

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: «Разработка самообучающейся системы для решения  
логических задач»

III 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:

**27.12.21**

**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
2.1 Содержательная постановка задачи.....	16
2.2 Математическая постановка задачи .....	16
Выводы .....	18
Список использованных источников.....	19

## **Список используемых основных сокращений**

- ИНС – искусственные нейронные сети;
- ЛЗ – логическая задача.

## **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 эти методы демонстрируют высокую эффективность и точность, однако, и у методов машинного обучения есть свои недостатки. Так, например, известны случаи, когда в Китае житель смог разблокировать чужой телефон с использованием своих биометрических данных. Но, пожалуй, самая главная проблема машинного обучения, которая относится к ИНС – это невозможность определения, почему ИНС обучилась именно таким образом? Эта проблема не позволяет ИНС эффективно решать задачи, преимущественно связанные с логической обработкой данных. В связи с этим, возникает потребность в создании самообучающихся систем, не связанных с ИНС для эффективного решения задач, связанных с логической обработкой данных.

В данной работе представлены алгоритмы для построения самообучающихся систем, которые позволят решать определённый набор логических задач, произведен анализ данных алгоритмов и произведена их оценка, выбран набор заданий, которые данная самообучающаяся система будет способна решать, а также описаны цели, задачи и используемый математический аппарат.

## **Цель работы**

Целью данной работы является анализ методов построения самообучающихся систем (в том числе самостоятельно разработанных), их оценка, построение самообучающейся системы с дальнейшим тестированием системы и проверкой эффективности обучения на реальных задачах.

# **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.

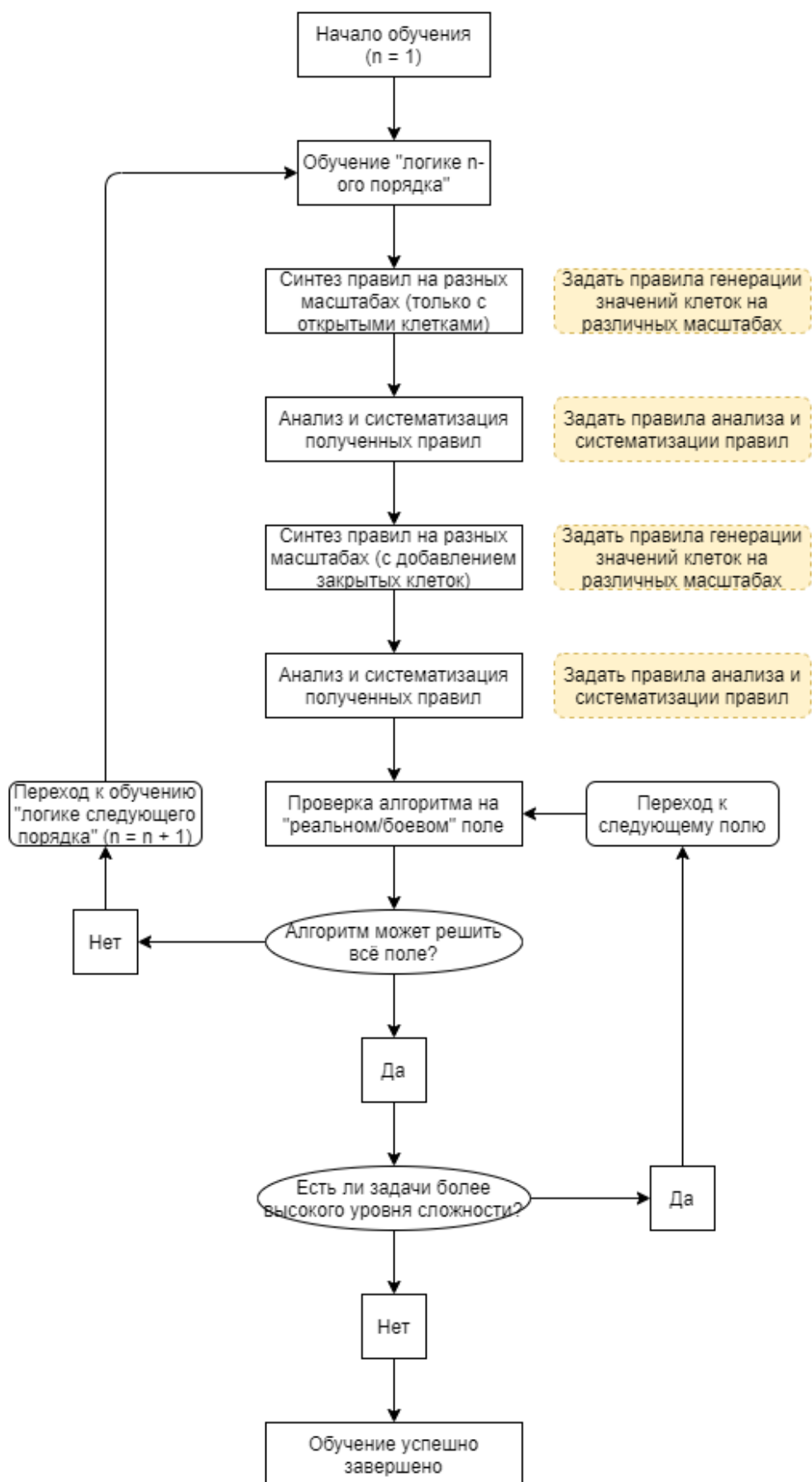


Рис. 1.Блок-схема разработанного алгоритма

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.

