

Институт ИТКН

Кафедра инженерной кибернетики

Направление подготовки: 09.04.03 Прикладная информатика

Квалификация (степень): магистр

Группа: МПИ-20-4-2


# ОТЧЕТ

## ПО НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

на тему: «Математическое и программное обеспечение для решения казуально-логических игр с использованием технологий самообучения»

IV семестр 2021 – 2022 у. г.

Студент:

 /Новицкий Д. А./

подпись

Фамилия И.О.

Руководитель НИР:

 /доцент, к.т.н. Кожаринов А. С./

подпись

должность, уч. степ. Фамилия И.О.

Оценка:

\_\_\_\_\_

Дата защиты:

10.04.22

Утвердил:

Председатель комиссии

\_\_\_\_\_/\_\_\_\_\_

подпись

Фамилия И.О.

## Оглавление

СПИСОК ИСПОЛЬЗУЕМЫХ ОСНОВНЫХ СОКРАЩЕНИЙ .....	3
ВВЕДЕНИЕ .....	4
Цель работы .....	4
1 АНАЛИТИЧЕСКИЙ ОБЗОР .....	5
1.1 Обзор логических задач .....	5
1.1.1 Виды головоломок.....	6
1.1.2 Выбор логической задачи для самообучающейся системы .....	7
1.2 Алгоритмы построения самообучающихся систем .....	8
1.2.1 Гибридные модели анализа ситуаций .....	8
1.2.2 Разработанный метод построения самообучающейся системы.....	10
1.2.3 Анализ алгоритмов построения самообучающихся систем.....	14
2 СПЕЦИАЛЬНАЯ ЧАСТЬ .....	16
2.2 Содержательная постановка задачи.....	17
2.3 Математическая постановка задачи .....	18
2.4 Описание алгоритма.....	22
2.4.1 Общий алгоритм .....	22
2.4.2 Первое правило вычисления значения в закрытой клетке .....	23
2.4.3 Применение гипотез для одной и нескольких закрытых клеток .....	25
2.4.4 Второе правило вычисления значения в закрытой клетке .....	29
2.4.5 Третье правило вычисления значения в закрытой клетке.....	33
ВЫВОДЫ .....	34
ТЕЗАУРУС .....	35
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	37
ПРИЛОЖЕНИЕ А. Правила игры «Minesweeper»/«Сапёр» .....	38

## СПИСОК ИСПОЛЬЗУЕМЫХ ОСНОВНЫХ СОКРАЩЕНИЙ

- **ИНС** – искусственные нейронные сети;
- **КЛИ** – казуально-логическая игра;
- **CCV** (close cell values) – значения закрытых клеток;
- **OCV** (open cell values) – значения открытых клеток;
- **GS** (game status) – статус игры.

## **ВВЕДЕНИЕ**

В последние десятилетия в научно-техническом обществе большое внимание уделяется развитию такой отрасли, как машинное обучение. Оно применяется во многих сферах деятельности людей, например, при поиске информации в интернете, при подборе музыки по предпочтениям, при доступе к банковским данным с применением биометрии, и это далеко не полный список, где используются методы машинного обучения. На данный момент наиболее популярными методами машинного обучения являются нейронные сети и экспертные системы. Нейронные сети позволяют решать большой объём задач, не связанных с логической обработкой данных, а экспертные системы позволяют решать задачи по строго заданному алгоритму. Однако, ни один из данных методов не позволяет найти решение логических задач без предварительной настройки, поскольку у данных методов отсутствуют алгоритмы самообучения. Разработка алгоритмов самообучения позволит решать широкий круг задач, связанных с логической обработкой данных.

Однако, для того чтобы определить основы для построения алгоритмов самообучения, необходимо пройти долгий путь. Для начала необходимо выбрать круг логических задач, разработать алгоритм их решения, затем разделить алгоритм на блоки, для каждого блока найти наиболее подходящий самообучающийся алгоритм (это может быть ИНС или что-то другое) и затем из самообучающихся блоков построить самообучающуюся систему для решения определённого круга задач. И это лишь примерный алгоритм разработки самообучающейся системы.

В данной работе рассматриваются методы построения системы с элементами самообучения для поиска эффективного решения определённого класса задач, связанных с логической обработкой данных. На основе данных методов разработаны алгоритмы и их программная реализация.

## **Цель работы**

Цель работы – разработка комплекса алгоритмов с элементами самообучения и их программной реализации для независимого от человека поиска эффективного решения казуально-логических игр (на примере игры «Minesweeper»/«Сапёр»).

# 1 АНАЛИТИЧЕСКИЙ ОБЗОР

## 1.1 Обзор логических задач

Всего существует огромное множество логических задач и важно определить круг задач, которые будут решаться разрабатываемой самообучающейся системой. Для того, чтобы выбрать круг логических задач, необходимо определить критерии, по которым будет происходить их выбор. Это такие критерии, как:

- сложность;
- формализуемость;
- наличие однозначного решения;

Поясним необходимость данных критериев. Сложность логической задачи – это основополагающий критерий выбора. Хотя данный критерий можно считать субъективным, поскольку нет единого стандарта оценки сложности той или иной логической задачи, в данном случае под сложностью будем подразумевать, количество входных данных, уровень однотипности логических операций и «объём» логических операций, необходимых для решения задачи. Если задача будет сложной, то и самообучающийся алгоритм также должен быть сложным, а, поскольку в данной работе предполагается разработка самообучающейся системы не трудных логических задач, в приоритете будут задачи с низким и средним уровнями сложности.

От уровня формализуемости зависит сложность представления логической задачи в математических терминах, а, чем выше сложность представления логической задачи в математических терминах, тем выше вероятность запутаться в данных, установить зависимости между ними, не учесть тот или иной элемент задачи. Таким образом, будем также ориентироваться на задачи с низким и средним уровнями формализуемости.

Наличие однозначного решения также важно при выборе логической задачи для самообучающейся системы. Задачи, имеющие однозначные решения, проще решать, поскольку такие задачи могут решаться по шаблону, которому самообучающейся системе необходимо научиться. Задачи, не имеющие однозначного решения, хоть и могут также решаться по шаблону, но такой метод не всегда может быть эффективным, поскольку в некоторых ситуациях необходим творческий подход к решению задачи, которому пока обучить машину не получилось. В данной работе будем отдавать предпочтение логическим задачам, которые имеют однозначное решение.

Головоломки – это распространённые и интересные логические задачи, решение которых зависит не от специальных знаний и творческих умений, а от сообразительности,

а, говоря на языке машины, зависит от алгоритма, определяющие порядок применения логических операций к тем или иным элементам задачи. Рассмотрим, какие бывают виды головоломок.

### **1.1.1 Виды головоломок**

Ещё никто не придумал общей классификации для задач на логику, однако, исходя из их смысла, можно выделить четыре категории [1]:

- головоломки с предметами;
- механические головоломки;
- печатные головоломки;
- устные головоломки;
- компьютерные игры-головоломки.

Рассмотрим подробнее каждую из этих категорий.

#### **1.1.1.1 Головоломки с предметами**

Не обязательно покупать головоломку, порой бывает достаточно предметов, которые имеются в каждом доме. Одной из таких, которая обрела невероятную популярность, является головоломка со спичками. Её суть сводится к тому, что нужно переставить определённое количество спичек, чтобы получилась другая фигура. Ещё одна разновидность: переставить спички так, чтобы вышло верное равенство (речь идёт о цифрах). Наподобие этого есть головоломка с монетками.

#### **1.1.1.2 Механические головоломки**

Всемирно известный пример – кубик Рубика, автор которого также придумал «змейку». Также здесь можно вспомнить любимую игру детей – пятнашки. Все эти головоломки объединяет одно – это предметы, которые созданы для решения поставленных задач. Описанные выше игры найдутся далеко не в каждом доме, но есть и такая головоломка, в которую играли, пожалуй, все – это пазлы.

### **1.1.1.3 Печатные головоломки**

К этому виду относятся головоломки, которые напечатаны на бумаге, а для их решения понадобится ручка. Примерами такой зарядки для ума являются:

- сканворды;
- кроссворды;
- ребусы;
- японские кроссворды;
- sudoku и другие.

### **1.1.1.4 Устные головоломки**

К данному типу игр относятся загадки, игра в «да или нет», а также шарады.

### **1.1.1.5 Компьютерные игры-головоломки**

К этой категории головоломок можно отнести наиболее известные игры, выпущенные как встроенные приложения в операционную систему Windows ещё в 90-е годы:

- сапёр;
- пинбол;
- пасьянс;
- червы;
- косынка;
- солитер;
- маджонг и другие.

## **1.1.2 Выбор логической задачи для самообучающейся системы**

Итак, из большого количества логических игр выберем ту, для которой будем разрабатывать самообучающуюся систему.

Головоломка со спичками имеет достаточно узкий круг возможных задач, притом, данная задача может решаться с помощью полного перебора вариантов решения, поэтому данная задача не подходит.

Кубик Рубика является достаточно сложной задачей, хотя и достаточно популярной с имеющимся алгоритмом решения. Основная проблема при решении данной задачи – возможность визуализации процесса решения, поэтому данная задача также не подходит.

Сканворды, кроссворды и ребусы основаны на поиске информации по заданным данным, а также на игре слов, и при их решении практически не используются логические элементы, а sudoku уже подходит по критериям, поскольку данная логическая задача имеет средний уровень сложности, хорошо формализуема, а также имеет однозначное решение.

Карточные игры, наподобие игр «Пасьянс», «Червы», «Косынка», «Солитер», а также «Маджонг» не всегда могут иметь однозначное решение, «Пинбол» достаточно трудно формализуемая задача, а «Сапёр» подходит по рассматриваемым критериям.

Таким образом, были выбраны две логические игры для построения самообучающейся системы: sudoku и сапёр.

## 1.2 Алгоритмы построения самообучающихся систем

Рассмотрим теперь существующие алгоритмы построения самообучающихся систем.

### 1.2.1 Гибридные модели анализа ситуаций

Наиболее частым подходом к построению современных интеллектуальных систем является использование гибридных моделей анализа ситуаций [2].

Рассмотрим обучаемую модель игры в составе следующих блоков:

1. Создание поля. Создание игрового поля состоит из двух этапов:

- с помощью функции генерации случайных чисел размещается заданное число единиц - «мин» по полю заданного размера в матрице **A**;
- для каждой клетки матрицы **A** вычисляется сумма окружающих ее «мин», значение записывается в соответствующую ячейку матрицы **B** того же размера.

