Теория алгоритмов

Конечные автоматы

Задача «Выполнимость»

Пусть $d_1, d_2, \dots, d_n$– некоторые дизъюнкты.

Тогда конъюнкция $d_1 \wedge d_2 \wedge \dots \wedge d_n$ (1) называется булевым выражением в КНФ.

<u>Постановка задачи:</u>

Пусть задано некоторое выражение в КНФ. Набор значений б.ф. называют выполняющим, если на этом наборе б.ф. =1. Требуется найти выполняющий набор для выражения (1).

(SAT-конференции SAT2000, SAT2002 …).

Имеются булевы выражения, для которых не существует выполняющего набора. Например, б.в.:

$(x \vee y \vee z) \wedge (x \vee \neg y) \wedge (y \vee \neg z) \wedge (z \vee \neg x) \wedge (\neg x \vee \neg y \vee \neg z)$ не является истинным ни для каких значений переменных.

Понятие алгоритма

Теория алгоритмов — раздел математики, изучающий общие свойства алгоритмов.

Тезис Чёрча: понятие рекурсивной функции является уточнением интуитивного понятия алгоритма.

- В 1936 году А. Чёрч опубликовал первое уточнение понятия вычислимой функции и привёл первый пример функции, не являющейся вычислимой.

Понятие алгоритма

Алгоритм — это процесс последовательного построения величин таким образом, что в начальный момент задаётся исходная конечная система величин, а в каждый следующий момент система величин получается по определенному закону из системы величин, имевшихся в предыдущий момент. Последовательный процесс построения величин должен быть конечным и давать результат, то есть решение задачи.

Алгоритм – это точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату.

Пусть D – область (множество) исходных данных задачи, а R – множество возможных результатов, тогда мы можем говорить, что алгоритм осуществляет отображение $D \ -> \ R$.

Понятие алгоритма

Проблема построения алгоритма, обладающего теми или иными свойствами, называется ***алгоритмической проблемой***.

Важный пример алгоритмической проблемы — проблема вычисления данной функции (требуется построить алгоритм, вычисляющий эту функцию). Функция называется ***вычислимой***, если существует вычисляющий ее алгоритм.

Основными математическими моделями понятия алгоритма являются машины Тьюринга, частично рекурсивные функции и др.

Конечный автомат

Конечный автомат - это модель вычислений, основанная на гипотетической машине состояний. В один момент времени только одно состояние может быть активным. Следовательно, для выполнения каких-либо действий машина должна менять свое состояние

Применение

- Для организации и представления потока выполнения чего-либо.
- При реализации интеллектуальных игр.

Представление

Конечный автомат можно представить в виде графа, вершины которого являются состояниями, а ребра — переходы между ними. Каждое ребро имеет метку, информирующую о том, когда должен произойти переход.

Реализация простого конечного автомата

Реализация конечного автомата начинается с выявления его состояний и переходов между ними.

Конечный автомат можно реализовать при помощи одного ***класса***. Идея состоит в том, чтобы реализовать каждое состояние как ***метод*** или ***функцию***.

Основная черта конечных автоматов — они описываются набором возможных состояний, набором сигналов (событий) и таблицей переходов. ***Таблица переходов*** — это сопоставление паре из текущего состояния и пришедшего сигнала нового состояния.

Абстрактный автомат АА

Кортеж $\mathrm{A} = (X, Y, S, fy, fs)$, где первые три компоненты – непустые множества:

X – множество входных сигналов AA,

Y – множество выходных сигналов AA,

S – множество состояний AA.

Две последние компоненты кортежа – характеристические функции:

fy – функция выходов;

fs – функция переходов AA из одного состояния в другое.

Если множества X, Y, S – конечные, то такой AA называют конечным автоматом (KA).

Классификация КА

I. ***По определенности характеристических функций***

В автоматах полностью определенных областью определения функций fs и fy является множество всех пар $(s_i, x_k) \in S \times X$, где $s_i \in S$, $x_k \in X$. В автоматах частично определенных либо обе характеристические функции, либо одна из них имеют областью определения строгое подмножество декартова произведения $S \times X$. Таким образом, характеристические функции подобных автоматов определены не для всех пар (s_i, x_k).

