МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра прикладної математики і механіки

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«АВТОМАТИЗАЦІЯ ПРОЦЕСУ ІНДУКТИВНОГО МОДЕЛЮВАННЯ СИСТЕМ ЗІ СТАНАМИ, ЩО ЗМІНЮЮТЬСЯ»**

Виконав: студент 2 курсу, групи 8.1138

спеціальності 113 прикладна математика
(шифр і назва спеціальності)

освітньої програми прикладна математика
(назва освітньої програми)

Є. В. Сорока
(ініціали та прізвище)

Керівник доцент кафедри прикладної математики і механіки, доцент, к.ф. - м.н. Кондрат'єва Н.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент декан математичного факультету, професор, д.т.н. Гоменюк С.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра прикладної математики і механіки

Рівень вищої освіти магістр

Спеціальність 113 прикладна математика
(шифр і назва)

Освітня програма прикладна математика

ЗАТВЕРДЖУЮ

Завідувач кафедри прикладної математики і механіки, д.т.н., професор

_____ Грищак В.З.
(підпис)

« 29 » 05 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ(СТУДЕНТЦІ)

Сороці Євгену Володимировичу
(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Автоматизація процесу індуктивного моделювання систем зі станами, що змінюються

керівник роботи (проекту) Кондрат'єва Наталія Олександрівна, к.ф.-м.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » 05 2019 року № 811-с

2. Строк подання студентом роботи 27.12.19

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі
2. Загальні теоретичні відомості
3. Практична частина

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 29.05.19

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	03.09.19-11.09.19	
2.	Збір вихідних даних.	11.09.19-24.09.19	
3.	Обробка методичних та теоретичних	25.09.19-11.10.19	
	джерел.		
4.	Розробка першого розділу.	12.10.19-1.11.19	
5.	Розробка другого розділу.	2.11.19-13.11.19	
6.	Розробка третього розділу.	13.11.19-3.12.19	
7.	Практична частина	4.12.19-15.12.19	
8.	Оформлення та нормоконтроль	17.12.19-27.12.19	
	кваліфікаційної роботи.		
9.	Захист кваліфікаційної роботи.	16.01.20	

Студент _____ Є.В. Сорока
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.О. Кондрат'єва
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ В.В. Леонтьєва
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Автоматизація процесу індуктивного моделювання систем зі станами, що змінюються»: 92 с., 10 рис., 3 табл., 20 джерел, 1 додаток.

ST-СИСТЕМА, ВИХІДНА СИСТЕМА, СИСТЕМА ДАНИХ, ІНДУКТИВНЕ МОДЕЛЮВАННЯ, ЕПІСТЕМОЛОГІЧНІ РІВНІ, СИСТЕМА З ПОВЕДІНКОЮ, СИСТЕМА ЗІ СТАНАМИ, ST-ВІДНОШЕННЯ, ЧІТКИЙ КАНАЛ СПОСТЕРЕЖЕННЯ.

Об’єкт дослідження – системи зі станами, що змінюються.

Мета роботи – побудова та дослідження математичної моделі системи зі станами, що змінюються. Проведення порівняльної характеристики між системою з поведінкою та ST-системою.

Метод дослідження – аналітичний.

У кваліфікаційній роботі розкривається проблема дослідження складних об'єктів. Наводяться спроби формалізувати поняття системи з станами, які змінюються, і визначення переваг та недоліків її використання при побудові математичної моделі об’єкта дослідження по відношенню до системи з поведінкою. Для формування уявлення про сутність індуктивного моделювання розглянуті найбільш важливі визначення і правила цієї ніші системного аналізу. Наводяться альтернативні методи побудови математичної моделі системи з станами, які змінюються. Проводитися аналіз об’єкта дослідження і розробляється програмний продукт, який вирішує поставлене перед кваліфікаційною роботою питання про автоматизацію процесу моделювання. Проведення інтерпретації отриманого результату відповідно до мети дослідження.

SUMMARY

Master’s Qualification Thesis «Automation of the Process of Inductive Modeling of Systems with Varying States»: 92 pages, 10 figures, 3 tables, 20 references, 1 supplement.

ST-SYSTEM, OUTPUT SYSTEM, DATA SYSTEM, INDUCTIVE MODELING, EPISTEMOLOGICAL LEVELS, BEHAVIOR SYSTEM, STATE-RELATIONSHIP RELATIONSHIP, ST-RELATIONSHIP.

Object of study – systems with varying states.

The aim of the study is to explore remarkable lines of a triangle, learn to use them in practice.

The method of research is analytical.

Qualifying work reveals the problem of researching complex objects. Attempts are made to formalize the concept of a system with varying states, and to identify the advantages and disadvantages of its use in the construction of a mathematical model of the object of study in relation to the system with behavior. To form an idea of the essence of inductive modeling, the most important definitions and rules of this niche of system analysis are considered. Alternative methods for constructing a mathematical model of a system with varying states are presented. The object of research is analyzed and a software product is developed that solves the question of automation of the modeling process. Conducting an interpretation of the result obtained according to the purpose of the study.

