

Оглавление

Глава 1. Введение и постановка задачи	2
1.1 Введение	2
1.2 Постановка задачи	2
Глава 2. Описание программы	4
2.1 Описание сущностей	4
2.1.1 Tuple	4
2.1.2 Function	5
2.1.3 Permutation	5
2.1.4 Predicate	5
2.2 Описание алгоритмов нахождения и перебора предикатов	5
2.2.1 Семейство P	5
2.2.2 Семейство O	6
2.2.3 Семейство E	6
2.2.4 Семейство L	7
2.2.5 Семейство C	7
2.2.6 Семейство B	8
2.3 Получение результатов	8
Глава 3. Результаты и заключение	9
Список литературы	10

Глава 1

Введение и постановка задачи

1.1 Введение

Исследование систем k -значных функций было инициировано Э.Постом [1, 3] для случая $k = 2$.

Все предполные классы в P_3 и P_4 были описаны С.В. Яблонским[2]

Как известно, каждый предполный класс в P_k можно задать как множество функций, сохраняющих некоторый предикат. Более того, Розенберг показал [5, 6], что все эти предикаты можно разделить на 6 попарно непересекающихся семейств: P, O, L, E, C, B , описание которых можно найти в книге Марченкова С.С. [4].

1.2 Постановка задачи

Пусть Q, R — замкнутые классы и $Q \subset R$ (включение строгое). Говорят, что класс Q предполон в классе R , если $[Q \cup \{f\}] = R$ для любой функции f из множества $R \setminus Q$.

- Выяснить какие из предикатов, задающих предполные классы, имеются в P_k , при $k = 4$.
- Разработать программный продукт, позволяющий работать с предикатами и функциями, в частности, для заданного конечного множества

функций в P_k предоставлять возможность проверки факта сохранения функцией предиката.

- Для каждой функции одной переменной из P_4 определить все предположенные классы, которым она принадлежит.

Глава 2

Описание программы

Для решения задачи была написана программа на языке программирования Java, исходный код которой можно найти в [7]. Там же можно найти результаты и вспомогательные материалы, в частности и этот текст.

Для создания проекта использовалась методология DDD (Domain-driven design).

Для решения поставленной задачи необходимо перебирать однотипные объекты. Для этой цели хорошо подходят интерфейсы `Iterator` и `Iterable`. Кроме того они позволяют облегчить понимание исходного кода. При написании программы часто использовались следующие обозначения:

- *Dim* – “значность”
- *Capacity* – «местность».

Опишем основные сущности, реализованные в Java классах.

2.1 Описание сущностей

Все определения сущностей находятся в пакете `predicates.domain`

2.1.1 Tuple

Класс, который описывает упорядоченный набор чисел из `capacity` чисел от 0 до $dim - 1$. реализует интерфейсы `Iterator` и `Iterable`, что позволяет перебирать кортежи по порядку (лексикографическому).

2.1.2 Function

Класс, который описывает функцию, зависящую от *capacity* переменных чисел, из P_{dim} , реализует интерфейсы `Iterator` и `Iterable`, что позволяет перебирать функции по порядку столбцов значений (лексикографическому).

2.1.3 Permutation

Класс, который описывает перестановку чисел от 0 до *capacity* – 1, реализует интерфейсы `Iterator` и `Iterable`, что позволяет перебирать перестановки по порядку (лексикографическому).

2.1.4 Predicate

Описывает предикат, используя `Set<ImmutableList<Integer> >` - множество векторов, удовлетворяющих предикату.

2.2 Описание алгоритмов нахождения и перебора предикатов

Для каждого семейства предикатов был написан Java-класс `PredicateFactory_X`, где X – название соответствующего семейства. Каждый из них реализует интерфейсы `Iterable<Predicate>`, `Iterator<Predicate>`.

Все они лежат в пакете `predicates.factory`.

Т.к. при нахождении необходимых предикатов во многих случаях требуется перебор нескольких параметров и при некоторых комбинациях могут получаться одинаковые результаты, необходимо было не выдавать уже полученные идентичные предикаты. Эта проблема чаще всего решалась использованием структур данных, реализующих интерфейс `Set<Predicate>`.

Перейдем к подробному описанию каждого из модулей программы, соответствующих семействам предикатов.

2.2.1 Семейство P

Определение 1. Семейство P . Предикаты этого семейства существуют при любом $k \leq 2$. Пусть π — перестановка на E_k , которая разлагается

в произведение циклов одной и той же простой длины. Для любой такой перестановки π в семейство P входит предикат $\pi(x_1) = x_2$, который называется графиком перестановки π .

Каждый из предикатов класса задается перестановкой, которая является произведением циклов одной и той же простой длины (при $k = 4$ получается 2 цикла длины 2), поэтому для нахождения всех предикатов программа перебирает все перестановки, проверяет на выполнение вышеописанного условия (в классе `Permutation` есть соответствующий метод) и, при успешном результате проверки, строит предикат.

2.2.2 Семейство O

Определение 2. Семейство O содержит любой двуместный предикат, который задает на E_k частичный порядок с наименьшим и наибольшим элементами (ограниченный частичный порядок).

Для нахождения всех таких предикатов программа перебирает все перестановки чисел от 0 до $dim - 1$ (от 0 до 3 при $k = 4$), считая, что первый элемент перестановки будет наибольшим, последний – наименьшим.

Далее берутся все пары элементов (a_i, a_j) , где $1 < i < j < k$ и перебираются все из $2^{(k-2)*(k-3)/2}$ вариантов считать или не считать, что эта пара принадлежит предикату, причем отсекаются те конфигурации, не обладающие свойством транзитивности. Для каждого подошедшего варианта строится предикат.

