СТРУКТУРНЫЙ СИНТЕЗ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

В.Э. Малышкин

Институт вычислительной математики
и математической геофизики СО РАН,
Новосибирск

Метод структурного синтеза программ имеет в настоящее время все шансы для хорошей реализации на нынешних мультикомпьютерах. В первую очередь потому, что способ представления знаний в нём – вычислительные модели, семантические сети – хорошо соответствует структуре мультикомпьютеров.

В зависимости от характера формализации ПО можно выделить несколько подходов к проблеме синтеза программ. Наиболее цельный из них в теоретическом отношении – *логический синтез* программ. При этом подходе формальным описанием ПО является некоторое достаточно полное математическое исчисление, представляющее закономерности предметной области. Задача синтеза ставится как задача нахождения доказательства теоремы существования, а программа извлекается из этого доказательства. К сожалению, этот подход весьма сложно реализуем на практике, так как, с одной стороны, для нетривиальных предметных областей создание полной непротиворечивой теории не осуществимо. С другой стороны, еще недостаточно развиты методы машинного доказательства теорем, особенно если ищется программа, обладающая нужными свойствами, например оптимальная по какому-либо критерию.

Эти трудности 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, удобное для восприятия и обработки. Описанный подход к синтезу программ получил название *структурного синтеза* программ.

Проблема синтеза программ на вычислительных моделях рассматривается в статической постановке: по представлению вычислительной модели, выделенной группе переменных, служащих для ввода начальных данных, и другой группе переменных, служащих для вывода результатов, требуется найти программу, вычисляющую по заданным входным данным

соответствующие результаты (если они для этих входных данных определены).

При одной и той же постановке задачи можно синтезировать различные программы.

Принципиально важно, что синтезируются программы наилучшего качества. Основными критериями качества являются надежность, оптимальность (в ряде смыслов), степень параллелизма и т.п. В целом в процессе синтеза программы решается несколько задач, укладывающихся в следующую общую схему.

Пусть даны:

– класс S спецификаций входных заданий;

– класс P результирующих выходных программ;

– отношение эквивалентности $\sim$ 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 $\{p|p\sim A(s)\}$.

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. Метод синтеза параллельных программ на вычислительных моделях отличает высокая прагматичность. Прагматичность метода основана на том, что вычислительная модель является, собственно, собранием хороших решений различных задач предметной области и решение новой задачи собирается (компонуется) из заготовленных хороших решений. В таких условиях есть надежда сконструировать на их основе хорошее решение новой задачи. Конечно, иногда для этого приходится включать в вычислительные модели новые необходимые решения.

Базой знаний в вычислительных моделях является множество алгоритмов и фрагментов программ, причем хороших алгоритмов. И комбинации хороших алгоритмов тоже могут быть хороши. Они хотя и не обязательно оптимальны, но и не самые худшие. Задача вывода приемлемого алгоритма становится простой и сводится к ограниченному управляемому перебору на графе. Однако качество сконструированной программы, разнообразие программ существенно зависят от размеров фрагментов. С уменьшением размеров фрагментов растет разнообразие и теоретически достижимое качество программ (увеличивается множество заготовленных алгоритмов) и одновременно быстро растёт сложность конструирования программы (растет число комбинаций алгоритмов), так что на текущий

момент приходится использовать весьма большие фрагменты и небольшой перебор.

В дополнение к этому, так же, как массив джунглей разбивается 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 формализованное описание предметной области строится как система теорий, связанных соотношениями модели.

Основные определения

Пусть заданы:

1) конечное множество $\boldsymbol{X}=\{x, y, ..., z\}$ переменных для представления вычисляемых и измеряемых величин;

2) конечное множество $\boldsymbol{F}=\{a, b, ..., c\}$ функциональных символов (операций) арности $m\times n$, $m\geq 0$, $n\geq 0$, m и n, вообще говоря, различные для разных операционных символов;

3) с каждым символом операции a арности $m\times n$ связан набор $\mathrm{in}(a)=(x_1, ..., x_m)$ входных и набор $\mathrm{out}(a)=(y_1, \ldots y_n)$ выходных переменных, при этом $i\neq j \rightarrow y_l \neq y_j$.