ЗМІСТ

Завдання на кваліфікаційну роботу…… 2
Реферат…… 4
Summary…… 5
Вступ…… 8
1 Визначення поняття системи зі станами, що змінюються. Сучасний стан проблеми автоматизації …… 11
1.1 Опис предметної області та сучасний стан досліджуваної проблеми …… 11
1.2 Визначення системи зі станами, що змінюються …… 13
1.3 Приклади систем зі станами, що змінюються …… 17
2 Індуктивне моделювання. Методи дослідження систем зі станами, що змінюються …… 19
2.1 Емпіричне дослідження. Задача та ціль емпіричного дослідження …… 19
2.2 Імітаційне моделювання …… 22
2.3 Апаратно-програмне моделювання …… 23
2.4 Індуктивне моделювання …… 25
3 Принципи автоматизації процесу моделювання системи зі станами, що змінюються…… 40
3.1 Етапи проектування математичної моделі …… 40
3.2 Технічний підхід до моделювання системи зі станами, що змінюються…… 43
4 Реалізація автоматизації побудови математичної моделі …… 46
4.1 Постановка задачі …… 46
4.2 Визначення вихідної системи …… 49
4.3 Формування системи даних …… 56
4.4 Побудова системи зі станами, що змінюються …… 60

Висновки…….. 66
Перелік посилань……....……... 68
Додаток А Код програмного продукту «CSB++»…… 70

ВСТУП

Впровадження сучасних технологій в процес дослідження структурно складних об'єктів на сьогоднішній день вважається фактичної необхідністю. Інформаційна база про зовнішній світ неухильно зростає з кожним днем, отже дослідникам доводиться зустрічатися з проблемою її обробки вже доступними методами, адже об'єкти можуть динамічно змінюватися за певних умов. Виходячи з цього, розроблені методики можуть застаріти разом з уявленням дослідників про досліджуваний об'єкт. Так як об'єкти найчастіше ускладнюються не тільки структурно, а й кількісно, тобто кількість описуючих його поведінку властивостей так само зростає, виникає потреба у деяких загальних методологічних засобах вивчення подібних об’єктів, які б на основі наявних даних та уявлень про його взаємодію з навколишньою середою, будували узагальнену математичну модель незалежно від специфіки самого об'єкту. Важливо те, що ці методологічні засоби повинні бути технічно-орієнтовані, щоб виступати в якості бази для побудови алгоритмів автоматизації всього процесу побудови математичної моделі системи зі станами, що змінюються. Таким чином, визначається системний підхід до вирішення поставленої проблеми [1-22].

В результаті взаємодії дослідника з об’єктом дослідження (як правило, це спостереження за поведінкою об’єкта), або на теоретичному уявленні про об’єкт у випадку, коли взаємодія неможлива, визначаються деякі вихідні дані, що служать фундаментом для побудови математичної моделі системи зі станами, що змінюються[5, 9]. Об’єкт дослідження розглядається як сукупність характеризуючих його поведінку властивостей та множин їх станів і може бути описаний у вигляді складної системи, яка містить інформацію про переходи з одного стану в інший. Під час процесу моделювання виділяються функціонально-структурні особливості складної системи, з урахуванням яких здійснюється розробка технічних рішень в рамках досліджуваного об’єкта.

Актуальною є проблема дослідження впливу кількості вихідних даних на точність обчислювальних експериментів при побудові математичної моделі, адже характер цієї проблеми пов’язаний з важливістю використання обчислювальної техніки в дослідженні заданого об’єкта. Беручи до уваги, що результати обчислювальних експериментів повинні бути якнайбільш точними, що безпосередньо впливає на відповідність між об’єктом дослідження та його математичною моделлю, яка повинна бути максимально наближеною до оригіналу, ці експерименти проводяться виключно на оптимальній математичній моделі. Важливо розуміти роль системи з станами, які змінюються, в контексті індуктивного моделювання, зокрема, у визначеному ним класі породжуючих систем, адже подібна система далеко не для кожного дослідження вважається зручною для побудови більш абстрактних моделей на її основі. Таким чином, метою даної кваліфікаційної роботи метою є не тільки побудова системи зі станами, що змінюються, засобами сучасних технологій, але й визначення переваг і недоліків її використання.

Кожний розділ в теоретичній частині кваліфікаційної роботи містить необхідні для проведення дослідження відомості, які в сукупності формують потужну базу для реалізації автоматизації побудови математичної моделі.

В практичній частині розробляється програмний продукт, який автоматизує процес побудови системи зі станами, що змінюються, а отриманий результат інтерпретується у найбільш доступному для розуміння користувача вигляді.

В першому розділі приводиться характеристика складної система з наведенням реальних прикладів. Визначається предметна область дослідження та сучасний стан проблеми.

В другому розділі розглядаються принципи емпіричного дослідження, проводиться загальна класифікація методів дослідження систем зі станами, що змінюються, та визначається методологія індуктивного моделювання, яка буде закладена в основу розробленого програмного продукту.

У третьому розділі розкривається суть поставленої перед дослідженням проблеми, наводиться поетапна побудова системи зі станами, що змінюються, за допомогою програмного продукту «CSB++» (Complex System Building++). Крім цього наводиться порівняльна характеристика між системою з поведінкою і системою зі станами, що змінюються, для визначення плюсів і мінусів кожної з них.