2.2.3 Семейство E

Определение 3. Семейство состоит из всех двуместных предикатов, которые представляют собой отношения эквивалентности на E_k , отличные от полного и единичного отношений (нетривиальные отношения эквивалентности). Таким образом, каждое отношение эквивалентности из E разбивает множество E_k на l классов попарно эквивалентных элементов, где $1 < l < k$.

Нахождение всех таких предикатов осуществляется при помощи перебора всех возможных разбиений и построения соответствующих предикатов.

2.2.4 Семейство L

Определение 4. Предикаты этого семейства существуют только при $k = p^l$, где p — простое число и $l > 0$. В этом случае на множестве E_k можно определить бинарную коммутативную операцию $+$ так, что $G = \langle E_k; + \rangle$ будет являться абелевой p -группой периода p . Иными словами, в абелевой группе G порядок любого элемента, отличного от нуля группы, равен p .

Итак, если $k = p^l$ и $G = \langle E_k; + \rangle$ — абелева p -группа периода p , то семейству L принадлежит предикат $x_1 + x_2 = x_3 + x_4$.

При $k = 4$ и $p = l = 2$ существует только одна таблица сложения, удовлетворяющая описанным условиям:

	x_1	x_2	x_3	x_4
x_1	0	1	2	3
x_2	1	0	3	2
x_3	2	3	0	1
x_4	3	2	1	0

Для нахождения всех таких предикатов программа перебирает все перестановки чисел от 0 до $\dim - 1$ (от 0 до 3 при $k = 4$) и строит предикат $x_1 + x_2 = x_3 + x_4$. При $k = 4$ имеется только предикат в этом семействе.

2.2.5 Семейство C

Определение 5. Семейство C состоит из всех центральных предикатов.

Определение 6. Предикат $p(x_1, \dots, x_m)$ называется центральным, если он вполне рефлексивен, вполне симметричен, отличен от тождественно истинного предиката и существует такое непустое подмножество C множества E_k (центр предиката p), что предикату p удовлетворяет всякий набор (a_1, \dots, a_m) из E_k^m , как только $(a_1, \dots, a_m) \cap C \neq \emptyset$.

Определение 7. Предикат $p(x_1, \dots, x_m)$ называется вполне рефлексивным, если либо $m = 1$, либо если $m > 1$ и $p(a_1, \dots, a_m) = \text{True}$ для любого набора (a_1, \dots, a_m) из E_k^m , содержащего не более $m - 1$ различных значений.

Определение 8. Предикат p называется вполне симметричным, если он не меняется при любой перестановке переменных.

Для перебора всех предикатов нам необходимо перебрать размерность предиката.

Далее, для каждой размерности, мы строим минимальный вполне рефлексивный предикат. На следующем шаге мы перебираем все возможные центры и добавляем вектора, которые имеют с ним непустое пересечение, в предикат.

На последнем этапе перебора мы всеми возможными способами пытаемся его расширить еще не вошедшими векторами, учитывая условия симметричности и нетривиальности.

2.2.6 Семейство B

Определение 9. Для любого $m \geq 2$ положим

$$\tau_m(x_1, \dots, x_m) = \bigvee_{1 \leq i < j \leq m} (x_i = x_j)$$

Определение 10. Если $h \geq 3, l \geq 1, k \geq h^l$ и q — отображение множества E_k на множество E_{h^l} , то семейству B принадлежит предикат, который является полным прообразом l -й декартовой степени предиката τ_h при отображении q .

При $k = 4$ нам подходит $h = 3, 4$ и $l = 1$.

Для каждого из двух вариантов построим множества предикатов следующим образом:

1. Переберем все отображения множества E_k на множество E_{h^l} .
2. Для каждого отображения будем строить предикат `answer`, перебирая все возможные кортежи и проверяя, должны ли они входить в него при условии того, что `answer` должен быть прообразом предиката τ .

2.3 Получение результатов

Теперь, получив все искомые предикаты, необходимо проверить, какие функции сохраняют их. Для такой проверки в проекте есть класс `PredicateService`, который содержит метод с сигнатурой

```
public static boolean checkSave (Predicate predicate, Function function).
```

Теперь достаточно перебрать все пары, состоящие из функций и найденных предикатов, и применить к ним данный метод.

Глава 3

Результаты и заключение

1. Были найдены все предикаты, описывающие предполные классы в P_4 , и построена таблица (см. приложение А) распределения функций одной переменной четырехзначной логики этим предикатам.
2. Была создан Java-проект с архитектурой, позволяющей использовать его для задач, связанных с предикатами и предполными классами при значениях $k \geq 3$ (сейчас только при нахождении предикатов семейств L и B есть ограничения на k). Исходный код программы находится в приложении Б.

Список литературы

1. Post E.L. Introduction to a general theory of elementary propositions // Amer. J. Math.— 1921.- V. 43, №4.- P. 163-185.
2. Е.Ю. Захарова, В.Б. Кудрявцев, С.В. Яблонский О предполных в k -значных логиках. // ДАН СССР, 1969, т.186, №3, стр.509-512
3. Post E.L. Two-valued iterative systems of mathematical logic // Annals of Math. Studies. Princeton Univ. Press.— 1941.— V. 5.
4. Марченков С.С. Функциональные системы с операцией суперпозиции
5. Rosenberg I.G. La structure des fonctions de plusieurs variables sur un ensemble fini // C.R. Acad. Sci. Paris. Ser A.B.— 1965.— V. 260.— P. 3817-3819.
6. Rosenberg I.G. Uber die funktionale Vollständigkeit in den mehrwertigen Logiken // Rozprawy Ceskoslovenske Akad. Ved. Rada Math. Pfir. Ved. Praha.— 1970.— Bd. 80.- S. 3-93.
7. Адрес проекта в интернете: <https://github.com/zloi-timur/predicates>