При решении задачи «сапера» пользователь или программа обращаются к матрице **B** во время открытия неизвестных ячеек; с помощью матрицы **A** проверяется количество оставшихся «мин» и условия проигрыша и победы. В матрице **C** того же размера ведется учет вероятностей нахождения «мин» в той или иной клетке. Матрица **D** содержит описание текущей видимой ситуации. В начальный момент все  $d_{i,j} = 10$ , т. е. не опознаны. Для клеток с «миной» будем использовать маркер  $d_{i,j} = 9$ , открытые клетки содержат маркеры от  $d_{i,j} = 0$  («мин» рядом нет) до  $d_{i,j} = 8$ , (вокруг только «мины»).



2. Модуль обработки жесткой логики содержит следующие правила:

- если количество «мин» вокруг данной клетки соответствует ее числу, можно открыть все остальные ячейки вокруг нее, т. е. вероятность нахождения «мины» устанавливается равной 0;
- если количество неизвестных «мин» вокруг данной клетки равно числу свободных клеток вокруг нее, то можно поставить там «мины», т. е. вероятность нахождения «мины» устанавливается равной 1;

3. Модуль принятия решения содержит следующие правила:

- при неоднозначной расстановке «мин» следует выбрать наиболее вероятные точки путем составления матрицы вероятных положений с учетом обучения и расположения открытых клеток;
- при равнозначном выборе в режиме обучения запросить помощь пользователя. Описать решение пользователя в виде модели размера  $K(n)_{5 \times 5}$  извлеченной из матрицы **D**. На основании его решений изменить параметры вероятности соответствующей клетки и проверить качество принятого решения: «удача\ошибка». Если решение было удачным, то вероятность отсутствия «мины» для клеток, соответствующих области типа  $K(n)_{5 \times 5}$  увеличить, иначе следует ее уменьшить. В начальный момент считаем, что  $P(K(n)_{5 \times 5} - \text{"мина"}) = 1$ , т. е. для всех образцов «мины» есть;
- в режиме игры при равнозначном выборе полагаться на выбор. На основании получившегося результата изменить параметры вероятности;

4. Модуль изменения параметров вероятности. Работа с матрицей вероятностей **C** включает в себя следующие правила:

- свободные закрытые клетки имеют собственную вероятность нулевого уровня, являющуюся суммой вероятностей нахождения «мины» около открытых ячеек. Соответствующие слагаемые вычисляются как отношение количества недостающих «мин» к количеству свободных закрытых клеток вокруг открытых ячеек, расположенных вокруг текущей клетки;
- изменение вероятности при поощрении решения вычисляется следующим образом:  $P_{t+1} = P_t + \Delta$ ;
- изменение вероятности при наказании решения вычисляется как:  $P_{t+1} = P_t - \Delta$ .

Таким образом, картина вероятности для каждой клетки, неоткрытой в текущий момент, обуславливается опытом удачных или неудачных решений в схожей ситуации и, с увеличением объема накопленных результатов, изменяется на соответствующую величину.

Результаты применения данного алгоритма представлены в таблице 1 [3].

	Без анализа вероятностей			С анализом вероятностей		
число мин	игр	побед	%	игр	побед	%
$q = 10$	50	38	76,0	49	34	69,4
$q = 15$	50	6	12,0	49	23	46,9
$q=20$	50	0,5	1,0	72	6	8,3

Табл. 1. Результаты применения алгоритма

### 1.2.2 Разработанный метод построения самообучающейся системы

Основная идея данного метода заключается в синтезе новых (производных) правил решения задачи на основе базовых правил.

Синтез новых правил будет происходить при помощи обучения на мини-полях размером 3\*3, 4\*4, 5\*5 и 6\*6 клеток. В данных полях для каждой клетки генерируются значения, которые могут находиться в клетках (это цифры [0; 8], мина (М), стена (С), закрытая клетка (З)). С помощью полного перебора значений в клетках и с помощью применения базовых правил выявляются:

- недопустимые (запретные) комбинации;
- единственно возможные комбинации (для прогнозирования значений в закрытых клетках).

Все выявленные комбинации записываются в отдельный список правил, которые являются производными правилами от базовых правил логической задачи.

Затем полученные правила анализируются и выявляются зависимости (для определения основных свойств отношений, применяемых в дискретной математике: рефлексивность, симметричность, транзитивность и др.). Это позволит компактнее сформировать уже синтезированные правила, а также провести синтез новых правил, таким образом сформировав набор шаблонов, по которым можно будет выявлять, есть ли в той или клетке мина.

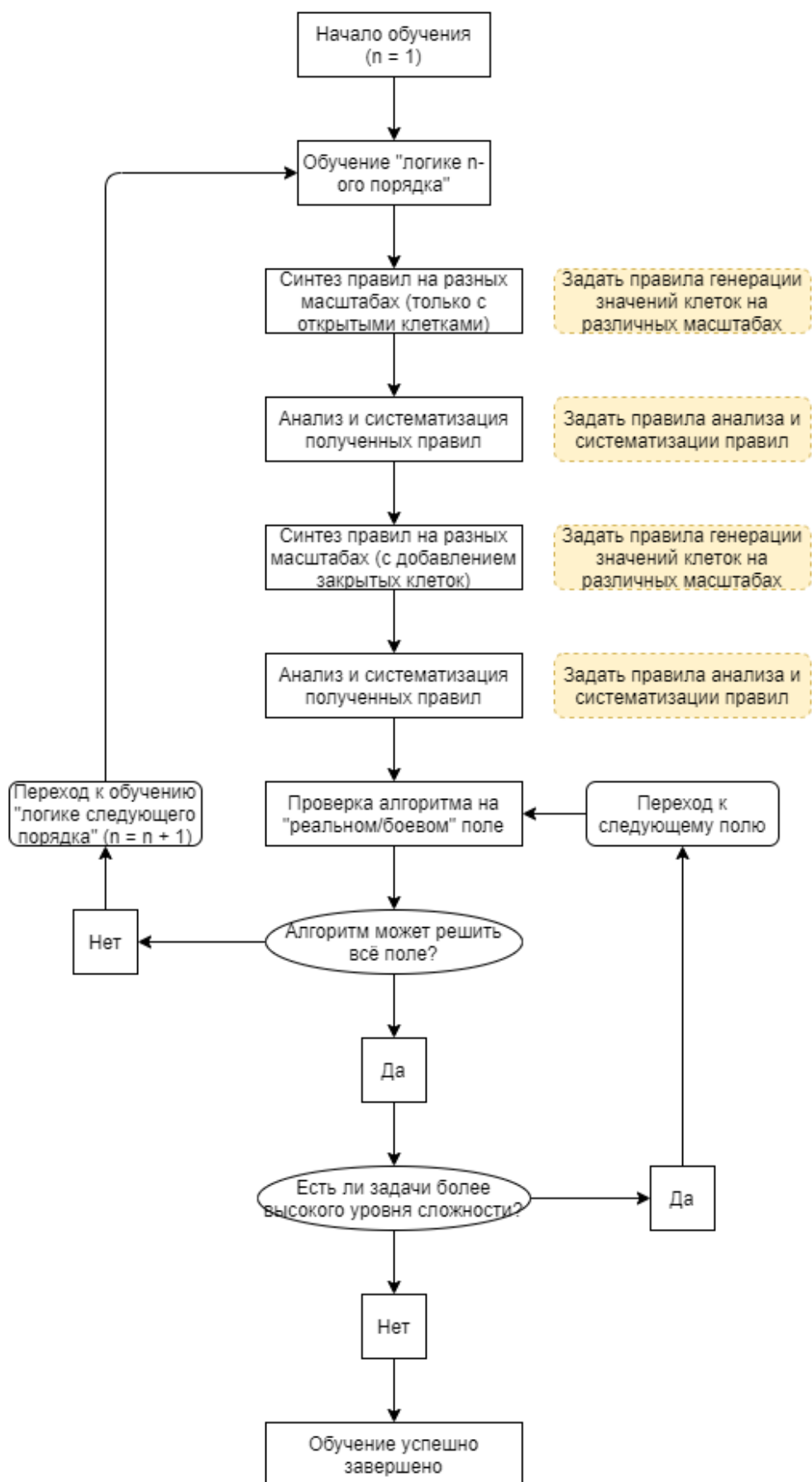


Рис. 1.Блок-схема разработанного алгоритма

Выявлять новые правила при помощи мини-полей будем итеративно. На первой итерации будут применяться только базовые правила игры. На последующих итерациях будут применяться правила, основанные на принципе «что, если», то есть, сначала, по определённым правилам, будет выдвигаться предположение для закрытой клетки (например: предположим, что в данной клетке мина) и, исходя из данного предположения, будут проверяться уже имеющиеся правила и синтезироваться новые.

Блок-схема данного метода представлена на рисунке 1.

Рассмотрим подробнее описание применения логики n-ого порядка.

Логика 1-го порядка основана на правиле поскольку в клетке 1 значение  $x$ , то в клетке 2 значение  $y$ . Пример логики 1-го порядка приведён на рисунке 2.

Пример логики 1-ого порядка				
	0	1	2	3
0	0	0	0	0
1	0	М	0	0
2	0	0	0	0
3	0	0	0	0

В клетке (1; 1) не может быть "мины", поскольку в клетке (0; 0) находится цифра 0

Рис. 2. Пример логики 1-го порядка

Логика 2-го порядка основана на следующем правиле. Исходя из базовых правил и правил, полученных в ходе применения логики 1-го порядка, установлено, что мина может быть в клетке 1 или в клетке 2. Тогда можно выделить 2 пункта для синтеза новых правил:

- Если мина находится в клетке 1, то, исходя из данного предположения, в соседних клетках будет определённый набор значений (назовём его *набор 1*). Если мина находится в клетке 2, то, исходя из данного предположения, в соседних клетках будет другой набор значений (назовём его *набор 2*). Тогда к *набору 1* и *набору 2*, а, точнее, к соответствующим закрытым значениям

клеток *набора 1* и *набора 2* можно применить логическую операцию **И**, чтобы выявить схожие значения.

- Для *набора 1* и *набора 2* производится применение правил низших порядков для выявления ошибочных значений.

Таким образом, если в первом пункте будут найдены схожие значения для закрытых клеток двух полей и/или во втором пункте будут найдены ошибки при применении правил низших порядков, полученные схемы будут занесены в список правил 2-го порядка.

Пример применения первого пункта логики 2-го порядка представлен на рисунке 3.