Классификация КА

- **II.** ***По однозначности функции переходов*.**
- В детерминированных автоматах выполняется условие однозначности переходов: если АА находится в некотором состоянии $s_i \in S$, то под воздействием произвольного входного сигнала $x_k \in X$ автомат может перейти в одно и только одно состояние $s_j \in S$, причем ситуация $s_i = s_j$ вовсе не исключается.
- В автоматах вероятностных при воздействии одного и того же входного сигнала возможны переходы из состояния s_i в различные состояния из множества S с заданной вероятностью.

Классификация КА

- **III. *По устойчивости состояний*:**
- В устойчивых автоматах выполняется условие устойчивости: если автомат под воздействием входного сигнала $x_k \in X$ оказался в состоянии $s_i \in S$, то выход из него и переход в иное состояние возможен только при поступлении на вход автомата другого сигнала $x_z \in X$,

$x_z \neq x_k$. Если условие устойчивости не выполняется хотя бы для одного состояния $s_j \in S$, то такой автомат называют неустойчивым.

Структура КА

Операционный автомат выполняет ряд действий над входными данными и выдает результат,

Управляющий автомат задает последовательность этих действий, то есть алгоритм функционирования операционного автомата.

Например, в случае кодового замка операционным автоматом является электромагнит, управляющий засовом, а управляющим автоматом – электронная схема, обеспечивающая считывание и анализ сигналов от клавиш, проверку кода, выдачу сигнала операционному автомату на открытие замка, сброс в начальное состояние.

Структура КА

Другой пример – устройство умножения двоичных чисел с фиксированной запятой.

Операционный автомат представляет собой ряд взаимосвязанных функциональных элементов – сумматора (например, дополнительного кода), регистров входных данных и результата, сдвигового регистра, цепи переноса двоичной единицы.

Управляющий автомат задает порядок, в котором должны действовать составные узлы операционного автомата, чтобы обеспечить последовательность шагов реализуемого алгоритма умножения.

Автоматное программирование

Автоматное программирование (АП) – программирование с явным выделением состояний – это метод разработки ПО, основанный на модели конечных автоматов.

Состояние

Базовым понятием АП является состояние, введенное А. Тьюрингом. Основное свойство состояния системы в момент времени t заключается в отделении прошлого от будущего в том смысле, что текущее состояние несет в себе всю информацию о прошлом системы, необходимую для определения ее реакции на любое входное воздействие, формируемое в момент времени t.

Состояние – это особая характеристика, которая объединяет все входные воздействия прошлого, влияющие на реакцию сущности в настоящий момент времени. Реакция зависит теперь только от входного воздействия и текущего состояния.

UML

Работу КА можно представлять в виде диаграммы состояний, или *графа переходов*. Вершины графа соответствуют состояниям автомата, а дуги – переходам между состояниями.

Автоматные модели *применяются* в математической лингвистике, логическом управлении, генетическом программировании, теории формальных языков, параллельных вычислениях и т.д.

Задачи логического управления

В системах управления логика может быть реализована как *программно*, так и *аппаратно*. Критерии оптимальности программной реализации автоматов в системах логического управления: возможность формального преобразования графа переходов в программный код; изоморфизм программного кода графу переходов КА; эффективность по времени и по памяти.

Представление в С

Схема алгоритма, реализующего КА, представима в виде обычной блок-схемы. В языке С функции выходов и переходов автомата представляются в виде таблиц либо с помощью инструкций выбора. Состояния описываются в ООП через классы.

Автоматы и алгоритмы дискретной математики

КА используются при построении алгоритма поиска подстрок.

Автоматные алгоритмы часто являются более структурированными, а их представление с помощью диаграмм переходов – более наглядным. Эти свойства приобретают особенно большое значение при обучении дискретной математике.

Обход двоичных деревьев

Три способа – *нисходящий, восходящий* и *смешанный.*

- Классические решения этой задачи – рекурсивное и методом итераций.
- Рекурсивные обладают низким быстродействием, а итерационные – более сложные.
- В автоматной реализации алгоритма дерево представлено в виде **struct**, и **класс**, включающий **функции** размещения вершины в стек, и удаления из стека.

Идея алгоритма в том, что при обходе двоичного дерева могут быть выделены лишь три направления движения: *влево, вправо и вверх*. Поэтому удобно сопоставить каждому направлению движения управляющее состояние автомата.

Построение визуализаторов

КА применимы для построения визуализаторов алгоритмов ДМ.

Визуализатор – это программа, в процессе работы которой на мониторе динамически демонстрируется применение алгоритма к выбранному набору данных.