1 ВИЗНАЧЕННЯ ПОНЯТТЯ СИСТЕМИ ЗІ СТАНАМИ, ЩО ЗМІНЮЮТЬСЯ. СУЧАСНИЙ СТАН ПРОБЛЕМИ АВТОМАТИЗАЦІЇ

У розділі розглядається сучасний стан проблеми дослідження складних систем зі станами, що змінюються, та розкривається проблематика автоматизація таких систем.

1.1 Опис предметної області та сучасний стан досліджуваної проблеми

Ключовою особливістю розвитку цілого ряду наукових напрямків, до числа яких відносяться кібернетика, загальносистемні дослідження, теорія управління, теорія систем тощо, є практично повна автоматизація процесів розв’язання системних задач, адже визначальними в них є інформаційні, реляційні та структурні аспекти, а зовсім не тип сутностей, що утворюють систему. Успішний розвиток комп'ютерної техніки призводить до регулярного нарощування обчислювальних потужностей, що дозволяє ефективно вирішувати системні задачі найвищої складності, а підвищення рівня абстракції досліджуваного об'єкта розповсюджує загальноприйняті методи системного аналізу[11] далеко за межі точних наук і охоплює більшість прикладних задач соціальної, політичної, військової та багатьох інших сфер.

Незалежно від того, в якій сфері ведеться дослідження, складність структури досліджуваного об’єкта зростає з кожним днем, отже зростає потреба або в зростанні обчислювальних потужностей для успішної обробки постійно зростаючих в обсязі даних про об'єкт, або в створенні нових методологічних коштів на технічній основі для спрощення процедури побудови узагальненої моделі дослідження[13]. Як правило для опису проявів

об’єкта за основу береться методика дослідження складних систем [17], отже вона визначає досліджувану сутність як сукупність характеризуючих її властивостей. Сутність формує об’єктну область, а описуючі характеристики представляються як знання про цю область, тобто знання, що відносяться до різних класів властивостей відносин в системі. Отримати ці знання можна або за допомогою математики, або за допомогою експериментів з моделлю системи, що реалізуються на комп’ютерній техніці. Враховуючи те, що вибірка вихідних даних, яка є попереднім представленням досліджуваного об’єкта, потенційно може бути надзвичайно масивною, автоматизація побудови математичної моделі для отримання знань об’єктної області є питанням актуальним.

Вибірка вихідних даних, що за своєю суттю, є попереднім уявленням дослідника про досліджуваний об’єкт, може бути надзвичайно масивною, тому автоматизація процесів побудови математичної моделі являє собою один з основних етапів сучасного дослідження складних систем. Крім цього, слід зазначити, що основною властивістю складної системи, окрім великого об’єму характеризуючих її поведінку проявів, є її унікальність. Тобто об’єкт дослідження не має схожих аналогів у природі, отже не існує готового алгоритму побудови його математичної моделі.

Слід зазначити, що система зі станами, що змінюються, передбачає наявність інформації про переходи досліджуваних вибіркових змінних з одного стану в інший і при аналітичному підході до побудови такої системи дослідник може одержати не фактичні дані, а лише узагальнені, адже з аналізу випадає багато динамічних аспектів поведінки об’єкта в силу того, що моделювання ведеться на основі даних обмеженого числа спостережень, які, варто зауважити, самі можуть являти собою автоматизований процес, участь дослідника в якому мінімальна, тому дані об’єкта потенційно можуть бути динамічними, а модель регулярно оновленою.

Більшість дослідників віддає перевагу саме таким системам, бо формат представлення даних дозволяє краще інтерпретувати їх візуально, наприклад,

за допомогою діаграм. Окрім цього, породжуючі та породжувані стани в системах зі станами, що змінюються, вибираються з однієї множини станів, а не з двох різних, як, наприклад, в системі з поведінкою [9], що спрощує процес побудови моделі системи. Проте існує загальний недолік для таких систем, який й приводить до необхідності автоматизації усього процесу моделювання, і полягає він у власній надлишковості через те, що поточний і наступний стани у ній накладаються. Масив даних, який необхідно обчислити для отримання фінальної оцінки якості побудованої системи, у такому випадку буде в значній мірі більшим, ніж, наприклад, при побудові системи з поведінкою.

Проектування системи завжди представляє собою процес підйому по рівням абстракцій, які її представляють, тому питання вибору типу системи є суто залежним від поставленої цілі, очікуваного кінцевого результату та обмежень, які накладаються на дослідження. Наприклад, якщо система зі станами, що змінюються, не є кінцевим результатом дослідження, та на її основі буде будуватися система більш абстрактного рівня, то в такому випадку замість системи зі станами, що змінюються, будуються системи з поведінкою.

1.2 Визначення системи зі станами, що змінюються.