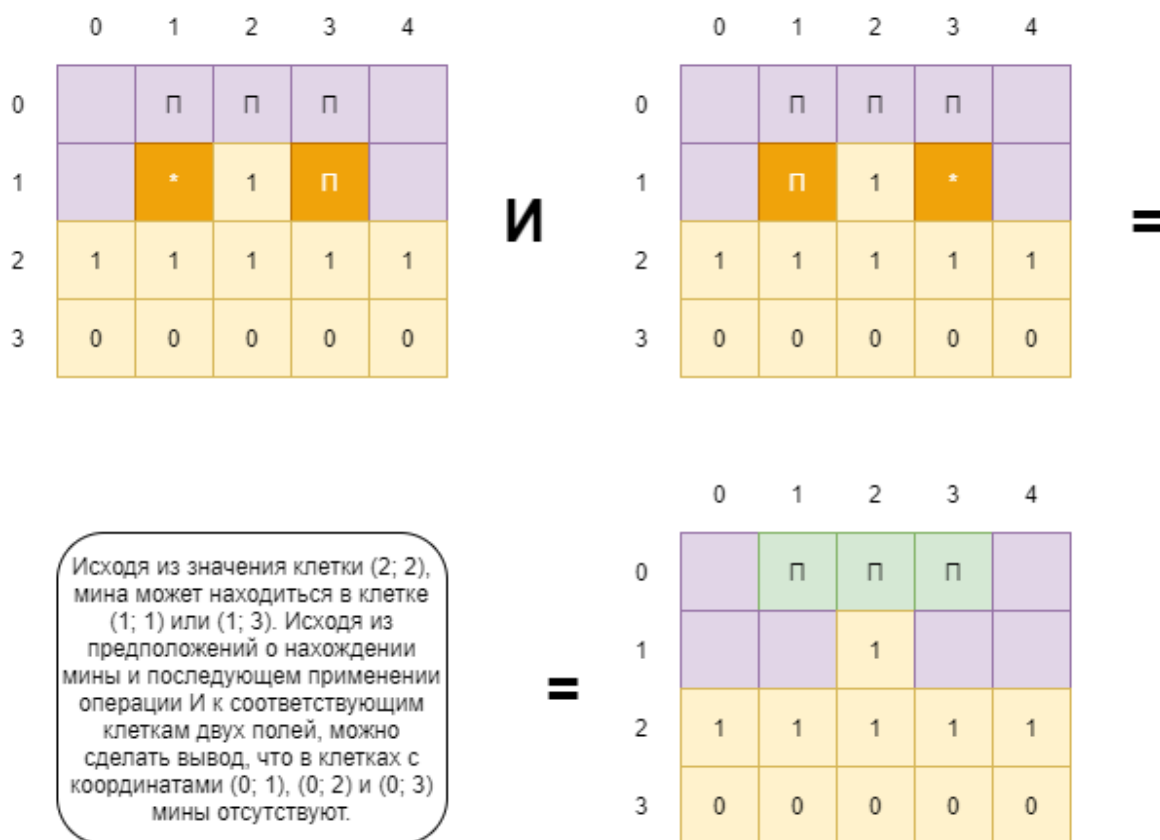


Рис. 3. Пример применения первого пункта логики 2-го порядка

Пример применения второго пункта логики 2-го порядка представлен на рисунке 4.

Пример логики 1-ого порядка				
	0	1	2	3
0	0	0	0	0
1	0	М	0	0
2	0	0	0	0
3	0	0	0	0
В клетке (1; 1) не может быть "мины", поскольку в клетке (0; 0) находится цифра 0				

Рис. 4. Пример применения второго пункта логики 2-го порядка

### 1.2.3 Анализ алгоритмов построения самообучающихся систем

Для определения предпочтительного метода для разработки самообучающегося алгоритма для решения логической задачи необходимо чётко определить критерии выбора.

Главным критерием является точность решения задачи. Задача, которая была выбрана в качестве задачи для реализации самообучающейся системы, имеет однозначное решение, поэтому для достижения максимальной эффективности важно, чтобы каждое поле головоломки решалось безошибочно.

Первый рассмотренный подход к разработке самообучающегося алгоритма основан на использовании вероятностной обучаемой стратегии. Данный алгоритм хоть и демонстрирует повышение эффективности при решении логической задачи «Сапёр», тем не менее, алгоритм не гарантирует безошибочность решения поля. Более того, данный алгоритм показывает невысокие значения доли побед для полей с числом мин  $q$ , равным 10 и 15 (0,694 и 0,469 соответственно), а для полей, с числом мин  $q$ , равным 20 алгоритм демонстрирует низкое значение доли побед (0,083). При заданных ограничениях применение данного подхода к разработке самообучающегося алгоритма недопустимо.

Второй рассмотренный подход основан на «запоминании» схем (шаблонов), которые увеличивают вероятность нахождения подходящего шаблона для того, чтобы найти хотя бы одну такую клетку, в которой отсутствует мина. Важным фактом является

то, что в случае, если самообучающаяся система не сможет найти хотя бы одну такую клетку, в которой отсутствует мина, то самообучающаяся система «уйдёт на дообучение», то есть будет тренироваться для нахождения зависимостей с применением «логики более высоких порядков» и, получив новые схемы, система продолжит пытаться найти решения поля. Таким образом, худшим вариантом при нахождении решения логической задачи может стать заикливание алгоритма, который с каждым разом будет искать шаблоны, используя логику и более и более высоких порядков. При этом возникать таких ситуаций, в которых поле решено с ошибкой, не должно.

Исходя из представленного анализа, можно сделать вывод о том, что разработанный метод для решения логической задачи является более подходящим, чем гибридная модель анализа ситуаций. Тем не менее, разработанный метод необходимо доработать, чтобы избежать заикливания при самообучении на полях, не подходящих под требования к исходным данным.

## 2 СПЕЦИАЛЬНАЯ ЧАСТЬ

Цель работы – разработка комплекса алгоритмов с элементами самообучения и их программной реализации для независимого от человека поиска эффективного решения казуально-логических игр (на примере игры «Minesweeper»/«Сапёр»).

Разработка специальной части начинается с описания правил игры «Сапёр». Далее, исходя из правил, будут представлены содержательная и математическая постановки задачи исследования. Исходя из них, будет разработана теоретическая часть, которая будет описывать связь входных и выходных данных. На основе теоретической части будет разработан алгоритм поиска эффективных решений, который впоследствии будет разделён на части (блоки), где далее для части блоков будет описан самообучающийся алгоритм, который будет выполнять функции данного блока. В завершении будет описано и представлено программное обеспечение, реализующее разработанный алгоритм с элементами самообучения, а также будут представлены результаты работы программы. Блок-схема специальной части представлена на рисунке 5.

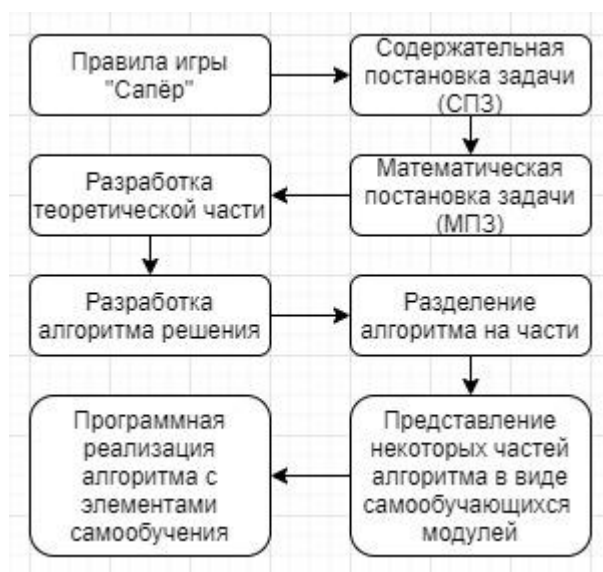


Рисунок 5. Блок-схема специальной части

Правила игры «Сапёр» представлены в приложении А.



## 2.2 Содержательная постановка задачи

Содержательная постановка задачи представляет из себя выжимку и краткое содержание правил игры. Таким образом, содержательная постановка задачи – это точная и небольшая по объёму формулировка, понятная читателю.

Основным элементом игры является поле с заданными значениями длины и ширины. Элементами поля являются клетки. Клетка может быть открыта или закрыта. В открытой клетке хранится целое значение  $[0; 8]$  или мина. Значения  $[0; 8]$  в клетке означает количество мин, находящихся в соседних клетках. В закрытой клетке находится флаг мины, знак вопроса или не находится ничего – пользователь сам решает, что будет находиться в закрытой клетке. Если клетка закрыта, то неизвестно, что в ней находится, в противном случае – известно.

На вход подаётся поле, большая часть клеток которого закрыты, а также число, означающее общее количество мин, находящихся на поле. Пользователь может открыть закрытую клетку, чтобы узнать значение, которое в ней хранится. Если это значение  $[0; 8]$ , то данное значение отображается в открываемой клетке. Если в открываемой клетке находится мина, то она также отображается в открываемой клетке, но при этом игра заканчивается поражением. Для победы необходимо открыть все клетки поля, в которых отсутствуют мины. Цель: победить.

Также определено, что решение задачи является детерминированным.

Таким образом, исходными данными для решения поставленной задачи являются:

- поле с заданными значениями длины и ширины, состоящее из клеток;
- общее количество мин, которое находится на поле.

Для исходных данных заданы следующие характеристики и логические связи:

- клетка может быть открыта или закрыта;
- если клетка закрыта, то неизвестно, что в ней находится, в противном случае – известно;
- в открытой клетке хранится целое значение  $[0; 8]$  или мина;
- в закрытой клетке находится флаг мины, знак вопроса или не находится ничего – пользователь сам решает, что будет находиться в закрытой клетке;
- пользователь может открыть клетку, чтобы узнать значение в ней;
- набор условий, определяющие правила игры, а именно:
  - цифра в клетке определяет количество мин в соседних клетках;

- условие победы (цель игры): открыть все клетки поля, не содержащие мины;
- условие поражения: открыть клетку с миной;
- решение задачи является детерминированным.

## 2.3 Математическая постановка задачи

Математическая постановка задачи представляет из себя переработанную содержательную постановку задачи, представленную с помощью математических терминов. Большинство терминов, которые представлены в данном разделе можно найти в тезаурусе, но, для лучшего понимания, определение некоторых терминов будет приведено сразу.

Основным элементом игры является кортеж кортежей  $F$  (field – поле), длина кортежа которого равна  $l$  (length – длина), а длина каждого подкортежа равна  $w$  (width – ширина).

Элементами подкортежей кортежа  $F$  являются кортежи  $C$  (cell – клетка), состоящие из 4-х элементов:

$$\langle cv \in CCV, ov \in OCV, v \in CCV \cup OCV, s \in St \rangle,$$

где  $s$  – (status – состояние) – элемент множества  $St$ , которое отвечает за состояние клетки – открыта она или закрыта,

$cv$  – (close value – закрытое значение) – элемент множества  $CCV$ , которое отвечает за значение, находящееся в закрытой клетке,

$ov$  (open value – открытое значение) – элемент множества  $OCV$ , которое отвечает за значение, находящееся в открытой клетке,

$v$  (value – значение) – элемент множества  $CCV \cup OCV$ , означающий значение, находящееся в клетке.