Пара $\boldsymbol{C}=(\boldsymbol{X},\boldsymbol{F})$ называется *простой вычислительной моделью* (ПВМ). Операция $a\in\boldsymbol{F}$ описывает возможность вычисления переменных $\mathrm{out}(a)$ из переменных $\mathrm{in}(a)$ с помощью некоторой процедуры. Модели удобно изображать графически (рис. 1).

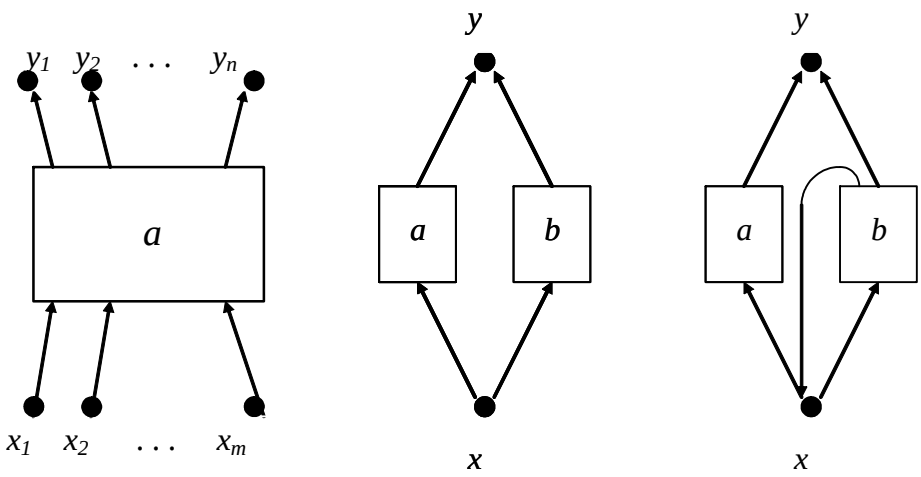


Рис. 1. Определение множества термов $T(V,F)$, порожденных V и F ($V\subseteq\boldsymbol{X}$, $F\subseteq\boldsymbol{F}$)

Основным объектом теории является функциональный терм. Формальное определение терма и его атрибутов задается следующими правилами.

1) Если $x \in V$, то x есть терм t, $x \in T(V,F)$.

2) Пусть t^1, ..., t^s входят в $T(V,F)$. Пусть $a \in F$, $\text{in}(a)=(x_1,...,x_s)$. Тогда терм t, равный $a(t^1,...,t^s)$, включается во множество $T(V,F)$ тогда и только тогда, когда выполняется $\forall i\ (x_i \in \text{out}(t^i))$.

Интерпретация

Пусть $V \subseteq \boldsymbol{X}$ – некоторое подмножество переменных, играющее роль входных переменных алгоритма. *Интерпретация* $\boldsymbol{I}$ в области интерпретации D есть функция, которая сопоставляет:

– каждой переменной $x \in V$ элемент $d_x=\boldsymbol{I}(x) \in D$, d_x называется значением переменной x в интерпретации $\boldsymbol{I}$ (далее просто значение);

– каждой операции $a \in \boldsymbol{F}$ вычислимую функцию f_a: $D^m \to D^n$, $\text{in}(a)=\{x_1, x_2, ..., x_m\}$, $\text{out}(a)=\{y_1, y_2, ..., y_n\}$;

– каждому терму $t=a(t_1,t_2,...,t_m)$ суперпозицию функций в соответствии с правилом $\boldsymbol{I}(a(t_1,t_2,...,t_m))=f_a(\boldsymbol{I}(t_1),\boldsymbol{I}(t_2),...,\boldsymbol{I}(t_m))$.

Пусть $t=a(t_1,t_2,...,t_m)$ – произвольный терм такой, что $\text{in}(a)=\{x1, x2, ..., xm\}$, $\text{out}(a)=\{y1, y2, ..., yn\}$. Тогда набору переменных $\text{out}(a)$ сопоставляется в интерпретации $\boldsymbol{I}$ набор (кортеж) значений

$$val(t)=(d_1,d_2,...,d_n)=f_a(val_{x1}(t_1),val_{x2}(t_2),...,val_{xn}(t_n)).$$

Здесь $val_z(t)$ обозначает значение того компонента набора $val(t)$, который соответствует переменной z.