Для більш змістовного розкриття системи зі станами, що змінюються, перш за все потрібно ввести поняття складної системи. Це поняття ще з початку його вживання в половині XX століття було доволі абстрактним, загальносмисловим і неконкретизованим. Відсутність строгої формалізації згодом ставала все більш загостреною проблемою, враховуючи, що обсяги оброблюваних даних для опису поведінки навіть найпримітивніших систем з часом тільки збільшувалися, а структура об'єктів дослідження починалась кардинально відрізнятися. Дослідники зіткнулися з проблемою категоризації систем. Об'єкт дослідження, складність вибору оптимальної на об'єкті вихідної системи, що найкращим чином відповідала б меті дослідження і

задовольняла поставленим обмеженням, самі обмеження, що представляють собою обмеження можливості вибору інструментів, ліміт фінансових і людських ресурсів та часу – всі ці передумови закономірно різнилися для кожного окремо взятого емпіричного дослідження. Виходячи з цього, системи, що описують об'єкт дослідження, могли відрізнятися по багатьом параметрам: за складністю множин складових підсистем, за структурою і функціонуванням, в кінцевому рахунку, за рівнем невизначеності. Виникав дисонанс в тому, до якої категорії співвідносити ту чи іншу систему, і чи вважається розглянута система складною взагалі. Це спонукало до впровадження більш строгої формалізації поняття складної системи [1-3].

У різних джерелах поняття складної системи трактується по різному. Щедровицький Г.П. [2] співставляв складність системи зі ступенем її невизначеністю, а Юдін Давид Беркович [3] заперечував те, що це поняття «складності» можна вважати скалярними. На його думку точний зміст суті складної системи є залежним від конкретних особливостей досліджуваної ситуації та поставлених проблем.

Більш сучасне трактування [4] пропонує наступне визначення: складна система – система, що складається з множини взаємодіючих підсистем, внаслідок чого набуває нових властивостей, які відсутні на підсистемному рівні і не можуть бути зведені до властивостей нижнього рівня. Таким чином поведінка системи визначається її структурою і характеристиками елементів, що її формують, а також умовами функціонування. Для систем, що не є складними, це означає, що змінити поведінку системи можна тільки змінивши її склад або структуру. Визначення зв'язку між структурою складної системи та її функціонуванням є неоднозначним.

Складні системи можна категоризувати за структурою та функціонуванням. Зустрічаються системи, складні за структурою, але прості по функціонуванню, але найбільш поширеними є системи, складність яких полягає в поєднанні і чергуванні функцій. Таким чином, складна або структура, або функціонування (набагато рідше і те, і інше).

Складною можна вважати ту систему, для якої трьох її описів (структурного, функціонального і кібернетичного) недостатньо для визначення її сутності та проведення аналізу поведінки об’єкта дослідження. Таку думку висунув дослідник конфліктуючих структур Лефевр В. А. [5]. Для повноцінного дослідження складної системи він вважав необхідним побудову інтегруючої моделі, що об'єднує всі перераховані вище описи.

Дж. Клір [9] не розрізняє прості і складні системи і не приводить конкретного визначення ні для одного з цих понять, натомість він міркує про поняття системі глобально, як про саму вершину ієрархії понять. Як доказ цього можна привести пряму цитату з його «Системології», в якій мова йде про систему в цілому: «Сила цього поняття в його абсолютній спільності». Не менш абстрактним є його пряме твердження: «Система – це те, що розрізняється як система». Опираючись на цю думку, можна зробити висновок, що визначення складної системи (як і системи в принципі) безпосередньо залежить від контексту, в якому цей термін використовується. Варто звернути увагу на те, що багато дослідників (включаючи самого Дж. Кліра) не вважали це поняття обов'язковим для визначення важливості описуючих досліджуваний предмет характеристик та складності їх взаємозв'язку.

У подальшому будемо вважати складною системою ту, для якої характерні наступні властивості:

– велике число змінних, що характеризують її поведінку;

– змінні досліджуваної системи можуть бути описані як кількісно, так і якісно;

– унікальність.

Складна система формується з даних, які були отримані за допомогою вимірів або спостережень, і містять знання про деякі інваріантні параметрам характеристики відносин змінних системи. За їх допомогою можна генерувати дані при відповідних початкових та граничних умовах, тобто опис параметрично інваріантного обмеження на розглянуті змінні

може бути використано для породження станів змінних при даній параметричній множині. Системи, які містять такі обмеження, слід називати системами, що породжують [9]. Існує форма задання параметрично інваріантного обмеження за допомогою правила зсуву. Якщо на параметричній множині властивості не визначені, то стани змінних будуть обмежуватися не тільки іншими станами, а й станами обраного сусідства для кожного значення параметра. Сусідство являє собою основу, на якій будується уявлення про параметрично інваріантне обмеження і зазвичай називається маскою. Маска визначається через змінні, параметричну множину і набір правил зсуву на цій множині. Оскільки завданням параметрично інваріантного обмеження є опис процесу, при якому стани основних змінних можуть породжуватися за багатьма параметрами за будь-яких умов, системи з поведінкою відносяться до рівня систем, що породжують. Маски в таких системах визначають множини вибіркових змінних, що породжують, та змінних, що породжуються.

Як правило, системи з поведінкою досить адекватно описують повне обмеження на вибіркові змінні, але існує й інша форма представлення такого обмеження – відношення зміни стану (надалі будемо використовувати більш короткий варіант – ST-відношення). ST-відношення визначається не на окремих станах, а на послідовних парах станів. Системи, що породжують, в яких використовується така форма, називаються системами зі станами, що змінюються, або ST-системами. Для таких систем не визначаються множини, що розділяють вибіркові змінні на ті, що породжують, і на ті, що породжуються. Ключовою особливістю таких систем є те, що вони містять характеристики переходів від одного стану до іншого та визначають рівновагу таких переходів, за допомогою чого можна враховувати в аналіз динамічні аспекти об’єкта дослідження.

