УДК 519.17:519.683.8

В.В. Кручинин

АЛГОРИТМЫ И ПЕРЕЧИСЛИТЕЛЬНЫЕ СВОЙСТВА ДЕРЕВЬЕВ И/ИЛИ

Рассматриваются вопросы использования деревьев И-ИЛИ для построения алгоритмов генерации информационных объектов. Описываются перечислительные свойства деревьев, приводятся алгоритмы генерации вариантов и определения их свойств. Предлагается метод построения алгоритмов генерации информационных объектов.

Генераторы информационных объектов становятся необходимым элементом сложных программных систем [1]. Генераторы сценариев, карт местности, изображений, тестовых заданий и вопросов – вот неполный перечень таких программ. Математической основой построения таких генераторов является, как правило, модель предметной области и датчик случайных чисел. Однако исследования в области перечислительной комбинаторики показали, что для комбинаторных объектов можно построить алгоритмы генерации, которые по заданному номеру объекта строит сам объект. Примерами таких алгоритмов являются алгоритмы генерации перестановок и сочетаний [2, 3].

Ниже предлагается рассмотреть подход к построению таких алгоритмов, основанных на использовании деревьев И/ИЛИ. Рассматриваются перечислительные свойства и алгоритмы.

ДЕРЕВЬЯ И/ИЛИ

Деревья И/ИЛИ, понятие которого впервые было предложено Слейглом [4], являются важным инструментом исследования и создания систем искусственного интеллекта [5–7]. Дерево И-ИЛИ содержит два типа узла: И-узел и ИЛИ-узел. В терминах решения задачи И-узел означает, что решение задачи разбивается на подзадачи. Решение всей задачи зависит от решения всех подзадач. ИЛИ-узел означает, что задача может быть решена несколькими методами. Соответственно для решения задачи, представленного ИЛИ-узлом, необходимо использовать какой-то один метод. Существуют и другие интерпретации узлов дерева И/ИЛИ, например, И-узел описывает структуру некоторой системы, подсистемы, блока и т.д., а ИЛИ-узел – некоторое множество типов структур.

Вариантом дерева И/ИЛИ назовем поддерево, которое получается из заданного путем отсечения выходных дуг кроме одной у всех ИЛИ-узлов. Вариант в терминах решения задачи задает одно из возможных решений задачи.

Алгоритм 1 получения варианта V дерева И/ИЛИ D следующий:

1. Корнем варианта становится корень (root) дерева D и $Stack \xrightarrow{push} root$.

2. Если стек пуст, завершаем работу алгоритма. Иначе вытаскиваем узел из стека и делаем его текущим $Stack \xrightarrow{pull} z$.

3. Если рассматриваемый узел z – ИЛИ-узел, то в вариант V заносится один s_j из сыновей узла z.

4. Если рассматриваемый узел z – И-узел, то все его сыновья s_j заносятся в вариант.

5. Если рассматриваемый узел z – лист, то происходит переход на шаг 2.

Все листья варианта V являются подмножеством множества листьев дерева И/ИЛИ D. Очевидно, что вариантов в дереве И/ИЛИ может быть некоторое множество. В связи с этим возникает задача подсчета количества вариантов в дереве D.

Рассмотрим алгоритм подсчета вариантов решений в дереве И/ИЛИ. Для этого запишем следующую рекурсивную функцию:

$$\omega(z)=\begin{cases}\sum_{i=1}^{n}\omega(s_i^z)\text{ для ИЛИ- у зла},\\ \prod_{i=1}^{n}(s_i^z)\text{для И– узла},\\ 1\text{ для листа},\end{cases} \quad (1)$$

где z – рассматриваемый узел дерева; $\{s_i^z\}$ – множество сыновей узла z; n – количество сыновей; $\omega(z)$ – количество вариантов для узла z.

Подсчитав значение функции для корня дерева, можно получить общее число вариантов решений, имеющихся в данном дереве. При этом будет подсчитано количество вариантов для каждого узла всего дерева. Пример подсчета вариантов показан на рис. 1.

