=zh_z/2=f(z,h_z)=$$
$$\sqrt{p(p-x)(p-y)(p-z)}=g(x,y,z).$$

(В последнем обозначении переменная p исключена, так как ее можно выразить через x, y, z). Конечно, выписанный список закономерностей, как и используемых параметров треугольника, не является полным, но он достаточен для иллюстративных целей.

Приведенные соотношения показывают принципиальную возможность вычисления одних параметров треугольника через другие. Для реализации этой возможности необходимо написать и включить в библиотеку программ соответствующие программные модули. Для нашего примера предположим, что на каком-то этапе эксплуатации в библиотеку включены модули для вычислений по следующим формулам:

$$p=b(x,y,z);\ s=g(x,y,z);\ s=f(x,h_x);$$

$$s=f(z,h_z);\ \ h_x=d(z,\beta);$$

$$h_z=d(x,\beta);\ \beta=a(\gamma,\alpha).$$

Эта ВМ изображена графически на рис.3. Заметим, что в списках модулей отсутствуют соотношения $\gamma=a(\alpha,\beta)$, $\alpha=a(\beta,\gamma)$, а также аналогичные соотношения для модулей *f* и *d*, хотя другие соотношения для этих модулей включены, а, следовательно, эти модули должны быть программно реализованы. Кроме того, на рис.3 имена модулей *f* и *d* имеют индексы. Все это является следствием определения ВМ. Для технических удобств ВМ определена так, что все операции имеют различные имена. Таким образом, операции для вычисления $s=f(x,h_x)$, $s=f(y,h_y)$, $s=f(z,h_z)$ считаются различными, несмотря на то, что отличаются лишь входными переменными.

Теперь можно попробовать сформулировать и решить задачу на этой модели. Пусть есть возможность измерить две стороны треугольника *x*, *z* и два противолежащих им угла α и β. Зная эти величины, требуется найти площадь фигуры. Это стандартная постановка задачи на вычислительной модели:

- известны значения переменных из некоторого подмножества *V*, здесь $V=\{x,z,\gamma,\alpha\}$;

- требуется найти значения переменных из множества *W*, здесь $W=\{s\}$.

Анализируя структуру связей модели, видим, что с помощью модуля *a* можно вычислить из γ и α переменную β; модули d_1 и d_2 позволяют вычислить переменные h_z и h_x из переменных *x*, *z*, β; а *s* можно найти либо из h_x и *x*, либо из *z* и h_z с помощью модулей f_1 и f_2. Таким образом, алгоритм вычисления *s* можно представить функциональными термами

$$t_1=f_1(x,d_1(z,a(\gamma,\alpha)));\ t_2=f_2(z,d_2(x,a(\gamma,\alpha)));$$

Смысл всех объектов этой ВМ - переменных и операций - понятен (они интерпретированы) для нас. Закономерности геометрии нашли свое отражение, во-первых, в синтаксисе модели, т.е. в характере связей между операциями и переменными, во-вторых, в семантике, т.е. в специфике этих операций.

Когда ВМ построена и надо найти цепочку операций для вычисления одной группы переменных из другой, тогда ее семантические особенности уже не важны. В процессе планирования - построения цепочки - можно учитывать только синтаксис модели, абстрагируясь от специфики операций. В общем случае, одна и та же ВМ может соответствовать различным предметным областям и только сопоставление операциям различных конкретных функций (процедур) определит ее специфику. Таким образом, ВМ является неинтерпретированным объектом. Именно абстрагирование от конкретной интерпретации позволяет с помощью лишь структурной составляющей создавать универсальные алгоритмы синтеза программ с реальными оценками сложности. Это и определяет технологическую реализуемость ВМ

5. Заключительные замечания

Таким образом, используя ВМ, можно обеспечить накопление, обработку и применение активных знаний в формализованных предметных областях, в математике, в её разнообразных приложениях. Это, в свою очередь, позволит создавать интеллектуальные системы, т.е. системы, которые действуют, опираясь на знания в активной форме и которые могут быть специализированы подбором подходящей базы знаний.

Можно высказать предположение, что если сейчас большое число, если не большинство, приложений компьютеров связано с разработкой математических моделей различных явлений, сложных процессов, устройств и технологий, то в следующие десятилетия все больше будем заниматься разработкой теоретических и технологических проблем создания и накопления баз активных знаний.