1.3 Приклади систем зі станами, що змінюються

Для більш точного уявлення про систему зі станами, що змінюються, наведемо кілька прикладів з [9, 10, 13].

За допомогою ST-систем зручно ефективно описувати правові обмеження в різних законодавчих установах. За її допомогою можна, наприклад, описати законодавчу систему США з точки зору обмежень на виконання побажань громадян в новому законі. Дана ST-система буде ґрунтуватися на наступних базових змінних і їх безлічі станів:

- політичний притяг(низький; високий)
- статус виконання побажань громадян в палаті представників (відхилено; пройшло простою більшістю; пройшло переважною більшістю);
- затверджено президентом (немає, нехай);
- правовий статус (відсутня; законопроект; закон).

Параметром виступатиме час, який буде визначатися станом змінних. Система описує тільки правові обмеження, але не враховує труднощі, які потенційно можливі при переході з одного стану в інший. Дані системи можуть бути використані експертами для оцінки ступенів можливості окремих переходів, коли мова йде про конкретне виконання побажань громадян і політичних або супутніх їм обставин.

Для осмислення просторової інваріантності розглянемо чорно-білу шахівницю. Її параметрична множина складається з 64 клітин просторової решітки. Вона описується координатами x, y. На множині визначена одна змінна v. Вона має два стани: Ч (для чорного кольору) і Б (для білого). Визначимо стан $v_{x,y}$ для клітки з координатами x, y. Таким чином віборкові змінні визначатимуться рівнянням

$$s_{k,x,y} = v_{x+\alpha,y+\beta},$$

де $s_{k,x,y}$ – стан вибіркової змінної s_k для клітки з координатами x, y. Для породження станів на параметричній множині достатньо дві вибіркові змінні:

$$s_{1,x,y} = v_{x,y},$$
$$s_{2,x,y} = v_{x+1,y}.$$

Значення s_2 однозначно визначається по s_1. Початок породження в такому випадку знаходиться у верхньому лівому куті. Для кожного правила породження змінна s_1 є справочною та породжуючою. Кожна маска у поточному прикладі потребує особливі початкові умови – границі шахівниці.

За допомогою ST-системи можна описати метаболізм бактерій певного класу з біохімічної точки зору. Стани вихідних змінних представляють множину усіх субстратів, які виникають внаслідок відповідних хімічних реакцій. Стани вхідних змінних представляють коензими, які беруть участь в хімічних реакціях. Таким чином бактерії розглядаються як складні сукупності, а метаболізм як множина всіх можливих послідовностей біохімічних реакцій в бактеріях.

2 ІНДУКТИВНЕ МОДЕЛЮВАННЯ. МЕТОДИ ДОСЛІДЖЕННЯ СИСТЕМ ЗІ СТАНАМИ, ЩО ЗМІНЮЮТЬСЯ

У другому розділі розглядається ціль та задача дослідження систем зі станами, що змінюються; наводиться опис емпіричного дослідження та розглядаються найбільш поширені методи дослідження [6, 7, 9, 12, 17].

2.1 Емпіричне дослідження. Задача та ціль емпіричного дослідження

Для визначення змістовного емпіричного дослідження необхідно визначити такі передумови, як об'єкт дослідження, мета і обмеження, при яких дослідження вважається здійсненним.

Об'єкт дослідження, як правило, є частиною реального світу, цілком самодостатньою протягом тривалого часу і підходить для конкретного дослідження.

Ціллю дослідження можна вважати набір конкретних питань про об'єкт, на які дослідження з тією або іншою точністю повинно дати відповіді.

Обмеженнями в дослідженні вважаються доступні набори інструментів, фінансові ресурси, виділені тимчасові рамки, потужністю обчислювальної техніки, правові норми тощо.

Кожне емпіричне дослідження починається з формування початкового уявлення дослідника про об'єкт дослідження, що представляється як етап побудови вихідної системи на об'єкті. Основною проблемою даного етапу є вибір з багатьох можливостей вихідної системи оптимальної, тобто тої системи, яка найкращим образом відповідає цілі дослідження та поставленим обмеженням. Перед тим, як обрати найбільш підходящу для досягнення цілі дослідження систему, необхідно провести додаткове дослідження для

вивчення різних гіпотетичних варіантів систем більш високих епістемологічних рівнів.

Після вибору вихідної системи, стає можливим збір даних, який зводиться до процесу спостереження або вимірювання вибраних властивостей при деяких умовах. Усі дані ретельно записуються для побудови на їх базі складної системи. Важливою проблемою даного етапу є знаходження вихідних властивостей, які формуються шляхом управління дослідником деякими існуючими властивостями, внаслідок чого експериментальним шляхом породжуються вихідні властивості. Таки чином формується система даних.

Коли система даних сформована, починається наступний крок – обробка даних. Як результат визначаються деякі параметрично інваріантні властивості змінних, які в значній мірі дозволяють економно представити дані. Такі властивості описують обмеження, яке накладається на змінні вихідної системи, яке не змінюється на параметричній множині.