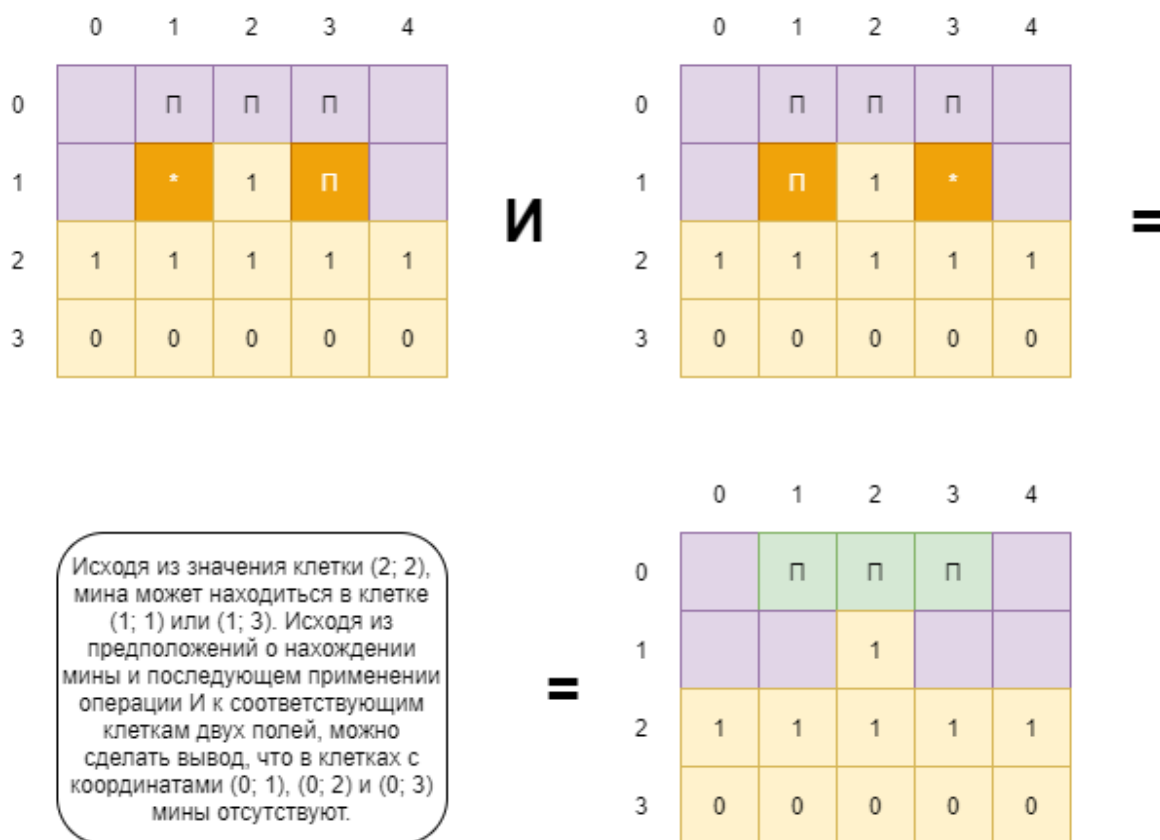


Рис. 3. Пример применения первого пункта логики 2-го порядка

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

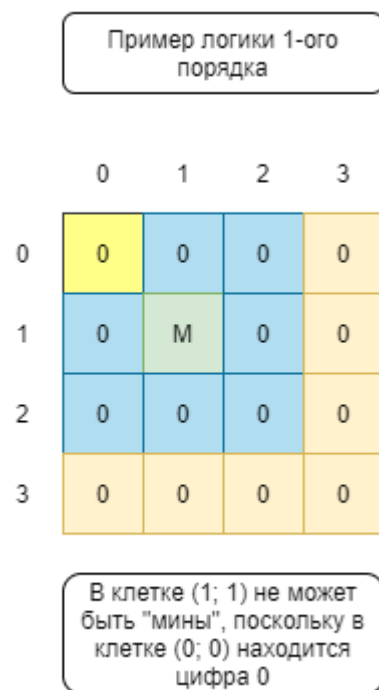


Рис. 4. Пример применения второго пункта логики 2-го порядка

### 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 – разработать и написать на одном из языков программирования самообучающийся алгоритм для решения логической задачи, основой которой является игра «Сапёр».