Множество  $CCV$  (close cell values – значения закрытых клеток) состоит из значений  $MF$  (mine flag – флаг мины),  $Q$  (question – вопрос/сомнение),  $E$  (emptiness – пустота).

Множество  $OCV$  (open cell values – значения открытых клеток) состоит из множества целых чисел  $[0; 8]$  и элемента  $M$  (mine – мина).

Множество  $St$  (status – состояние) состоит из двух элементов:  $C$  (close – закрыта) и  $O$  (open – открыта).

Изначально любой элемент  $cv$  кортежа  $C$  имеет значение  $E$ . Изначально, если элемент  $s$  кортежа  $C$  имеет значение  $C$ , то значение элемента  $v$  равно значению элемента  $cv$ . Если элемент  $s$  кортежа  $C$  имеет значение  $O$ , то значение элемента  $v$  равно значению элемента  $ov$ . Также пользователь не имеет доступ к элементам  $cv$  и  $ov$  кортежа  $C$ .

Определим множество  $B = \{0, 1\}$  (binary – двоичный) и кортеж кортежей  $F_b$  (field binary – двоичное поле), длина кортежа которого равна  $l$ , а длина каждого подкортежа равна  $w$  (аналогично кортежу кортежей  $F$ ). Элементами подкортежей кортежа  $F_b$  являются элементы множества  $B$ .

Зададим несколько функций над множеством  $F$ :

$$sign_M = f(x, y): F \rightarrow F_b = \begin{cases} 1, \text{ если } ov \in F_c[x, y] = M \\ 0, \text{ если } ov \in F_c[x, y] \neq M \end{cases} \quad (1)$$

$$sign_s = f(x, y): F \rightarrow F_b = \begin{cases} 1, \text{ если } s \in F_c[x, y] = O \\ 0, \text{ если } s \in F_c[x, y] = C \end{cases} \quad (2)$$

Определим кортеж кортежей  $F_{ac}$  (adjacent cells of field – соседние клетки поля), длина кортежа которого равна  $l$ , а длина каждого подкортежа равна  $w$  (аналогично кортежу кортежей  $F$ ). Элементами подкортежей кортежа  $F_{ac}$  являются кортежи, элементами которого являются кортежи  $C$ .

Определим функцию  $AC$  (adjacent cells – соседние клетки) над множеством  $F$ :

$$AC = f(x, y): F \rightarrow F_{ac} = \begin{cases} \langle F[x-1][y-1], F[x-1][y], F[x-1][y+1], F[x][y-1], \\ F[x][y+1], F[x+1][y-1], F[x+1][y], F[x+1][y+1] \rangle, \\ \text{если } 0 < x < l, 0 < y < w \\ \langle F[x][y-1], F[x][y+1], F[x+1][y-1], F[x+1][y], \\ F[x+1][y+1] \rangle, \text{ если } x = 0, 0 < y < w \\ \langle F[x-1][y-1], F[x-1][y], F[x-1][y+1], F[x][y-1], \\ F[x][y+1] \rangle, \text{ если } x = l, 0 < y < w \\ \langle F[x-1][y], F[x-1][y+1], F[x][y+1], F[x+1][y], \\ F[x+1][y+1] \rangle, \text{ если } 0 < x < l, y = 0 \\ \langle F[x-1][y-1], F[x-1][y], F[x][y-1], F[x+1][y-1], \\ F[x+1][y] \rangle, \text{ если } 0 < x < l, y = w \\ \langle F[x][y+1], F[x+1][y], F[x+1][y+1] \rangle, \text{ если } x = 0, y = 0 \\ \langle F[x][y-1], F[x+1][y+1], F[x+1][y] \rangle, \text{ если } x = 0, y = w \\ \langle F[x-1][y], F[x-1][y+1], F[x][y+1] \rangle, \text{ если } x = l, y = 0 \\ \langle F[x-1][y-1], F[x-1][y], F[x][y-1] \rangle, \text{ если } x = l, y = w \end{cases} \quad (3)$$

Определим кортеж  $T_b$  (tuple binary – двоичный кортеж), элементы которого принадлежат множеству  $B$ . Также определим кортеж кортежей  $F_{bac}$  (binary adjacent cells of field – двоичные соседние клетки поля), длина кортежа которого равна  $l$ , а длина каждого подкортежа равна  $w$  (аналогично кортежу кортежей  $F$ ). Элементами подкортежей кортежа  $F_{bac}$  являются кортежи  $T_b$ . Длины кортежей подкортежей кортежей  $F_{bac}$  будем обозначать как  $n[x, y]$ .

Определим функцию над множеством  $F_{ac}$ :

$$sign_{M_{ov}} = f(x, y, z): F_{ac} \rightarrow F_{bac} = \begin{cases} 1, \text{ если } ov \in F_{ac}[x][y][z] = M \\ 0, \text{ если } ov \in F_{ac}[x][y][z] \neq M \end{cases} \quad (4)$$

Определим функцию  $R$  (reflection – отражение) над множеством  $F$ :

$$R = f(x, y): F \rightarrow F = \begin{cases} < v = v, cv = cv, ov = \sum_{i=0}^{n[x,y]} F_{acb}[x][y][i], s = s >, \\ & \text{если } ov \in F[x][y] \neq M \\ < v = v, cv = cv, ov = ov, s = s >, \text{если } ov \in F[x][y] = M \end{cases} \quad (5)$$

Также задаётся множество состояний игры  $GS$  (game status – статус/состояние игры), состоящее из 3-ёх элементов:  $V$  (victory – победа),  $D$  (defeat – поражение) и  $N/O$  (not over – игра не окончена). Определим переменную  $gs$  (game status – статус/состояние игры), значение которой принадлежит множеству  $GS$ . В начале игры переменная  $gs$  имеет значение  $N/O$ . Игра считается завершённой, когда значение переменной  $gs$  изменится на  $V$  (в таком случае считается, что игра завершилась победой) или на  $D$  (в таком случае считается, что игра завершилась поражением).

Общее количество мин на поле задаётся параметром  $tm$  (total mines – общее количество мин), значение которого рассчитывается по следующей формуле:

$$tm = \sum_{i=0}^l \sum_{j=0}^w \begin{cases} 1, \text{если } ov \in F[i][j] = M \\ 0, \text{если } ov \in F[i][j] \neq M \end{cases} \quad (6)$$

Зададим параметр  $cwm$  (cells without mines – клеток без мин) и зададим переменную  $occ$  (open cell current – текущее количество открытых клеток). Параметр  $cwm$  рассчитывается по следующей формуле:

$$cwm = l * w - tm \quad (7)$$

Начальное значение переменной  $occ$  рассчитывается по следующей формуле:

$$occ = \sum_{i=0}^l \sum_{j=0}^w \begin{cases} 1, \text{если } s \in F[i][j] = O \\ 0, \text{если } s \in F[i][j] = C \end{cases} \quad (8)$$

Также определим множество пользовательских допустимых изменений состояний (переходов)  $UT$  (user transition – пользовательский переход) и множество условных переходов  $CT$  (conditional transition – условный переход). Будем называть пользовательским переходом такое изменение переменной и/или элемента или элементов тех или иных множеств, которое может осуществлять пользователь. Данное множество будет состоять из следующих элементов:

- Для  $F: v \neq MF, s: C \rightarrow O; v, s \in F[x][y]$ ,
- Для  $F: s = C, cv: E \rightarrow MF; s, cv \in F[x][y]$ ,
- Для  $F: s = C, cv: E \rightarrow Q; s, cv \in F[x][y]$ ,
- Для  $F: s = C, cv: MF \rightarrow E; s, cv \in F[x][y]$ ,

- Для  $F: s = C, cv: MF \rightarrow Q; s, cv \in F[x][y]$ ,
- Для  $F: s = C, cv: Q \rightarrow E; s, cv \in F[x][y]$ ,
- Для  $F: s = C, cv: Q \rightarrow MF; s, cv \in F[x][y]$ .

Назовём условным переходом такое изменение переменной и/или элемента или элементов тех или иных множеств, которое происходят автоматически, исходя из изменения переменной и/или элемента или элементов множеств после пользовательского перехода. Условный переход состоит из двух частей: условной (которое, в свою очередь является пользовательским переходом) и действительной части (определяет переход, который осуществляется при условии выполнения указанного пользовательского перехода). Данное множество будет состоять из следующих элементов:

- ЕСЛИ для  $F: v \neq MF, s: C \rightarrow O; v, s \in F[x][y]$ ,  
ТО для  $F: v: v \rightarrow ov, v \in F[x][y]$ ,
- ЕСЛИ для  $F: s = C, cv: E \rightarrow MF; s, cv \in F[x][y]$ ,  
ТО для  $F: cv: cv \rightarrow v, cv \in F[x][y]$ ,
- ЕСЛИ для  $F: s = C, cv: E \rightarrow Q; s, cv \in F[x][y]$ ,  
ТО для  $F: cv: cv \rightarrow v, cv \in F[x][y]$ ,
- ЕСЛИ для  $F: s = C, cv: MF \rightarrow E; s, cv \in F[x][y]$ ,  
ТО для  $F: cv: cv \rightarrow v, cv \in F[x][y]$ ,
- ЕСЛИ для  $F: s = C, cv: MF \rightarrow Q; s, cv \in F[x][y]$ ,  
ТО для  $F: cv: cv \rightarrow v, cv \in F[x][y]$ ,
- ЕСЛИ для  $F: s = C, cv: Q \rightarrow E; s, cv \in F[x][y]$ ,  
ТО для  $F: cv: cv \rightarrow v, cv \in F[x][y]$ ,
- ЕСЛИ для  $F: s = C, cv: Q \rightarrow MF; s, cv \in F[x][y]$ ,  
ТО для  $F: cv: cv \rightarrow v, cv \in F[x][y]$ ,
- ЕСЛИ для  $F: s: C \rightarrow O, ov = M; s, ov \in F[x][y]$ ,  
ТО  $gs: N/O \rightarrow D$ ,
- ЕСЛИ для  $F: s: C \rightarrow O, ov \neq M; s, ov \in F[x][y]$ ,  
ТО  $occ: occ \rightarrow occ + 1$ ,
- ЕСЛИ для  $occ: occ \rightarrow occ + 1, cwm = occ$ ,  
ТО  $gs: N/O \rightarrow V$ ,
- ЕСЛИ для  $F: ov = 0, s: C \rightarrow O; ov, s \in F[x][y]$ ,  
ТО  $\forall F_{ac}[x][y] \subset F: v: \rightarrow ov, s: C \rightarrow O, v, s \in F_{ac}[x][y] \subset F$ .