Після формування параметрично інваріантних властивостей, необхідно належним чином інтерпретувати дані з врахуванням цілі дослідження. В певній мірі, на поточному етапі визначається наскільки корисні знайдені дані для пошуку відповідей на питання дослідження. Якщо відповіді задовольняють дослідника, тоді дослідження вважається закінченим.

Система, отримана в результаті обробки даних, як правило, має або дуже складну структуру, або вміщує великий масив інформації. Тобто, система занадто складна для того, щоб дослідник міг охопити її цілком, а тому не може допомогти у досягненні цілі дослідження. В такому випадку актуальною стає проблема спрощення складності системи, адже користувач потрібен мати засоби для проведення цього процесу з врахуванням можливості управління критеріями, на яких цей процес і базується.

Після обробки даних та інтерпретації отриманих властивостей може виникнути потреба у проведенні додаткових експериментів для підтвердження зібраних даних, або для пошуку нових. Часто постає задача перегляду

властивостей на основі нових даних, якщо структура об'єкта змінюється, або потрібно провести дослідження для більшого числа вхідних змінних, щоб розширити уявлення про досліджуваний об'єкт. Повторний збір даних змінює систему даних, але може не вихідну систему.

Вся процедура емпіричного дослідження зводиться до того, що модель системи зі станами, що змінюються, повинна мати можливість:

– виводу з заданих даних параметрично інваріантних властивостей;

– порівняння виведених властивостей та виключення тих систем, властивості яких не задовольняють критеріям користувача;

– зручну для користувача інтерпретацію даних;

– можливість багаторазового використання.

Для дослідження функціонування систем зі станами, що змінюються, найбільш оптимальним вважається моделювання їх роботи на ЕОМ, що допомагає суттєво скоротити час і кошти на розробку.

2.2 Імітаційне моделювання

Реалізація методу імітаційного моделювання полягає в розробці спеціального моделюючого алгоритму [22], в основі якого реалізуються підходи та методи системології [11]. Відповідно до моделюючого алгоритму в ЕОМ виробляється інформація, що описує елементарні процеси досліджуваної системи з урахуванням взаємозв'язків і взаємовпливів виділених змінних для подальшого дослідження процесу функціонування системи зі станами, що змінюються. Алгоритм будується відповідно до логічної структури системи зі збереженням послідовності протікаючих у ній процесів і відображенням основних станів системи.

З основних етапів проектування системи відповідно до методу імітаційного моделювання згідно з [17] можна виділити наступні:

– відокремлення загальних характеристик, що описують об’єкт дослідження відповідно до зазначеної цілі;

– формування вихідної системи;

– формування системи даних;

– побудова множини математичних моделей об’єкту дослідження;

– визначення оптимальних математичних моделей;

– інтерпретація отриманих результатів моделювання;

– аналіз отриманих результатів відповідно до цілі дослідження та накладених обмежень.

Приведений алгоритм може повторюватися для різних наборів вхідних і вихідних змінних, утворюючи автоматизований цикл моделювання. У зовнішньому циклі організовується перегляд заданих варіантів станів змінних модельованої системи. Процедура вибору оптимального варіанту управляє переглядом варіантів, вносячи відповідні корективи в імітаційну модель і в моделі вхідних і вихідних змінних.

Залежно від фактичних результатів моделювання, процедура побудови системи зі станами, що змінюються, контролю точності та коригування структури самої моделі змінює внутрішній цикл. Таким чином, виникає зовнішній цикл, що відображає діяльність дослідника.

За допомогою налагодженого процесу контролю і управління побудованої за допомогою моделюючого алгоритму моделі, можна вирішувати задачі виключної складності. Система може одночасно містити елементи безперервної і дискретної дії, бути схильною до впливу численних випадкових чинників будь-якої природи, описуватися досить громіздкими співвідношеннями тощо. Метод імітаційного моделювання не вимагає створення спеціальної апаратури для кожної нової задачі і дозволяє легко змінювати значення параметрів досліджуваних систем і початкових умов. Ефективність методу імітаційного моделювання залежить від етапу, на якому він починає використовуватись.

Для побудови математичної моделі необхідно на створений моделюючий алгоритм накласти реалізацію за допомогою мови програмування. Для таких цілей, як правило, використовуються мови вищого рівня (C++, Java, Python), або потужні нижчого рівня (C, C++, Assembler).

Перевага імітаційного моделювання в порівнянні з натурним експериментом полягає в незначних витратах робочого часу та матеріальних ресурсів. Варто зазначити, що результати такого моделювання за своєю цінністю відносно практичного застосування близькі до результатів натурного експерименту.

2.3 Апаратно-програмне моделювання

Існують системи зі станами, що змінюються, рівень складності яких потребує для обробки та аналізу отриманих властивостей дуже потужні ЕОМ. Для таких цілей використання загальнодоступних технічних пристроїв та обчислювальної техніки не вважається оптимальним рішенням. Тому для самих систем впроваджують спеціальні ЕОМ, вся обчислювальна потужність яких направлена на досягнення поставленої цілі дослідження. Найбільш поширеним цей метод є для радіоелектронних систем. Серед них різні автоматичні системи, в тому числі системи автоматичної комутації, системи радіозв'язку, системи радіолокації і радіонавігації, системи управління.