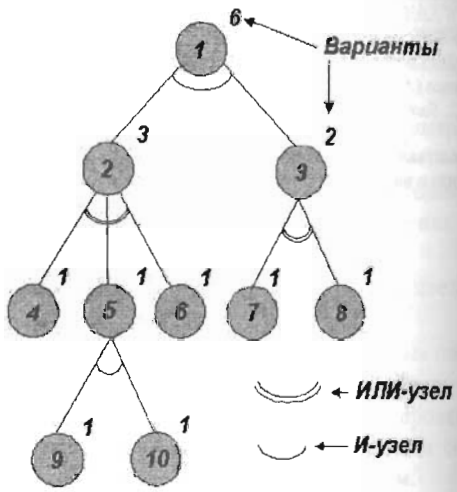

Рис.1. Пример подсчета вариантов в дереве И/ИЛИ

Используя значения функции $\omega(z)$ для каждого узла z, можно построить алгоритм нумерации вариантов. Этот алгоритм по номеру из данного дерева получает необходимый вариант поддерева. Предварительно зададим две функции нумерации, которые по номеру варианта для рассматриваемого узла вычисляют номера вариантов для сыновей. Для И-узла будет следующая функция:

$$I_A(s_i^z)=\begin{cases}\dfrac{I_A(z)}{\prod_{j=1}^{i-1}\omega(s_j^z)}\bmod\omega(s_i^z), & i>1,\\ I_A(z)\bmod\omega(s_i^z), & i=1.\end{cases} \quad (2)$$

Для ИЛИ-узла необходимо определить не только номер варианта, но и номер соответствующего сына. Для этого запишем следующее уравнение относительно k:

$$I_0(s_k^z)=\begin{cases}I_0(z)\text{ при }I_0(z)<\omega(s_k^z),\ k=1,\\ \min\left[I_0(z)-\sum_{j=1}^{k}\omega(s_j^z)\right]\text{ при }I_0(s_k^z)\ge 0,\ k>1.\end{cases} \quad (3)$$

Пусть дано И/ИЛИ дерево D, где $\{s_i\}_{i=1}^{n}$ – множество узлов, и некоторое число L, $0 \le L < \omega(s_{root})$, где s_{root} – корень дерева.

Тогда **алгоритм 2** построения варианта поддерева по заданному номеру L будет следующий.

1) Первоначально производится подсчет количества вариантов для каждого узла дерева $\omega(z)$, $z \in \{s\}$.

2) Корень дерева записывается в вариант и заносится в список $list.add(s_{root}, L)$.

3) Из списка вынимается пара $<z, L_z>=list.pull()$. Если список пуст, то завершить работу.

4) Определяется тип текущего узла. Если это И-узел, то переход на шаг 5, иначе переход на шаг 6.

5) Все сыновья $\{s_j^z\}_{j=1}^{m}$ рассматриваемого узла z записываются в данный вариант, добавляются в список и для каждого узла вычисляется $L(s_i^z)$, используя выражение (2).

6) Если это ИЛИ-узел, то определяется единственный сын s_k^z, заносится в список и определяется $L(s_k^z)$, используя выражение (3).

7) Переход на шаг 3.

На рис. 2 показаны все варианты для дерева И/ИЛИ, приведенного на рис.1, получаемые описанным алгоритмом генерации варианта поддерева по заданному номеру.

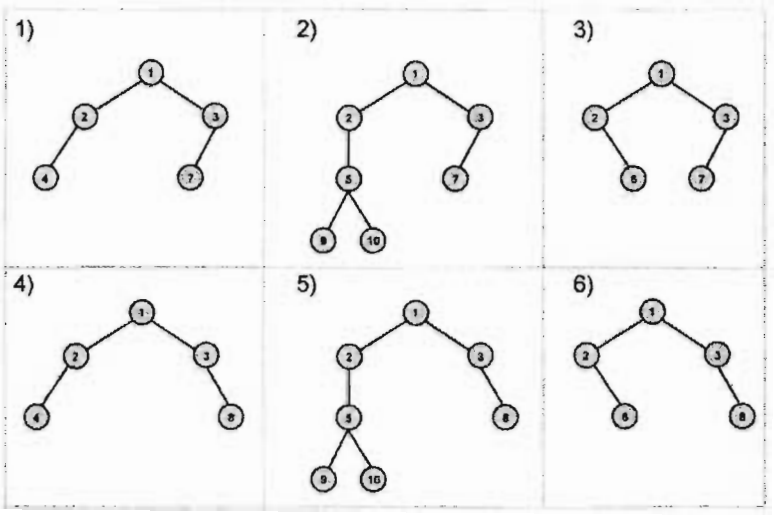

Рис. 2. Все варианты для примера И/ИЛИ дерева

ПЕРЕЧИСЛИТЕЛЬНЫЕ СВОЙСТВА ДЕРЕВЬЕВ И/ИЛИ

Рассмотрим теперь перечислительные свойства деревьев И/ИЛИ.

Свойство 1. Пусть дано два дерева И/ИЛИ d_1 и d_2. Известно, что деревья описывают два разных класса одного и того же объекта, тогда можно создать новое дерево И/ИЛИ Z, корнем которого будет ИЛИ-узел, а сыновьями – корни деревьев d_1, d_2 и $\omega(Z) = \omega(d_1) + \omega(d_2)$.