### 2.1 Содержательная постановка задачи

Исходными данными для решения поставленной задачи являются:

- поле прямоугольной формы, которое разбито на определённое количество клеток;
- количество клеток в каждом столбце и каждой строке одинаковое количество;
- в каждой из клеток может находиться одна из цифр  $[0; 8]$  или мина;
- каждая клетка может быть открыта или закрыта;
- в случае, если клетка закрыта, то неизвестно, что в ней находится, в противном случае – известно;
- общее количество мин, которое находится на поле;
- набор условий, определяющие правила игры, а именно:
  - цифра в клетке определяет количество мин, расположенных рядом с ней (рядом, имеется в виду, справа, снизу, слева, сверху, справа сверху, справа снизу, слева снизу и слева сверху);
  - цель данной задачи, следующая: игроку необходимо открыть все закрытые клетки таким образом, чтобы остались закрытыми только те клетки, в которых расположены мины;
  - если игрок вскрывает закрытую клетку и в ней оказывается мина, то считается, что игрок проиграл.

### 2.2 Математическая постановка задачи

Математическая постановка задачи сформулирована следующим образом: с применением самообучающейся системы посредством последовательного определения значений клеток поля, для которых выполняется неравенство  $P_{i,j}(x) \neq -1$  и получения ответной информации в виде значения координаты  $P_{i,j}(x)$  вычислить координаты клеток на поле, для которых выполняется условие:  $P_{i,j}(x) = -1$ , где



$P_{i,j}(x)$  – клетка поля с координатами  $(i, j)$ , где  $i$  – номер строки,  $j$  – номер столбца, а  $x$  – параметр, принимающий значения 0 или 1, где цифра 0 означает, что клетка закрыта, а цифра 1 означает, что клетка открыта;

$P_{i,j}(1)$  может принимать значения в диапазоне  $[0; 8]$ , а также значение -1, которое означает, что в данной клетке находится мина;

$P_{i,j}(0)$  не может принимать значений и обозначает только то, что данная клетка закрыта;

$l_b$  – количество столбцов на исходном поле;

$l_t$  – количество столбцов на изменённом поле ( $l_t = l_b + 2$ );

$w_b$  – количество строк на исходном поле;

$w_t$  – количество строк на изменённом поле ( $w_t = w_b + 2$ );

$m_t$  – общее количество мин, находящихся на поле;

$c_{bt}$  – общее количество ячеек на исходном поле ( $c_{bt} = l_b * w_b$ );

$c_{tt}$  – общее количество ячеек на изменённом поле ( $c_{tt} = l_t * w_t$ );

$c_o$  – общее количество открытых ячеек на изменённом поле;

$c_c$  – общее количество закрытых ячеек на изменённом поле ( $c_c = c_{tt} - c_o$ );

$w$  – успешно или неуспешно найдено решение поля (при  $w = 0$  задача решена неверно, при  $w = 1$  задача решена верно).

Процесс решения задачи заключается в следующем: игрок определяет такие значения  $i$  и  $j$ , при которых, по его мнению,  $P_{i,j}(x) \neq -1$  (где  $x = 0$ ) и компьютер возвращает истинное значение клетки  $P_{i,j}(x)$ . Если  $P_{i,j}(x) = -1$  процесс решения задачи автоматически останавливается и значение переменной  $w$  устанавливается в 0, в противном случае цикл повторяется до тех пор, пока не выполнится условие проигрыша (для выбранных  $i$  и  $j$  не выполнится равенство  $P_{i,j}(x) = -1$  или до тех пор, пока не выполнится условие победы ( $c_c = m_t$ )).

## **Выводы**

В данной работе был произведён поиск и анализ логических задач, которые подходили по заданным критериям для разработки самообучающейся системы. Из большого набора представленных логических задач – головоломок, была выбрана задача поиска решения в игре сапёр.

Для выбранной логической задачи был найден и описан подход к разработке самообучающегося алгоритма, а также описан собственный алгоритм для самообучающейся системы, произведён сравнительный анализ представленного подхода и алгоритма и определён более предпочтительный вариант для реализации.

Также в данной работе описана содержательная постановка задачи исследования и представлен математический аппарат, с помощью которого будет реализован самообучающийся алгоритм.

В дальнейшем планируется разработка самообучающейся системы, основанной на предложенном алгоритме, тестирование данной системы и проведение сравнительного анализа работы разработанной самообучающейся системы и других алгоритмов решения поставленной задачи, в том числе выявления достоинств и недостатков в работе программы.

В случае успеха разработки самообучающейся системы планируется модифицировать её для решения других логических задач, таких как sudoku и «Бэнг!».

### **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.