Характерною особливістю таких ЕОМ є наявність в системах декількох процесорів, об'єднаних різними способами в спеціалізовану ОС. При цьому здійснюється перехід від жорсткої логіки функціонування технічних систем до універсальної програмної логіки. Головну роль в таких системах грає спеціалізоване системне і прикладне програмне забезпечення.

На етапах розробки, проектування, налагодження та випробування складних систем з досить високою потребою до апаратно-програмних засобів обчислювальної техніки ставиться задача аналізу і синтезу варіантів

організації структури апаратних засобів, а також розробки і налагодження спеціалізованого програмного забезпечення великого обсягу. Це завдання може бути вирішене за допомогою апаратно-програмного моделювання з використанням універсальних моделюючих комплексів, побудованих на базі однорідних ОС з програмованою структурою.

Апаратно-програмне моделювання можна вважати окремим випадком напівнатурного моделювання. На першому етапі розробляється концептуальна модель заданого класу систем на основі аналізу типових процесів, структур і апаратних блоків. Концептуальна модель реалізується на апаратно-програмних засобах моделюючого комплексу. При цьому комплекс може налаштовуватися на відповідну структуру системи програмним шляхом за рахунок можливості програмування структури використовуваної мікропроцесорної ОС. Частина апаратних і програмних засобів мікропроцесорної ОС моделюючого комплексу безпосередньо відображає апаратно-програмні засоби, що входять до досліджуванної системи (апаратне моделювання), інша частина реалізує імітаційну модель функціональних засобів досліджуваної системи, зовнішньої обстановки, впливу перешкод і т.п. (програмне моделювання).

Розробка апаратно-програмних моделюючих комплексів є складним технічним завданням. Незважаючи на це, застосування таких комплексів знаходить все більшого поширення. При достатній продуктивності обчислювальних засобів комплексу процес дослідження системи може вестися у реальному часі та у повному масштабі. У складі комплексу можуть використовуватися як універсальні мікроЕОМ загального призначення, так і обчислювальні засоби, які безпосередньо входять в досліджувану систему. Подібні змодельовані комплекси є універсальними стендами для розробки і налагодження апаратно-програмних засобів, проектуючих систем заданого класу [6].

2.4 Індуктивне моделювання

Дж. Клір [9] називає індуктивним моделюванням важливий клас системних задач, пов’язаних з підйомом по епістемологічній ієрархії систем. Епістемологічна ієрархія систем представляє собою чітке розподілення визначених класів систем згідно із епістемологічною класифікацією задач. Дана ієрархія спирається на кілька елементарних понять: дослідник (спостерігач) і його середовище, досліджуваний (спостережуваний) об'єкт і його середовище і взаємодія між дослідником і об'єктом.

Найнижчий рівень в цій ієрархії позначається як нульовий рівень. На даному рівні дослідник вибирає спосіб, яким він хоче взаємодіяти з досліджуваним об'єктом. У більшості випадків цей вибір не цілком довільний. Принаймні частково він визначається метою дослідження, умовами дослідження, а також наявними знаннями, які відносяться до даного дослідження. Як правило, система нульового епістемологічного рівня визначена через множину змінних, множину потенційних станів (значень), що виділяються для кожної змінної, і якийсь операційний спосіб смислового опису цих станів в термінах проявлень відповідних атрибутів даного об'єкта.

При цьому, сам об'єкт сприймається як сукупність характеризуючих його властивостей. Для певних на цьому рівні систем використовується термін вихідна система, що вказує на те, що подібна система є, принаймні потенційно, джерелом емпіричних даних. При цьому системи, в яких змінні розділені на вхідні і вихідні, називаються спрямованим; системи, в яких такий поділ не задано, називаються нейтральними. Формально система нульового епістемологічного рівня визначається за формулою (2.1):

$$S = (O, \dot{I}, I, Q, \varepsilon), \tag{2.1}$$

де O – система об’єкту, $\dot{I}$ – конкретна представляюча система, I– узагальнена представляюча система, Q – чіткий канал спостережень, ε – канал абстрагування.

Система об’єкту O включає суттєві властивості і базові властивості (базу) та представляється у виді:

$$O = \{(a_i, A_i \mid i \in N_n), (b_j, B_j \mid j \in N_m)\}. \quad (2.2)$$

Канал спостережень Q характеризує перехід від системи об’єкта O до конкретної представляючої системи $\dot{I}$. та визначається по формулі (2.3):

$$Q = \{(A_i, \dot{V}_i, o_i \mid i \in N_n), (B_j, \dot{W}_j, s_j \mid j \in N_m)\},$$

$$o_i: A_i \to \dot{V}_i, s_j: B_j \to \dot{W}_j. \quad (2.3)$$

Конкретна представляююча система $\dot{I}$ описує об'єкт в термінах каналів спостереження всіх змінних і множин їх значень. Її математична інтерпретація представлена у вигляді (2.4):

$$\dot{I} = \{(\dot{v}_i, \dot{V}_i \mid i \in N_n), (\dot{w}_j, \dot{W}_j \mid j \in N_m)\}. \quad (2.4)$$