Свойство 2. Пусть дано два дерева И/ИЛИ d_1 и d_2. Известно, что данные деревья описывают два разных объекта, из которых строится данный объект, тогда можно создать новое дерево И/ИЛИ Z, корнем которого будет И-узел, а сыновьями – корни деревьев d_1, d_2 и $\omega(Z) = \omega(d_1) + \omega(d_2)$.

Свойство 3. Суть алгоритмов нумерации не изменится, если в выражении (1) вместо единицы для листа, записать константу, зависящую от конкретного листа дерева. Тогда данное выражение будет следующее:

$$\omega(z) = \begin{cases} \sum_{i=1}^{n} \omega(s_i^z) \text{ для ИЛИ - узла,} \\ \prod_{i=1}^{n} \omega(s_i^z) \text{ для И - узла,} \\ C_k \text{ для } k\text{ - го листа,} \end{cases} \quad (4)$$

где C_k – константа k-го листа дерева.

Эта константа может быть записана ранее или получена некоторым алгоритмом, который связан с k-м листом дерева.

Можно перечислить следующие классы вариантов: минимальные; максимальные варианты; варианты с нижней границей числа листьев; варианты с верхней границей числа листьев; варианты с весами на листьях; с максимальной глубиной; варианты с минимальной глубиной.

Вариант дерева И/ИЛИ будет максимальным, если у него максимальное число листьев. Подсчет числа листьев у максимального варианта можно выполнить по следующей рекурсивной функции:

$$\mu(z) = \begin{cases} \max \mu(s_i^z) \text{ для ИЛИ - узла,} \\ \sum_{i=1}^{n} \mu(s_i^z) \text{ для И - узла,} \\ 1 \text{ для листа.} \end{cases} \quad (5)$$

Вариант будем называть минимальным, если у него минимальное число листьев.

Подсчет числа листьев у минимального варианта можно выполнить по следующей рекурсивной функции:

$$\nu(z) = \begin{cases} \min \nu(s_i^z) \text{ для ИЛИ - узла,} \\ \sum_{i=1}^{n} \nu(s_i^z) \text{ для И - узла,} \\ 1 \text{ для листа} \end{cases} \quad (6)$$

Алгоритм 3 нахождения варианта с максимальным числом листьев.

Дано: дерево И/ИЛИ D и *root*– корень этого дерева.

Необходимо: найти вариант V с максимальным числом листьев.

1. Находим значение функции $\mu(root)$ (выражение (5).

2. Записываем корень дерева в стек $Stack \xleftarrow{push} root$ и создаем корень дерева варианта V.

3. Если стек пуст, то завершаем работу алгоритма. Иначе вытаскиваем узел из стека и делаем его текущим $Stack \xrightarrow{pull} z$.

4. Если z И-узел, то переходим на шаг 5. Если z – ИЛИ-узел, переходим на шаг 6. Если это лист, то переходим на шаг 3.

5. Все сыновья узла z записываются в стек $Stack \xleftarrow{push} s_i$ и в вариант V.

6. Определяем, какой из сыновей S_i узла z войдет в вариант, для этого используем значения $\mu(z)$ и $\mu(s_i)$

for $(i=1, n)$

if $\mu(z)=\mu(s_i)$ *then* $k=i$ *break* *endif*

Заносим найденный узел в вариант и в стек $Stack \xleftarrow{push} s_k$.

Очевидно, что описанный выше алгоритм можно модифицировать для нахождения варианта с минимальным числом листьев. Для этого необходимо заменить функцию $\mu(s)$ при описании шагов 1 и 6 на функцию $\nu(s)$.

Приведенный выше алгоритм находит один вариант. Однако таких вариантов может быть много, все зависит от значений функции $\mu(s)$ (или $\nu(s)$) при определении сына ИЛИ-узла. Максимальных вариантов будет два, если при выполнении шага 6 будет выполнено соотношение $\mu(z)=\mu(s_j)=\mu(s_k)$.

Общее число вариантов с максимальным количеством листьев будет равно сумме всех совпадений значений $\mu(z)=\mu(s_j)$.

Можно предложить следующий **алгоритм 4** вычисления количества максимальных вариантов:

1. Находим значение функции $\mu(root)$.

2. Записываем корень дерева в стек $Stack \xleftarrow{push} root$ $var=1$.

3. Если стек пуст, то завершаем работу алгоритма. Иначе вытаскиваем узел из стека и делаем его текущим $Stack \xrightarrow{pull} z$.

4. Если z – И-узел, то переходим на шаг 5. Если z ИЛИ-узел, переходим на шаг 6. Если это лист, то переходим на шаг 3.