И последнее замечание. Нынешняя грамотность человека на основе фонетического письма обеспечила ему описание, накопление и применение знаний в пассивной форме. Такая грамотность в большинстве случаев позволяют человеку освоить и применять знания только в одной предметной области, например, быть хорошим музыкантом и никаким инженером. На большее ни сил ни времени не хватает. Грамотность на основе активных знаний невиданно расширит возможности освоения человеком новых знаний, кардинально изменит все наши промышленные и научные технологии и видимо изменит и самого человека.

СПИСОК ЛИТЕРАТУРЫ

1. *Z.Manna, R.Waldinger.* Synthesis: dreams $\Rightarrow$ *programs*// IEEE Tr. on SE, 1979, Vol. SE-5, pp. 294-398.
2. *Genesereth, M.R. and Nilsson, N.J.* Logical Foundation of Artificial Intelligence. - Morgan Kaufmann, Los Altos, California, 1987.
3. *Francis Giannesini, Henry Kanoui, Robert Pasero, Michel van Caneghem*. Prolog. International Computer Science Series, Addison-Wesley, 1986
4. *Yu.L.Ershov, S.S.Goncharov and Sviridenko D.I.* Semantic programming // Information Processing: Proc. IFIP 10th World Comput. Congr. Ser. 10. — Amsterdam, 1986. — P. 1113–1120
5. *В.А.Вальковский, В.Э.Малышкин.* Синтез параллельных программ и систем на вычислительных моделях // Наука, Новосибирск, 1988, 128 стр.
6. *Malyshkin V.* How to Create the Magic Wand. Currently Implementable Formulation of the Problem // In the Proceedings of the 5-th Int. Conference, series of Proceedings on New Trends in Software Methodologies, Tools and Techniques. Frontiers in Artificial Intelligence and Applications, IOS Press, Vol. 147, pp. 127-132. 28-30 September 2006, Quebec, Canada.
7. *Victor Malyshkin*. Magic Wand Approach to Representation of Personal Technologies. // In the Proceedings of the 9th SoMeT_10, 2010, series of Proceedings on New Trends in Software Methodologies, Tools and Techniques. Frontiers in Artificial Intelligence and Applications, IOS Press, Vol. 217, pages 59-69, September 29 – October 1, 2010, Yokohama, Japan.
8. *I.B.Zadykhailo*. Составление циклов по параметрическим записям специального вида. – Журнал вычислительной математики и математической физики. - 1963, том 3, №2, стр. 337-357

9. *A.N.Andrianov, K.N.Efimkin, I.B.Zadykhailo*. Nonprocedural Language for Mathematical Physics // Programming and Computer Software, V.17, No.2, 1992, pp.10-22
10. *A.N.Andrianov, K.N.Efimkin, V.Y.Levashov, I.N.Shishkova*. The Norma language application to solution of strong nonequilibrium transfer process problem with condensation of mixtures on the multiprocessor system // Computational Science - ICCS 2001, Lecture Notes in Computer Science, V.2073, p.502-510, May, 2001
11. *V.Malyshkin*. Assembling of Parallel Programs for Large Scale Numerical Modeling. – In the Handbook of Research on Scalable Computing Technologies. // IGI Global, USA, 2010, 1021 pp, Chapter 13, pp. 295 – 311. ISBN 978-1-60566-661-7
12. *S.Kireev and V.Malyshkin* Fragmentation of numerical algorithms for parallel subroutines library - The Journal of Supercomputing, Springer, Volume 57, Number 2 / August 2011 pp. 161-171
13. *Malyshkin V.E., Perepelkin V.A*. LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem - In: Proceedings of the 11th Conference on Parallel Computing Technologis, LNCS 6873 - pp. 53-61, Springer, 2011
14. *Victor Malyshkin and Vladislav Perepelkin*. Optimization methods of parallel execution of numerical programs in the LuNA fragmented programming system // The Journal of Supercomputing, Special issue on Enabling Technologies for Programming Extreme Scale Systems, Volume 61, Number 1 (2012), pp. 235-248, DOI: 10.1007/s11227-011-0649-6