Впредь всегда предполагается, что каждой функции $f_a=\boldsymbol{I}(a)$ соответствует модуль mod_a, который и будет использоваться в программе для вычисления f_a. Если существуют два различных алгоритма вычисления переменной y (существуют два различных терма t_1 и t_2 таких, что $y \in \text{out}(t_1) \cap \text{out}(t_2)$, $\text{in}(t_1) \cup \text{in}(t_2) \subseteq V$) из одних и тех же начальных переменных V, то по смыслу модели эти два значения переменной y должны быть эквивалентны (равны в простейшем случае). Например, площадь треугольника может быть вычислена различными алгоритмами (различными термами), однако вычисленные значения должны быть равны (если, конечно, начальные данные действительно определяют треугольник и запрограммированные модули верно вычисляют функции). Это одна из причин, по которой следует строить избыточные вычислительные модели, в которых существует несколько алгоритмов вычисления переменных.

Дальше будем работать только с такими *корректными* интерпретациями. По определению, в корректной интерпретации для любой переменной y и любых термов t_1 и t_2 таких, что $y \in \text{out}(t_1) \cap \text{out}(t_2)$, значения $val_y(t_1)=val_y(t_2)$, т.е. оба терма *вырабатывают* одно и то же значение переменной y. Это позволяет говорить о значениях переменных, вычисляемых не только отдельными термами, но и множествами термов.

Все термы из множества $T(V,F)$ обладают свойством: $t\in T(V,F)\rightarrow \text{in}(t)\subseteq V$. Термы множества $T(V,F)$ определяют все вычисления (все алгоритмы), которые могут быть произведены из переменных множества V (если заданы их значения).

Определение. Терм называется *избыточным*, если на пути от входных переменных к вершине терма (на ветви расширенного дерева, представляющего терм) одна и та же операция встречается более одного раза для вычисления одной и той же своей выходной переменной.

Множество всех не избыточных термов из $T(V,F)$ обозначается $T1$, $T1\subseteq T(V,F)$. Впредь будем работать только с термами из $T1$.

Определение. Множество термов $T_V^W=\{t\in T1 \mid \text{out}(t)\cap W\neq\varnothing\}$. Это множество задает все вычисления, которые основаны на V и завершаются в W.

Определение. Множество термов $R\subseteq T_V^W$ такое, что $\forall x\in W$ $\exists t\in R(x\in\text{out}(t))$ называется (V,W)-планом вычислений.

Такой (V,W)-план вычислений определяет алгоритм решения задачи: на C вычислить W из V.

Планирование алгоритма

Пусть задана ПВМ $\boldsymbol{C}=(\boldsymbol{X},\boldsymbol{F})$, которая после трансляции представлена в виде двух таблиц *ТХ* и *ОП*. Каждая строка таблицы *ТХ* имеет вид $(x,A(x),\text{comp}(x))$, а таблицы *ОП* – $(a,\text{in}(a),\text{out}(a))$, $x\in\boldsymbol{X}$, $a\in\boldsymbol{F}$, $\text{comp}(x)=\{a\in\boldsymbol{F} \mid x\in\text{out}(a)\}$, $A(x)=\{a\in\boldsymbol{F} \mid x\in\text{in}(a)\}$.

Чтобы не делать далее несущественных оговорок, предполагается, что $\text{in}(a)\neq\varnothing$ для любых $a\in\boldsymbol{F}$. Алгоритм планирования состоит из двух частей: восходящей и нисходящей.

В *восходящей* части алгоритма строятся множества переменных и операций, используемых в термах из множества $T_V=T(V,F)$.