5. Все сыновья узла z записываются в стек $Stack \xleftarrow{push} \{s_k\}_{k=1}^{n}$.

6. *for* $(i=1, n)$

if $\mu(z)=\mu(s_j)$ *then* $k=k+1$ $Stack \xleftarrow{push} s_i$ *endif*

endfor

$var = var + k - 1$

Для нахождения всех минимальных вариантов необходимо модифицировать алгоритм 4. Для это на шаге 6 подсчет максимальной глубины варианта в дереве И/ИЛИ можно вычислить по следующей формуле:

$$\eta(z)=\begin{cases}\max \eta(s_i)+1 \text{ для узла,}\\ 1 \text{ для листа.}\end{cases} \qquad (7)$$

Как видно из выражения (7), максимальная глубина варианта совпадает с глубиной дерева И/ИЛИ.

Подсчет минимальной глубины варианта в дереве И/ИЛИ можно произвести по следующей формуле:

$$\eta(z)=\begin{cases}\min\limits_i \lambda(s_i) \text{ для ИЛИ - узла,}\\ \max\limits_i \lambda(s_i) \text{ для И - узла,}\\ 1 \text{ для листа.}\end{cases}$$

ИСПОЛЬЗОВАНИЕ ДЕРЕВЬЕВ И-ИЛИ ДЛЯ ПЕРЕЧИСЛЕНИЯ ИНФОРМАЦИОННЫХ ОБЪЕКТОВ

Пусть имеется некоторая информационная модель, записанная в виде некоторой грамматики, графа и т.д., которая описывает всю совокупность информационных объектов (возможно, бесконечную). Запишем основные этапы метода построения генерирующего алгоритма:

1. Проводятся исследования с целью прямого преобразования информационной модели в фиксированное дерево И/ИЛИ. На данном этапе используются свойства 1, 2. Дерево можно строить от корня к листьям, используя методы декомпозиции, или начиная с листьев и производя агрегацию. Листом может стать любой информационный элемент, который будет представлен в единичном виде, или элемент, для которого известен алгоритм пересчета (используется свойство 3).

2. Если прямого преобразования не найдено или оно не эффективно, то проводятся исследования с целью получения алгоритма построения дерева И/ИЛИ.

3. Проводятся исследования свойств полученного дерева И/ИЛИ или алгоритма построения.

4. Производятся исследования, связанные с удалением бесконечностей (рекурсии) и ограничением глубины рекурсии.

Данный метод был использован для разработки алгоритмов генерации тестовых заданий в компьютерных учебных программах [8].

ЗАКЛЮЧЕНИЕ

Деревья И/ИЛИ явились удобным инструментом перечисления разнообразных информационных объектов, имеющих достаточно сложную структуру. Предложенные алгоритмы для заданного дерева определяют общее число вариантов, по указанному номеру строят соответствующий вариант, определяют параметры построенного варианта.

На основании рассмотренных перечислительных свойств деревьев предложен метод построения генерирующих алгоритмов:

1) строится и исследуется И/ИЛИ дерево;

2) используется алгоритм 2 для построения варианта по заданному номеру;

3) производится преобразование варианта в конкретное описание информационного объекта.

Предложенные модели и алгоритмы применялись для генерации вопросов и тестовых заданий в компьютерном тестировании [8].

ЛИТЕРАТУРА

1. *Кручинин В.В.* Разработка компьютерных учебных программ. Томск: Изд-во Том. ун-та, 1998. 211 с.

2. *Akl S.G.* A comparison of combination generation methods // ACM Trans. of Math. Software, 7(1981). P. 42–45.
3. *Akl S.G.* Adaptive and optimal parallel algorithms for enumerating permutations and combinations // The Computer Journal. 1987. Vol. 30. P. 433–436.
4. *Slagle J.R.* A heuristic program that solves symbolic integration problems in freshmen calculus // Eds. E. Feigenbaum and J. Feldman computer and thought. N-Y.: McGraw-Hill, 1963. P. 192–203,
5. *Nilsson N.* Principles of artifical intelligence. Spring Verlag, 1983.
6. *Ефимов Е.И.* Решатели интеллектуальных задач. М.: Наука, 1982. 320 с.
7. *Братко И.* Программирование на языке Пролог для искусственного интеллекта. М.: Мир, 1990. 560 с.
8. *Кручинин В.В.* Генераторы в компьютерных учебных программах. Томск: Изд-во Том. ун-та, 2003. 200 с.

Статья представлена кафедрой промышленной электроники факультета электронной техники Томского государственного университета систем управления и радиоэлектроники, поступила в научную редакцию «Кибернетика и информатика» 15 мая 2004 г.