Визуализаторы позволяют изучать работу алгоритмов как в автоматическом так и в пошаговом режимах.

Машина Тьюринга

Машина Тьюринга — абстрактное устройство, состоящее из бесконечной в обе стороны ленты, считывающей и печатающей головки, способной перемещаться вправо и влево, и управляющего устройства. Лента разбита на ячейки (клетки). Считывающая и печатающая головка перемещается вдоль ленты так, что в каждый момент времени она обозревает ровно одну ячейку ленты. В ячейках могут быть записаны символы некоторого конечного алфавита (*внешний алфавит*)

Машина Тьюринга

$$A = \{a_0, a_1, \dots, a_k\}$$
$$Q = \{q_0, q_1, \dots, q_n\}$$

$$qa \to q'a'D$$

Где $a, a' \in A$; $q, q' \in Q$; $D \in \{R, L, S\}$; R, L, S – вправо, влево, стоп.

Машина Тьюринга

Команда расшифровывается так:

если машина находится в состоянии q и считанный с ленты символ равен a, то машина переходит в состояние q', печатает в текущей клетке символ a' и затем выполняет одно из трех действий D.

Если $D = R$, то машина смещается на одну клетку вправо,

если $D = L$, то на одну клетку влево,

а если $D = S$, то машина никуда не смещается.

Машина Тьюринга

Необходимо, чтобы в программе не было разных команд с одинаковыми входами вида

$qa \ \rightarrow \ q'a'D'$ и $qa \ \rightarrow \ q''a''D''$

— это противоречит однозначности алгоритма.

Изначально машина находится в состоянии q_1 . Если машина пришла в состояние q_0, то она останавливается.

ПРИМЕР 1: Найти результат применения машины Тьюринга, заданной программой

$$q_1 0 \to q_1 0 R,$$
$$q_1 1 \to q_2 0 R,$$
$$q_2 0 \to q_0 1 S,$$
$$q_2 1 \to q_1 0 R,$$

к записям на ленте

$P_1 = 0\ 1\ 1\ 1\ 0\ 1\ 0$ и $P_2 = 0\ 1\ 1\ 1\ 1\ 0$.

ПРИМЕР 1: $P_1 = 0111010$ и $P_2 = 011110$

Решение. Имеем

$$P_1\colon \underset{q_1}{0}111010 \to 0011\underset{q_2}{0}10 \to 000101\underset{q_1}{0} \to$$

$$\to 00000\underset{q_2}{1}0 \to 000011\underset{q_0}{0},$$

$$P_2\colon \underset{q_1}{0}11110 \to 0011\underset{q_2}{1}0 \to 00011\underset{q_1}{0} \to$$

$$\to \underset{q_2}{0}00010 \to 000\underset{q_1}{0}00 \to 0000\underset{q_1}{0}00 \to \cdots$$

Машину Тьюринга удобно применять при вычислении функций вида

$$f: \mathbb{Z}_+^k \to \mathbb{Z}_+^m,$$

$$\mathbb{Z}_+ = \{0,1,2,\dots\}$$

$$n_1, n_2, \dots, n_k$$

$$0 \to 010;$$
$$1 \to 0110;$$
$$2 \to 01110;$$
$$\vdots$$

$$\dots 0\,\underbrace{11\dots1}_{n_1+1}\,0\,\underbrace{11\dots1}_{n_2+1}\,0 \dots 0\,\underbrace{11\dots1}_{n_k+1}\,0\ \dots.$$

Пример 2

Построим машину Тьюринга, которая к числу на ленте будет прибавлять 1. Она дойдет до конца массива из единиц, поставит туда 1 и вернется назад, т. е.

$$0\ \underbrace{11\ldots1}_{q_1\ \ n+1}\ 0\ \to 0\ \underbrace{11\ldots1}_{q_0\ \ n+2}\ 0.$$

$$q_1 1 \to q_1 1R,$$
$$q_1 0 \to q_2 1L,$$
$$q_2 1 \to q_2 1L,$$
$$q_2 0 \to q_0 0S.$$

Пример 3

$$011100011110 \to 011111111110.$$

q_1 q_0

$q_1 1 \to q_1 1R,$ $q_1 0 \to q_2 1R,$	1-й шаг
$q_2 0 \to q_2 1R,$ $q_2 1 \to q_3 1R,$	2-й шаг
$q_3 1 \to q_3 1R,$ $q_3 0 \to q_0 0S.$	3-й шаг