Обозначим $V_0=V$, тогда

$$F_0=\{a\in\boldsymbol{F} \mid \text{in}(a)\subseteq V_0\}=\bigcup_{x\in V_0}\{a\in A(x) \mid \text{in}(a)\subseteq V_0\}$$

содержит все операции ПВМ такие, что $\text{in}(a)\subseteq V_0$. Далее формируется множество $V_1=\{x\in\boldsymbol{X} \mid x\in\text{out}(a)\wedge a\in F_0\}\cup V_0$, на основе V_1 строится множество

$$F_1=\bigcup_{x\in V_1\setminus V_0}\{a\in A(x) \mid \text{in}(a)\subseteq V_1\}$$

и т. д. до тех пор, пока при некотором целом положительном k не окажется, что $F_k=\varnothing$. На этом завершается восходящая часть алгоритма планирования. Множества V_i и F_I, $i=0,...,k$ содержат все переменные и операции, используемые в термах из множества T_V.

Если $W\not\subseteq V_k$, то планирование можно прекращать, так как в этом случае существует переменная в W, которая не вычисляется никаким термом

из множества T_V, и, следовательно, не существует алгоритма решения сформулированной задачи на основе имеющихся знаний о ПО. В этом случае говорим, что сформулированная задача синтеза *не разрешима*. В противном случае можно начать строить множества переменных и операций, используемых в термах из T_V^W. Обозначим $F^*=\bigcup_{i=0} F_i$ и определим множества

$$G_i = \bigcup_{x \in H_{i-1}} \left\{ a \in F^* \middle| a \in \mathrm{comp}(x) \wedge a \notin \bigcup_{m=1}^{i-1} G_m \right\}, \quad H_i = \bigcup_{a \in G_i} \mathrm{in}(a).$$

Построение множеств G_I и H_i завершается, когда при некотором целом положительном r окажется $G_r = \varnothing$. Множества G_I и H_i, $i = 1, ..., r$ содержат все переменные и операции, используемые в термах из множества T_V^W. Кроме того, в них остаются операции и переменные, необходимые для построения некоторых дублирующих вычислений.

Таким образом, результатом планирования является ПВМ, оставшаяся от ***C*** после удаления из *ТХ* и *ОП* «лишних» переменных и операций. Множество T_V^W не строится, подходящий в некотором смысле (V,W)-план T строится в каждом конкретном случае процедурой выбора алгоритма.

В случае, когда $W \not\subseteq V_k$, сформулированная задача синтеза оказывается неразрешимой и необходимо изменить формулировку задачи, т. е. либо уменьшить W, удалив из него невычислимые переменные, либо расширить V, включив в него такие новые переменные, что станут вычислимыми все переменные из W. Для уменьшения затрат на расширение V может быть использован алгоритм планирования. Для этого необходимо выполнить его *нисходящую* часть из множества переменных $W'=W\backslash V_k$ с использованием всех операций из F. Все переменные из построенных при этом множеств H_i, i=1, 2, ..., r являются кандидатами на включение в V. Из них человек может выбрать те переменные, значения которых ему доступны. Нетрудно также построить человеко-машинные алгоритмы, помогающие сделать такой выбор.

Как можно было бы определить вычисления на вычислительной модели? Как определено интерпретацией. Вначале вносятся значения во все переменные из V. Затем выполняются все операции из F_1 (выполняются модули mod_a для всех $a \in F_1$), при этом (в соответствии с определением интерпретации) вычисляются значения переменных из $V_1\backslash V_0$. Далее выполняются все операции из F_2 и т.д. до F_k. Ясно, что такие вычисления могут быть весьма неэффективными, поскольку будут реализованы все алгоритмы вычисления W, которые определены моделью, – и самые хорошие и самые плохие. Кроме того, могут быть вычислены и переменные, которые не входят в W и которые не надо вычислять вообще.

Основной смысл планирования именно в том и состоит, чтобы до начала вычислений статическим анализом

а) удалить операции и переменные, не участвующие в вычислении *W*,

и

б) подготовить информацию для конструирования приемлемого по качеству или даже оптимального алгоритма вычисления *W*.

Это, конечно, существенная и высокоэффективная оптимизирующая процедура. Известно, что наибольший оптимизационный эффект при построении решения задачи извлекается именно на этапе конструирования алгоритма. На всех последующих этапах эффект оптимизации значительно ниже.

Литература

1. Вальковский В.А., Малышкин В.Э. Синтез параллельных программ и систем на вычислительных моделях. – Новосибирск: Наука, 1988. – 128 с.