Итак, входными данными являются:

- кортеж кортежей  $F$ , длина кортежа которого равна 1, а длина каждого подкортежа равна  $w$
- значение параметра  $tm$ .

Успешным выполнением задачи является достижение переменной  $gs$  значения  $V$ .  
 Неуспешным выполнением задачи является достижение переменной  $gs$  значения  $D$ .

## 2.4 Описание алгоритма

### 2.4.1 Общий алгоритм

Исходя из выводов предыдущего пункта, целью данной игры является достижение переменной  $gs$  значения  $V$ . Рассмотрим теперь шаги, с помощью которых возможно достичь поставленной цели. Для этого будем задавать вопросы и отвечать на них с применением имеющихся данных. Сначала введём следующие условные обозначения:

- В (с) – вопрос с точки зрения содержательной постановки задачи.
- В (м) – вопрос с точки зрения математической постановки задачи.
- О (с) – ответ с точки зрения содержательной постановки задачи.
- О (м) – ответ с точки зрения математической постановки задачи.

Начнём задавать вопросы с вопроса о возможных способах достижения цели.

В (с): Как достичь цели (победы)?

О (с): Открыть последнюю закрытую клетку поля.

О (м): С помощью осуществления перехода:

ЕСЛИ для  $oss: oss \rightarrow oss + 1, cwt = oss$ ,  
 ТО  $gs: N/O \rightarrow V$ .

В (с): Как открыть последнюю закрытую клетку поля?

В (м): Как осуществить переход:

для  $oss: oss \rightarrow oss + 1, cwt = oss$ ?

О (с): Найти последнюю закрытую клетку поля, в которой отсутствует мина.

О (м): С помощью осуществления перехода:

ЕСЛИ для  $F: s: C \rightarrow O, ov \neq M; s, ov \in F[x, y]$ ,  
 ТО  $oss: oss \rightarrow oss + 1, cwt = oss$ .

В (о): Как определить, что в закрытой клетке поля отсутствует мина?

В (м): Как осуществить переход:

для  $F: s: C \rightarrow O, ov \neq M; s, ov \in F[x, y]$ ?

О (с): С помощью применения тождественного отображения  $R$  множества  $F$  в себя же.

## 2.4.2 Первое правило вычисления значения в закрытой клетке

Рассмотрим тождественное отображение  $R$  кортежа  $F$  в себя же. Поскольку при  $ov \in F[x][y] = M$  каждый элемент кортежа  $F[x][y]$  отображается в себя, данное отображение мы рассматривать не будем. Рассмотрим подробнее отображение при  $ov \in F[x][y] \neq M$ . Элементы  $v, cv, s$  кортежа  $F[x][y]$  также аналогично отображаются в себя же, а элемент  $ov$  кортежа  $F[x][y]$  отображается в себя же с применением функции для кортежа  $F_{bac}$ . Рассмотрим данное отображение подробнее.

$$\begin{aligned} R_m &= f(x, y): ov \in F[x][y] \rightarrow ov \in F[x][y] = \\ &= \sum_{i=0}^{n[x,y]} F_{bac}[x][y][i], \text{ если } ov \in F[x][y] \neq M \end{aligned} \quad (9)$$

Данное выражение можно переписать следующим образом:

$$\begin{aligned} R_m &= f(x, y): ov \in F[x][y] \rightarrow ov \in F[x][y] = \\ &= \sum_{i=0}^{n[x,y]} \begin{cases} 1, \text{ если } ov \in F_{ac}[x][y][i] = M \\ 0, \text{ если } ov \in F_{ac}[x][y][i] \neq M \end{cases} \end{aligned} \quad (10)$$

Таким образом, для элемента  $ov \in F[x][y] \subset F_c$  определено следующее равенство:

$$ov = \sum_{i=0}^{n[x,y]} \begin{cases} 1, \text{ если } ov \in F_{ac}[x][y][i] = M \\ 0, \text{ если } ov \in F_{ac}[x][y][i] \neq M \end{cases} \quad (11)$$

Перепишем данное равенство следующим образом:

$$ov = \sum_{i=0}^{n[x,y]} \{ 1, \text{ если } ov \in F_{ac}[x][y][i] = M + \quad (12)$$

$$+ \sum_{i=0}^{n[x,y]} \{ 0, \text{ если } ov \in F_{ac}[x][y][i] \neq M$$

$$\begin{aligned} ov &= 1 * \sum_{i=0}^{n[x,y]} ov \in F_{ac}[x][y][i] = M + \\ &+ 0 * \sum_{i=0}^{n[x,y]} ov \in F_{ac}[x][y][i] \neq M \end{aligned} \quad (13)$$

Выполним следующую замену:

$$k_1 = \sum_{i=0}^{n[x,y]} ov \in F_{ac}[x][y][i] = M, k_2 = \sum_{i=0}^{n[x,y]} ov \in F_{ac}[x][y][i] \neq M \quad (14)$$

Параметр  $k_1$  означает количество мин в соседних с выбранной клеткой клетках. Параметр  $k_2$  означает количество соседних с выбранной клеткой клеток, в которых отсутствуют мины. Тогда предыдущее равенство можно записать следующим образом:

$$ov = 1 * k_1 + 0 * k_2 = 1 * k_1 = k_1 \quad (15)$$

Поскольку часть соседних клеток с выбранной клеткой могут быть закрыты, а для закрытых клеток значение  $ov$  скрыто от пользователя, то можно провести следующую замену:

$$k_1 = k_{11} + k_{12}, k_2 = k_{21} + k_{22} \quad (16)$$

где  $k_{11}$  – количество открытых с выбранной клеткой клеток без мины,

$k_{12}$  – количество закрытых с выбранной клеткой клеток без мины,

$k_{21}$  – количество закрытых с выбранной клеткой клеток с флагами мины,

$k_{22}$  – количество закрытых с выбранной клеткой клеток с минами (без учёта закрытых клеток с флагом мины).

Таким образом,  $k_{12}$  и  $k_{22}$  – неизвестные переменные, а  $k_{11}$  и  $k_{21}$  – рассчитываемые параметры. Получается, что формулу () можно записать следующим образом:

$$v = 0 * (k_{11} + k_{12}) + 1 * (k_{21} + k_{22}) \quad (17)$$

Исходя из того, что известно количество соседних с выбранной клеткой клеток, можно определить следующее равенство:

$$(k_{11} + k_{12}) + (k_{21} + k_{22}) = |F_{ac}[x, y]| \quad (18)$$

Таким образом, можно определить следующую систему уравнений:

$$\begin{cases} v = 0 * (k_{11} + k_{12}) + 1 * (k_{21} + k_{22}) \\ (k_{11} + k_{12}) + (k_{21} + k_{22}) = |F_{ac}[x, y]| \end{cases} \quad (19)$$

Из данной системы уравнений возможно вычислить значения переменных  $k_{12}$  и  $k_{22}$ .

Поскольку для каждого поля существует детерминированное решение, то для каждой закрытой клетки поля определено одно и только одно значение из множества допустимых значений закрытой клетки, то есть из множества  $OCV$ .

Исходя из того, что условием поражения является открытие клетки с миной, а необходимым условием победы является открытие закрытой клетки, в которой отсутствует мина (то есть находится значение  $x \in [0; 8]$ ), определим следующую функцию:

$$H = f(x): OCV \rightarrow B = \begin{cases} 0, \text{ если } x \in OCV \neq M \\ 1, \text{ если } x \in OCV = M \end{cases} \quad (20)$$

Определим теперь количество возможных комбинаций размещения элементов множества  $B$  в выбранной закрытой клетке:



$$A_n^k = \frac{n!}{(n-k)!} \quad (21)$$

где  $n = 2$  – количество элементов множества  $B$ ,

$k = 1$  – количество закрытых клеток, в которых размещаются элементы множества  $B$ .

Таким образом, можно вычислить количество комбинаций размещения:

$$A_2^1 = \frac{2!}{(2-1)!} = \frac{2!}{1!} = 2 \quad (22)$$

Теперь определим количество допустимых комбинаций размещения элементов множества  $B$  в соседних с выбранной открытой клеткой закрытых клетках. Если в соседних закрытых клетках находится  $k_2$  мин, то количество возможных комбинаций определяется числом сочетаний  $C_n^{k_2}$ , где  $n$  – количество соседних закрытых клеток.

Поскольку для каждого поля существует детерминированное решение, то для каждого элемента  $ov$  кортежа закрытых клеток однозначно соответствует значение  $x \in OCV$ . Исходя из этого, существует только одна комбинация размещения элементов множества  $B$  в соседних с выбранной открытой клеткой закрытых клетках. В таком случае, необходимо определить, при каких значениях  $k_2$  и  $n$   $C_n^{k_2} = 1$ .

$$C_n^{k_2} = \frac{n!}{k!(n-k)!} = 1 \quad (23)$$

Данное равенство верно при  $n = k$ . Таким образом, если количество мин в соседних клетках или количество соседних клеток без мин (соседних с выбранной открытой клеткой) равно количеству соседних закрытых клеток, то возможно однозначно определить значения в соседних закрытых клетках. Исходя из этого, можно сформировать **первое правило открытия закрытых клеток**:

Если некоторая клетка поля  $F$  открыта и в ней не находится мина, то:

- 1) Если число в клетке равно количеству соседних закрытых клеток, то в данных закрытых клетках находятся мины.
- 2) Если число в клетке равно количеству соседних закрытых клеток с флагом мины, то в других соседних закрытых клетках мин нет.

### 2.4.3 Применение гипотез для одной и нескольких закрытых клеток

В предыдущем разделе удалось определить два метода проверки гипотез:

- Проверка гипотез для одной выбранной закрытой клетки
- Проверка гипотез для нескольких соседних с выбранной открытой клеткой закрытых клеток.

Теперь возникает необходимость в том, чтобы определить, как связаны методы между собой и в том, чтобы определить более эффективный метод.

Сначала рассмотрим первый метод: проверка гипотез для одной выбранной закрытой клетки. Для этого определим поле F, состоящее из шести клеток и для большей наглядности изобразим его (рисунок 6). Синим цветом будем обозначать закрытые клетки, белым – открытые, а зелёным – открытую клетку, с помощью которой рассчитаем возможные комбинации в соседних закрытых клетках. Точка в закрытой клетке означает, что для выбранной закрытой клетки приводится применение двух противоположных гипотез. Исходя из значения в зелёной открытой клетке (2) и исходя из количества соседних с ней закрытых клеток (3), в данных клетках должно находиться 2 мины и 1 клетка должна быть пустой. Исходя из рисунка 6, для соседних с зелёной клеткой закрытых клеток можно полностью рассчитать возможную комбинацию при условии, что для закрытой клетки, в которой находится точка, будет определена гипотеза об отсутствии мины в данной клетке. В противном случае будет необходимо проверять ещё одну гипотезу для оставшихся закрытых клеток.