Говорят, что машина Тьюринга *вычисляет* функцию $\boldsymbol{f(x_1, \dots, x_k)}$: $\mathbb{Z}_+^{\boldsymbol{k}} \to \mathbb{Z}_+^{\boldsymbol{m}}$, если на любом наборе $(\boldsymbol{a_1}, \dots, \boldsymbol{a_k}) \in \boldsymbol{D}(\boldsymbol{f})$ машина останавливается и на ленте остается результат $(\boldsymbol{b_1}, \dots, \boldsymbol{b_m}) = \boldsymbol{f}(\boldsymbol{a_1}, \dots, \boldsymbol{a_k})$, а в случае $(\boldsymbol{a_1}, \dots, \boldsymbol{a_k}) \notin \boldsymbol{D}(\boldsymbol{f})$ она работает вечно, т. е. неприменима к таким входным данным.

Универсальная кодировка машины Тьюринга

R	L	S	a_0	a_1	$\cdots$	a_k	q_0	q_1	$\cdots$	q_n
1	3	5	7	9	$\cdots$	$2k+1$	0	2	$\cdots$	$2n$

$$\{1,*\}$$

$$K(M) = K_1 * K_2 * \ldots * K_p,$$

где K_i — коды всех команд программы.

ПРИМЕР 4: Построить код машины Тьюринга с программой

$$q_1 1 \to q_1 1R,$$
$$q_1 0 \to q_2 1L,$$
$$q_2 1 \to q_2 1L,$$
$$q_2 0 \to q_0 0S,$$

Решение. Закодируем набором из **5** чисел каждую команду, используя таблицу кодов:

$2 * 9 * 2 * 9 * 1,$ $\quad 2 * 7 * 4 * 9 * 3,$ $\quad 4 * 9 * 4 * 9 * 3,$ $\quad 4 * 7 * 0 * 7 * 5.$

ПРИМЕР 4 (окончание)

Теперь представим коды команд с помощью алфавита $\{1,*\}$:

$$1^3 * 1^{10} * 1^3 * 1^{10} * 1^2 * 1^3 * 1^8 * 1^5 * 1^{10} * 1^4 * 1^5 * 1^{10} * 1^5 * 1^{10} * 1^4 * 1^5 * 1^8 * 1 * 1^8 * 1^2,$$

где 1^k есть единица, повторенная k раз.

Алгоритмически неразрешимые проблемы

- Символы 1 и $*$.
- Если при работе над собственным кодом машина Тьюринга ***M*** останавливается, то она называется *самоприменимой*.
- Существует ли машина ***Ms*** , которая по коду любой машины ***M*** определяет, самоприменима ли она?

Теорема. Ms не существует, то есть проблема самоприменимости алгоритмически неразрешима.

Доказательство: Пусть машина Тьюринга $\boldsymbol{S}$ решает проблему самоприменимости, т.е., начав работу с кода машины $\boldsymbol{T}$, приходит в состояние $\ldots \boldsymbol{q_0 1} \ldots (*)$, если машина $\boldsymbol{T}$ самоприменима, и в состояние $\ldots \boldsymbol{q_0 0} \ldots (**)$, если $\boldsymbol{T}$ несамоприменима.

Продолжение доказательства:

Рассмотрим машину ***R***, программа которой состоит из всех команд машины ***S*** и еще двух команд $\boldsymbol{q_0 1 \to q_0 1}$ и $\boldsymbol{q_0 0 \to q_0' 0}$.

Продолжение доказательства:

Здесь $\boldsymbol{q_0}$— не заключительное, а $\boldsymbol{q_0'}$— заключительное состояние. Если машина $\boldsymbol{R}$ самоприменима, то, начав работу со своего кода, она, в силу команд машины $\boldsymbol{S}$ придет в состояние **(*)**. Затем в силу команды $\boldsymbol{q_0 1} \to \boldsymbol{q_0 1}$ она будет работать бесконечно. Это значит, что $\boldsymbol{R}$ несамоприменима. Противоречие.

Окончание доказательства:

Точно так же, если ***R*** несамоприменима, она придет сначала в состояние **(**)**, а затем остановится в силу команды $\boldsymbol{q_0 0 \to q_0' 0}$. Значит, ***R*** самоприменима. Полученное противоречие и доказывает теорему.

Тезис Тьюринга

Всякий алгоритм представим в форме машины Тьюринга.