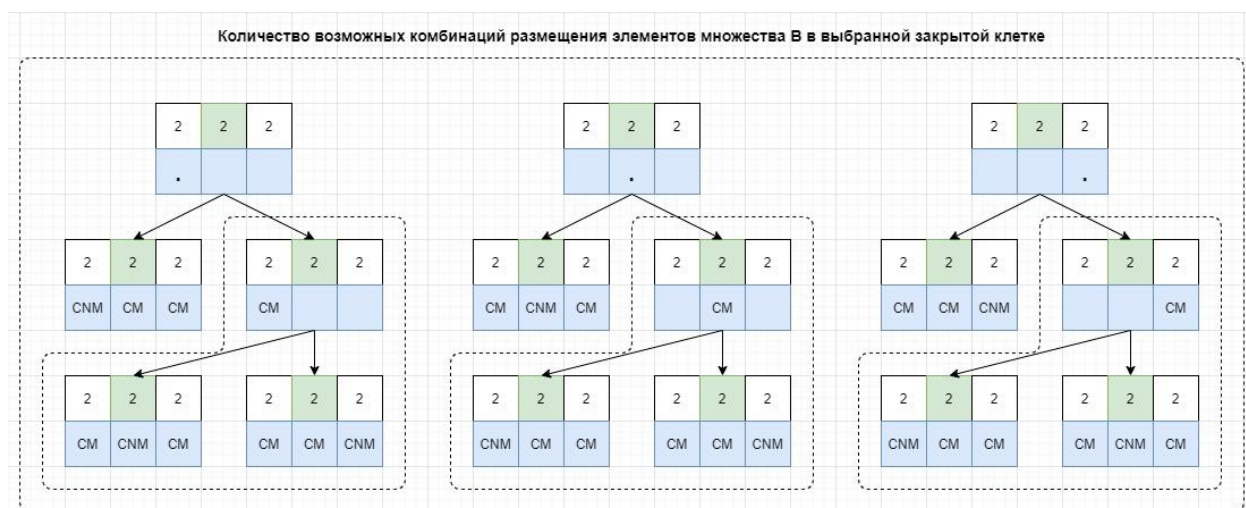


Рисунок 6. Проверка гипотез для одной выбранной закрытой клетки

Определим теперь возможные комбинации для нескольких соседних с выбранной открытой клеткой закрытых клеток (рисунок 7). Всего их будет 3. Можно заметить, что данные комбинации полностью совпадают с теми комбинациями, которые возникают в случае проверки гипотезы об отсутствии мины в закрытой клетке для каждой закрытой клетки (рисунок 6). Таким образом, можно предположить, что при проверке менее вероятной гипотезы для каждой из соседних с выбранной открытой клеткой закрытых клеток (определим как менее вероятную гипотезу такую гипотезу, для которой количество

выделенных соседних закрытых клеток меньше количества выделенных соседних закрытых клеток для противоположной гипотезы), получается кортеж, который аналогичен кортежу, полученный при размещении допустимых комбинаций элементов множества В в соседних с выбранной открытой клеткой закрытых клетках. Проверим данное предположение.

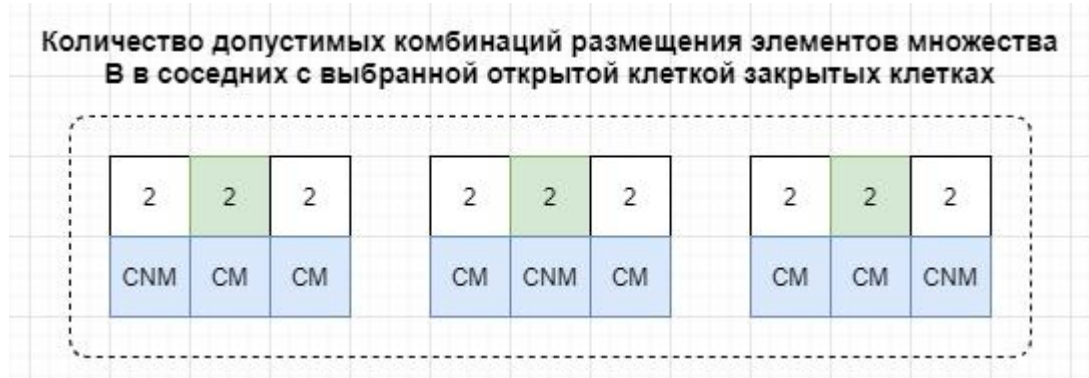


Рисунок 7. Проверка гипотез для нескольких соседних с выбранной открытой клеткой закрытых клеток

Если общее количество соседних с выбранной открытой клеткой закрытых клетках равно  $n$ , количество мин в них равно  $k_1$ , а количество клеток без мин –  $k_2$ , то число размещений определяется формулой:

$$C_n^{k_1} = \frac{n!}{k_1! (n - k_1)!} = C_n^{k_2} = \frac{n!}{k_2! (n - k_2)!} \quad (24)$$

Если  $k_1 \neq n$  и  $k_2 \neq n$ , то однозначно невозможно определить значения во всех соседних закрытых клетках.

Предположим, что  $k_1 \neq n$  и  $k_2 \neq n$ . Тогда введём гипотезу о том, что в одной из соседних закрытых клеток находится мина. Рассчитаем теперь количество возможных комбинаций в оставшихся соседних закрытых клетках:

$$C_{n-1}^{k_1-1} = \frac{(n-1)!}{(k_1-1)! (n-1-(k_1-1))!} = \frac{(n-1)!}{(k_1-1)! (n-k_1)!} \quad (25)$$

$$C_{n-1}^{k_1-1} = C_{n-1}^{k_2} = \frac{(n-1)!}{k_2! (n-1-k_2)!}$$

Таким образом, если  $k_1 - 1 = n - 1$  или  $k_2 = n - 1$ , то в таком случае, можно однозначно определить значения в оставшихся соседних закрытых клетках. Однако,  $k_1 - 1 \neq n - 1$ , поскольку ранее было выдвинуто предположение, что  $k_1 \neq n$ . В таком случае, чтобы однозначно рассчитать значения в оставшихся соседних клетках, необходимо, чтобы  $k_2 = n - 1$  или, по-другому,  $k_2 + 1 = n$ , то есть, чтобы количество соседних закрытых

клеток было бы на 1 больше, чем количество соседних закрытых клеток без мин. В противном случае необходимо вводить как минимум ещё одну гипотезу, чтобы однозначно рассчитать значения в оставшихся соседних закрытых клетках.

Теперь введём гипотезу о том, что в одной из соседних закрытых клеток отсутствует мина. Рассчитаем теперь количество возможных комбинаций в оставшихся соседних закрытых клетках:

$$C_{n-1}^{k_1} = \frac{(n-1)!}{(k_1)!(n-1-k_1)!}$$

$$C_{n-1}^{k_2-1} = C_{n-1}^{k_1} = \frac{(n-1)!}{(k_2-1)!(n-1-(k_2-1))!} = \frac{(n-1)!}{(k_2-1)!(n-k_2)!} \quad (26)$$

Таким образом, если  $k_1 = n-1$  или  $k_2-1 = n-1$ , то в таком случае, можно однозначно определить значения в оставшихся соседних закрытых клетках. Однако,  $k_2-1 \neq n-1$ , поскольку ранее было выдвинуто предположение, что  $k_2 \neq n$ . В таком случае, чтобы однозначно рассчитать значения в оставшихся соседних клетках, необходимо, чтобы  $k_1 = n-1$  или, по-другому,  $k_1+1 = n$ , то есть, чтобы количество соседних закрытых клеток было бы на 1 больше, чем количество соседних закрытых клеток с минами. В противном случае необходимо вводить как минимум ещё одну гипотезу, чтобы однозначно рассчитать значения в оставшихся соседних закрытых клетках. Определим, сколько необходимо ввести гипотез для однозначного определения значений в соседних с выбранной открытой клеткой закрытых клетках.

Получается, чтобы не вводить ещё одну гипотезу, для расчёта значений во всех соседних с выбранной открытой клеткой закрытых клетках необходимо для выбранной закрытой клетки вводить наименее вероятную гипотезу, исходя из значения в соседних открытых клетках.

Определим теперь, какое минимальное количество гипотез потребуется для того, чтобы рассчитать значения в соседних с выбранной открытой клеткой закрытых клетках. Для этого продолжим далее вводить гипотезы для соседних с выбранной открытой клеткой закрытых клеток до тех пор, пока не получится однозначно рассчитать значения во всех соседних с выбранной открытой клеткой закрытых клетках. Критерием останова будет являться условие

$$k_1 + x_1 = n \text{ или } k_2 + x_2 = n, \quad (27)$$

где  $x_1$  — это количество принятых гипотез для соседних с выбранной открытой клеткой закрытых клеток о том, что в выбранной закрытой клетке находится мина,

$x_2$  — это количество принятых гипотез для соседних с выбранной открытой клеткой закрытых клеток о том, что в выбранной закрытой клетке отсутствует мина.

При этом необходимо выполнить критерий оптимизации

$$x_1 + x_2 \rightarrow \min \quad (28)$$

Исходя из формулы () можно рассчитать значения  $x_1$  и  $x_2$

$$x_1 = n - k_1, x_2 = n - k_2 \quad (29)$$

Поскольку достаточно выполнения одного из условий (), критерий оптимизации можно переписать следующим способом

$$\min(x_1, x_2), \min(n - k_1, n - k_2) \quad (30)$$

Таким образом, минимальное количество гипотез, необходимых для расчёта значений в соседних с выбранной открытой клеткой закрытых клетках равно  $\min(n - k_1, n - k_2)$ .

Также, исходя из расчётов, можно сказать, что комбинации значений в соседних с выбранной открытой клеткой закрытых клетках – это частные случаи применения определённых гипотез для данных закрытых клеток.

Таким образом, с точки зрения эффективности будет лучше проверять гипотезы для одной выбранной закрытой клетки, чем для всех соседних с выбранной открытой клеткой соседних клеток.

#### 2.4.4 Второе правило вычисления значения в закрытой клетке

В предыдущем разделе удалось определить, что эффективнее проверять гипотезы для одной выбранной закрытой клетки, чем для всех соседних с выбранной открытой клеткой соседних клеток. Теперь возникает необходимость «проверять» комбинации значений для одной закрытой клетки (будем называть «проверку» комбинаций значений для закрытых клеток проверкой гипотез, где проверяемой гипотезой является одна из возможных комбинаций значений для закрытых клеток) на их допустимость. Для этого рассмотрим несколько методов, с помощью которых можно однозначно вычислить значения в закрытых клетках.

Первый метод основывается на следующей теореме: «Если при проверке гипотезы для одной выбранной закрытой клетки поля обнаруживается хотя бы одна некорректная клетка, то данная гипотеза должна быть отклонена». Рассмотрим подробнее данную теорему. Предположим, что в выбранной закрытой клетке поля находится или отсутствует мина. Исходя из данного предположения, будем вычислять значения в соседних закрытых с выбранной клеткой клетках поля, затем в соседних с соседними закрытыми клетками (если это возможно) и т. д. Если при вычислении значений в закрытых клетках поля найдётся хотя бы одна открытая клетка, для которой не выполняется условие, что число в

клетке равно количеству соседних клеток, в которых находится мина, то предположение было неверным.

Исходя из описания метода, возникает вопрос о необходимости определить очерёдность выбора закрытых клеток для проверки гипотез. Однозначный алгоритм для определения очерёдности закрытых клеток для проверки трудно определить, поскольку сложно выявить однозначные зависимости между свойствами закрытой клетки и вероятностью успешной проверки одной из гипотез для данной клетки. Поэтому при определении приоритетных клеток для проверки гипотез воспользуемся имеющимся опытом при решении полей игры «Сапёр».

Сначала будем выбирать такие закрытые клетки, для которых количество соседних закрытых клеток минимальное количество. В случае, если таких закрытых клеток будет несколько, будем ориентироваться на среднее арифметическое значение, рассчитываемое для соседних открытых клеток, которое показывает вероятность наличия/отсутствия мины в случайно выбранной соседней с выбранной открытой клеткой закрытой клетки. Таким образом, определим следующие шаги:

- 1) Для каждой закрытой клетки поля рассчитываем количество соседних закрытых клеток.
- 2) Сортируем список закрытых клеток поля по увеличению количества соседних закрытых клеток
- 3) Для клеток списка, для которых количество соседних закрытых клеток поля одинаково, получаем список соседних открытых клеток поля и их количество  $n_o$ .
- 4) Для каждой открытой клетки поля рассчитываем значения  $n, k_1, k_2, V_1, V_2$ .

$$V_1 = \frac{k_1}{n}, V_2 = \frac{k_2}{n} \quad (31)$$

- 5) Для клеток списка, для которых количество соседних закрытых клеток поля одинаково, рассчитываем значения  $V_{sr1}, V_{sr2}$ .

$$V_{sr1} = \frac{\sum V_1}{n_o}, V_{sr2} = \frac{\sum V_2}{n_o} \quad (32)$$

- 6) Сортируем подсписок клеток списка, для которых количество соседних закрытых клеток поля одинаково по возрастанию значения  $\min(V_{sr1}, V_{sr2})$ .

Определим теперь, какие значения необходимо добавить, чтобы реализовать математическую часть метода. Для клеток поля необходимо определить дополнительное значение, с помощью которого можно будет отличить те клетки, для которых проверяется допустимая комбинация значений (будем называть данные клетки фокусными клетками) от всех остальных клеток. После выдвижения гипотезы необходимо вычислять в соседних с

фокусной клеткой закрытых клетках, находятся ли в данных клетках мины или нет. Для этого также необходимо определить дополнительное значение. Также необходимо изменить функцию  $R$  таким образом, чтобы данная функция учитывала вычисленные значения в закрытых клетках, исходя из значений в фокусных клетках.

Опишем теперь математическую составляющую. Определим новый кортеж кортежей  $F_m$  (modified field – модифицированное поле), длина кортежа которого равна  $l$ , а длина каждого подкортежа равна  $w$  (аналогично кортежу кортежей  $F$ ). Элементами подкортежей кортежа  $F_m$  являются кортежи  $C_m$  (modified cell – модифицированная клетка), которые в свою очередь являются надкортежом кортежа  $C$ . Кортеж  $C_m$  состоит из 6-ти элементов:

$$\langle pv \in CV, b \in B, cv \in CCV, ov \in OCV, v \in CCV \cup OCV \cup OCV_p, s \in St \rangle,$$

где  $pv$  – (probability value – вероятностное значение) – элемент множества  $CV$ , которое отвечает за вычисленное значение в клетке, исходя из значений в фокусных клетках,  $b$  – (binary – двоичный) – элемент множества  $B$ , которое отвечает за то, является ли данная клетка фокусной или нет (1 – данная клетка является фокусной, 0 – данная клетка не является фокусной).

Множество  $CV$  (calculated value – вычисленное значение) состоит из значений  $NC$  (not computed – не вычислено),  $CM$  (calculated mine – рассчитана мина),  $CNM$  (calculated not mine – рассчитана не мина).

Определим несколько функций  $R_i$  (increased reflection – увеличенное отражение) и  $R_r$  (reduced reflection – уменьшенное отражение):

$$R_i = f(x, y): F \rightarrow F_m = \langle pv_1 = NC, b_1 = 0, v_1 = v_2, ov_1 = ov_2, cv_1 = cv_2, s_1 = s_2 \rangle, \quad (33)$$

$$\text{где } pv_1, b_1, v_1, ov_1, cv_1, s_1 \in F_m[x, y], v_2, ov_2, cv_2, s_2 \in F[x, y]$$

$$R_r = f(x, y): F_m \rightarrow F = \langle v_1 = v_2, ov_1 = ov_2, cv_1 = cv_2, s_1 = s_2 \rangle, \quad (34)$$

$$\text{где } v_1, ov_1, cv_1, s_1 \in F[x, y], v_2, ov_2, cv_2, s_2 \in F_m[x, y]$$

Определим кортеж кортежей  $F_{mac}$  (modified adjacent cells – модифицированные соседние клетки), длина кортежа которого равна  $l$ , а длина каждого подкортежа равна  $w$  (аналогично кортежу кортежей  $F$ ). Элементами подкортежей кортежа  $F_{mac}$  являются кортежи, элементами которого являются кортежи  $C_m$ .

Определим функцию  $AC_m$  (modified advanced cell – модифицированные соседние клетки) над множеством  $F_m$ :

$$\begin{aligned}
AC_m &= f(x, y): F_m \rightarrow F_{mac} = \\
&= \begin{cases} < F_m[x-1][y-1], F_m[x-1][y], F_m[x-1][y+1], F_m[x][y-1], \\ & F_m[x][y+1], F_m[x+1][y-1], F_m[x+1][y], F_m[x+1][y+1] >, \\ & \text{если } 0 < x < l, 0 < y < w \\ & < F_m[x][y-1], F_m[x][y+1], F_m[x+1][y-1], F_m[x+1][y], \\ & F_m[x+1][y+1] >, \text{если } x = 0, 0 < y < w \\ & < F_m[x-1][y-1], F_m[x-1][y], F_m[x-1][y+1], F_m[x][y-1], \\ & F_m[x][y+1] >, \text{если } x = l, 0 < y < w \\ & < F_m[x-1][y], F_m[x-1][y+1], F_m[x][y+1], F_m[x+1][y], \\ & F_m[x+1][y+1] >, \text{если } 0 < x < l, y = 0 \\ & < F_m[x-1][y-1], F_m[x-1][y], F_m[x][y-1], F_m[x+1][y-1], \\ & F_m[x+1][y] >, \text{если } 0 < x < l, y = w \\ & < F_m[x][y+1], F_m[x+1][y], F_m[x+1][y+1] >, \text{если } x = 0, y = 0 \\ & < F_m[x][y-1], F_m[x+1][y+1], F_m[x+1][y] >, \text{если } x = 0, y = w \\ & < F_m[x-1][y], F_m[x-1][y+1], F_m[x][y+1] >, \text{если } x = l, y = 0 \\ & < F_m[x-1][y-1], F_m[x-1][y], F_m[x][y-1] >, \text{если } x = l, y = w \end{cases} \quad (35)
\end{aligned}$$

Определим кортеж кортежей  $F_{mbac}$  (modified binary adjacent cells of field – модифицированные двоичные соседние клетки поля), длина кортежа которого равна  $l$ , а длина каждого подкортежа равна  $w$  (аналогично кортежу кортежей  $F$ ). Элементами подкортежей кортежа  $F_{mbac}$  являются кортежи  $T_b$ . Длины кортежей подкортежей кортежей  $F_{mbac}$  будем обозначать как  $n_2[x, y]$ .

Определим функцию над множеством  $F_{mac}$ :

$$\begin{aligned}
sign\_M_p &= f(x, y, z): F_{mac} \rightarrow F_{mbac} = \\
&= \begin{cases} 1, \text{если } (p \in F_{mac}[x][y][z] = PM \text{ ИЛИ } cv \in F_{mac}[x][y][z] = MF) \\ 0, \text{если } (p \in F_{ac}[x][y][z] = PNM \text{ ИЛИ} \\ (s \in F_{mac}[x][y][z] = O \text{ И } ov \in F_{mac}[x][y][z] \neq M)) \end{cases} \quad (36)
\end{aligned}$$

Определим функцию над множеством  $F_m$ :

$$\begin{aligned}
R_2 &= f(x, y): F_m \rightarrow F_m = \\
&= \begin{cases} < p = p, b = b, v = v, cv = cv, ov = \sum_{i=0}^{n[x,y]} F_{mbac}[x][y][i], s = s >, \\ & \text{если } (s \in F_m[x][y] = O \text{ И } ov \in F_m[x][y] \neq M) \\ & < p = p, b = b, v = v, cv = cv, ov = ov, s = s >, \\ & \text{если } s \in F_m[x][y] = C \end{cases} \quad (37)
\end{aligned}$$

Таким образом, вся необходимая математическая составляющая для проверки гипотез определена.



### **2.4.5 Третье правило вычисления значения в закрытой клетке**

Рассмотрим теперь второй метод, с помощью которого можно вычислить значение в закрытой клетке.

Данный метод основывается на следующей теореме: Если при одновременной проверке двух противоположных гипотез для одной выбранной закрытой клетки поля, для другой закрытой клетки поля вычисляется одинаковое значение, исходя из значения в фокусной клетки, то в данной клетке находится вычисленное значение.

Рассмотрим доказательство данной теоремы. Всего имеется 2 гипотезы для проверки значения в закрытой клетки, из которых только одна является верной. При проверке гипотезы значения в закрытых клетках вычисляются по правилам, исходя из значения в фокусной клетки. Таким образом, если гипотеза окажется верной, то и верными будут все значения в закрытых клетках, которые вычислены, исходя из значения в фокусной клетки. Исходя из этого можно сделать вывод, что если при одновременной проверке двух противоположных гипотез для одной выбранной закрытой клетки поля, для другой закрытой клетки поля вычисляется одинаковое значение, исходя из значения в фокусной клетки, то в данной клетке находится вычисленное значение, поскольку вне зависимости от того, какая из гипотез окажется верной, значение в определённой закрытой клетке при проверке обоих гипотез вычислено одинаковое.

## ВЫВОДЫ

В данной работе произведён поиск и анализ логических задач, которые подходили по заданным критериям для разработки системы с элементами самообучения. Из рассмотренного списка логических задач выбран класс казуально-логических игр, а в качестве примера игры для рассмотрения выбрана игра «Сапёр»/«Minesweeper».

Для выбранной казуально-логической игры представлены правила, описана содержательная постановка задачи исследования и математический аппарат. Также описан алгоритм, включающий в себя ряд методов поиска эффективного решения. Некоторые методы алгоритма содержат самообучающиеся элементы, что позволяет системе проходить процесс самообучения, в ходе которого вычисляются схемы, которые по завершении обучения позволят эффективно находить решения казуально-логической игры «Сапёр».

В дальнейшем планируется добавить описание нескольких методов, содержащих элементы самообучения, представить доказательство достаточности представленных методов для достижения цели игры при любых входных данных с описанными ограничениями, а также описать, разработать и протестировать программу, основанную на представленном алгоритме с элементами самообучения.

## ТЕЗАУРУС

- **Поле** – это кортеж кортежей  $F$ , длина кортежа которого равна  $l$  (length – длина), а длина каждого подкортежа равна  $w$  (width – ширина). Элементами подкортежей кортежа  $F$  являются кортежи  $C$ .

- **Клетка** – это кортеж  $C$  (cell – клетка), состоящий из 4-ёх элементов:

$$\langle cv \in CCV, ov \in OCV, v \in CCV \cup OCV, s \in St \rangle,$$

где  $s$  – (status – состояние) – элемент множества  $St$ , которое отвечает за состояние клетки – открыта она или закрыта,

$cv$  – (close value – закрытое значение) – элемент множества  $CCV$ , которое отвечает за значение, находящееся в закрытой клетке,

$ov$  (open value – открытое значение) – элемент множества  $OCV$ , которое отвечает за значение, находящееся в открытой клетке,

$v$  (value – значение) – элемент множества  $CCV \cup OCV$ , означающий значение, находящееся в клетке.

- **Множество  $CCV$**  (close cell values - значения закрытых клеток) - множество, содержащее элемент  $MF$  (mine flag - флаг мины), элемент  $Q$  (question - вопрос/сомнение), элемент  $E$  (emptiness - пустота).
- **Множество  $OCV$**  (open cell values - значения открытых клеток) - множество, содержащее целые числа  $[0; 8]$ , элемент  $M$  (mine - мина).
- **Множество  $St$**  (status – состояние) – множество статусов клетки, состоящее из двух элементов:  $C$  (close – закрыта),  $O$  (open – открыта).
- **Закрытая клетка** – клетка, значение  $s$  которой равно  $C$ .
- **Открытая клетка** – клетка, значение  $s$  которой равно  $O$ .
- **Мина** – это элемент  $M$  множества  $OCV$ . Словосочетание «в клетке находится мина» означает, что в заданной клетке значение  $ov = M$ . Словосочетание «в клетке отсутствует мина» означает, что в заданной клетке значение  $ov \neq M$ .
- **Координаты клетки** – это упорядоченная пара  $(x, y)$ , элементы которой позволяют получить клетку поля  $F[x][y]$ .
- **Соседние клетки** (для заданной клетки поля  $F[x][y]$ ) – это кортеж клеток  $F_{ac}[x][y]$ . Соседняя клетка (для заданной клетки поля  $F[x][y]$ ) – это клетка поля  $F[x_1][y_1] \in F_{ac}[x][y]$ .

- **Множество GS** (game status - статус игры) – это множество статусов игры, состоящее из 3-ёх элементов: V (victory - победа), D (defeat - поражение) и N/O (not over - игра не окончена).
- **Изолированная клетка** – это такая клетка поля, для которой все соседние клетки – закрытые.
- **Связанные клетки** – это кортеж кортежей закрытых клеток поля, в котором суммарное количество мин в клетках  $x \neq 0$ .
- **Гипотеза** – это предположение, что в выбранной закрытой клетке С поля F значение  $ov = x$ , где  $x \in H$ .
- **Множество H** (hypothesis - гипотеза) – это множество, состоящее из 2-ух элементов: M и множества  $[0; 8]$ .
- **Противоположные гипотезы** – это две гипотезы из множества гипотез H.
- **Проверка гипотезы** – подтверждение или опровержение гипотезы практическим путём.
- **Фокусная клетка** – при проверке гипотезы это закрытая клетка, для которой проверяется выполнение гипотезы.
- **Схема** – это кортеж кортежей S (scheme – схема), элементами подкортежей которого являются пары  $\{клетка; состояние\}$ , для которых выполняется логическое условие: ЕСЛИ  $\{клетка\_1; 0\}$  И  $\{клетка\_2; 0\}$  И ... И  $\{клетка\_n; 0\}$ , ТО  $\{клетка\_n+1; 1\}$ .
- **Применение схемы** (к полю) – для каждой клетки заданного подкортежа кортежа S поиск подобных клеток поля F
- **Корректная клетка** – это такая открытая клетка С поля  $F_p$ , для которой выполняется функция  $R_2$ .
- **Нейтральная клетка** – это такая открытая клетка С поля  $F_p$ , для которой пока невозможно проверить выполнение функции  $R_2$ .
- **Некорректная клетка** – это такая открытая клетка С поля  $F_p$ , для которой не выполняется функция  $R_2$ .

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Виды головоломок. Саморазвитие 2.0. URL: <http://pruslin.ru/vidy-golovolomok/> (дата обращения: 26.12.21).
2. Е. Ю. Корлякова, М. О. Корлякова. Подход к разработке самообучающегося алгоритма игры в «Сапёр». Научно-технические технологии в приборостроении и машиностроении и развитие инновационной деятельности в вузе. Материалы Всероссийской научно-технической конференции. Том 2. Изд. КФ МГТУ им. Баумана, Калуга, 2016. С. 23-24.
3. Е. Ю. Корлякова. Подход к разработке самообучающегося алгоритма игры в «Сапёр». Презентация к докладу. Калужский филиал МГТУ им. Баумана, Калуга, 2016.

## ПРИЛОЖЕНИЕ А. Правила игры «Minesweeper»/«Сапёр»

Игра имеет три уровня сложности: Новичок, Любитель и Профессионал. Уровни различаются размером игрового поля, а также общим количеством мин на поле.

На уровне Новичок размер поля равен 8x8, однако в операционной системе Windows XP он равен 9x9.

На уровне Любитель размер поля равен 16x16; на уровне Профессионал размер поля равен 30x16.

Суть игры заключается в том, что некоторые клетки поля свободны, а некоторые заминированы. Каждая клетка поля либо свободна, либо заминирована; третьего не дано.

Перед началом игры точно известно, сколько всего мин будет установлено на поле. По умолчанию, на уровне Новичок устанавливается 10 мин; на уровне Любитель — 40 мин; на уровне Профессионал — 99 мин. Это число можно изменить. Количество обнаруженных мин указывается в левом верхнем углу.

Однако при этом неизвестно, какие именно клетки заминированы. До того, как игрок не откроет клетку или не закончится игра, неизвестно, какие именно клетки свободны, а какие заминированы.

Если игрок считает, что некая клетка свободна, он может нажать её левой кнопкой мыши. При этом клетка откроется. Возможны три варианта:

1) Если открываемая клетка была заминирована, то игра показывает это. В той клетке, которая была открыта, появляется мина, залитая красным. Также при этом открываются все остальные клетки с минами: появляется рисунок мины, но уже без красной заливки. Игра немедленно заканчивается, игрок умер (проиграл).

2) Если открываемая клетка была свободна, но хотя бы одна из соседних клеток содержит мину, то в открываемой клетке появляется цифра, указывающая общее количество мин в соседних клетках. 1 — одна соседняя клетка заминирована; 2 — две соседние клетки заминированы, и так далее. Соседними в Сапёре считаются клетки, которые имеют либо общую сторону (соприкасаются по горизонтали или по вертикали), либо имеют общий угол (соприкасаются в углу по диагонали). Очевидно, что в клетке может появляться одна из цифр от 1 до 8. Каждая цифра имеет свой цвет: 1 — голубой, 2 — зелёный, 3 — красный, 4 — синий, 5 — коричневый, и так далее. Игра продолжается.

3) Если открываемая клетка была свободна и все соседние клетки тоже свободны, то все соседние клетки, как и открываемая клетка, автоматически разминируются и помечаются как свободные (окрасятся в серый цвет). Если какая-либо из разминированных клеток также будет иметь только свободных соседей, то процесс автоматического разминирования на

этом же ходу продолжится далее. И так далее, пока не обнаружатся клетки, имеющие заминированных соседей, и не появятся цифры. Игра продолжается.

Если игрок считает, что в клетке находится мина, он может нажать её правой кнопкой мыши. При этом клетка пометится как потенциально заминированная: в этой клетке появится своеобразный флажок. Кроме того, количество необнаруженных мин при этом уменьшится на единицу. Подчёркиваю, что флажок означает мнение игрока о наличии мины, но вовсе не означает фактическую мину.

Нельзя открывать (нажимать левой кнопкой) клетки, помеченные флажком. Нельзя помечать флажками разминированные клетки — то есть серые либо помеченные цветными цифрами.

Но если игрок сомневается, то он может нажать на помеченную флажком клетку правой кнопкой мыши вторично. При этом флажок поменяется на вопросительный знак. Вопросительный знак в клетке означает, что клетка имеет неопределённый статус и заслуживает внимания. Эту клетку можно оставить на время и вернуться к ней позже.

Можно нажать на клетку со знаком вопроса правой кнопкой мыши в третий раз. При этом знак вопроса исчезнет, и клетка вернётся к своему изначальному состоянию — неопределённое состояние, отсутствие изображения.

Игра продолжается до тех пор, пока каждая клетка поля не будет иметь один и только один из следующих двух статусов:

- 1) разминирована (либо серая, либо содержит цветную цифру);
- 2) правильно помечена флажком как содержащая мину.

Если игрок пройдёт всё поле — он выиграл.

Очевидно, что часто игрок в любой момент может наткнуться на мину, это означает, что он проиграл. При этом часто бывает, что одна или несколько клеток помечены игроком неверно. В этом случае игра помечает такие клетки перечёркнутой миной, что означает, что мины там не было, несмотря на то, что игрок поставил там флажок.