Узагальнена представляюча система I задає об'єкт в абстрактному вигляді введенням узагальнених шкал вимірювання значень змінних і параметрів. Для того, щоб ввести компоненти цієї системи використовується канал абстрагування, який характеризує перехід від $\dot{I}$ до I. Канал абстрагування визначається наступним чином:

$$\varepsilon = \{(\dot{V}_i, V_i, \gamma_i \mid i \in N_n), (\dot{W}_j, W_j, \omega_j \mid j \in N_m)\},$$

$$\gamma_i : \dot{V}_i \to V_i \,, \omega_j : \dot{W}_j \to W_j. \quad (2.5)$$

Сформувавши канал абстрагування, можна визначити узагальнену представляючу систему I, використовуючи формулу (2.6):

$$I = \{v_i, V_i \mid i \in N_n), (w_j, W_j \mid j \in N_m)\}. \quad (2.6)$$

На більш високих епістемологічних рівнях системи відрізняються одна від одної рівнем знань відносно змінних відповідної вихідної системи. У системах більш високого рівня використовуються всі знання відповідних систем нижчих рівнів і, крім того, містяться додаткові знання, недоступні нижчим рівнями. Таким чином, вихідна система міститься у всіх системах більш високих рівнів.

Після того як вихідна система доповнена даними, тобто дійсними станами основних змінних при певному наборі параметрів, ми розглядаємо нову систему (вихідну систему з даними), визначену на епістемологічному рівні 1. Системи цього рівня називаються системами даних. Дані можуть бути отримані зі спостережень за допомогою вимірювань [24] або в результаті вибору будь-яких бажаних потенційних станів. Тип систем даних визначається методологічними відзнаками, які у свою чергу визначені на нульовому епістемологічному рівні. Серед основних типів систем даних можна визначити наступні: D – система даних з чіткими даними, ${}^{S}D$ – система даних з семантикою, $\widetilde{D}$ – нейтральна система даних з нечіткими даними, $\widehat{\widetilde{D}}$ – направлена система даних з нечіткими даними, ${}^{S}\widehat{D}$ – спрямована система даних з семантикою і чіткими даними, ${}^{S}\widetilde{D}$ – нейтральна система даних з семантикою і нечіткими даними.

У другому етапі практичної частини необхідно побудувати систему даних, опираючись на отримані у попередньому етапі емпіричні дані. Так як будуватимуться системи даних лише двох типів – нейтральна система даних з чіткими даними та нейтральна з семантикою і чіткими даними, – то визначимо лише їх.

$$D = (I, d);\ \ d: W \to V. \tag{2.7}$$

Формула (2.7) визначає систему даних з чіткими даними без семантики, а (2.8) – систему даних з семантикою.

$$^{S}D = (S, d);\ \ d: W \to V, \tag{2.8}$$

де функція d зв’язана з системою S за допомогою комбінацій:

$$(o_i \circ \gamma_i{}^{-1}),$$
$$(s_j \circ \omega_j{}^{-1})$$

для усіх $i \in N_n$ та $j \in N_m$.

Більш високі епістемологічні рівні містять знання про деякі інваріантні параметри характеристик відношень розглянутих змінних, за допомогою яких можна генерувати дані при відповідних початкових або граничних умовах.

Генеровані дані можуть бути детермінованими або стохастичними (нечіткими, недетермінованими).

Оскільки системи рівня 2 параметрично інваріантно описують обмеження на змінні представляючої системи та, крім цього, мають опис процесу породження даних з використанням зазначеного обмеження, то на даному етапі побудови математичної моделі складної системи вони мають назву породжуючих систем з поведінкою [9, 13]. Системи другого

епістемологічного рівня повинні бути сумісними з системами нижчих рівнів та визначаються таким чином:

$$F_b = (I, M, f_b), \tag{2.9}$$

де M – маска, причому $M \subset V \times R$ – відношення сусідства на W, R – множина правил зрушення, $r(w) \in R$ – правило зрушення, яке визначається як:

$$r(w): W \to W.$$

Так як W – цілком упорядкована множина, то $r(w) = w + q$, де q – це константа зсуву.

I – узагальнена представляюча система, $f_b: C \to \{\mathbf{0}, \mathbf{1}\}$ – функція поведінки, де $C = \times_{k=\overline{1,|M|}} S_k$ – множина станів вибіркових змінних S_k, які визначаються: $S_{k,w} = V_{i,r(w)}$.

Для визначення ідентифікатора k вибірковій змінній $S_{k,w}$ застосовується функція кодування $\rho: M \to N_{|M|}$, де $N_{|M|}$ – потужність маски M.

Інакше кажучи, породжуючими системами можна вважати такі системи, які мають параметрично інваріантні обмеження, описані на змінні таким чином, що цей опис може бути використано для породження станів змінних при даній параметричній множині. Поведінка являє собою одну з форм завдання цього обмеження і саме систему з поведінкою було визначено по формулі (2.9).

Для заданої узагальненої представляючої системи діапазон можливих типів параметрично інваріантних обмежень залежить від властивостей, приписуваних параметричній множині. Так як ця множина ніяких властивостей не містить (як це часто буває для груп), то стани змінних можуть обмежувати тільки один одного. Однак якщо параметрична множина

впорядкована, стани змінних можуть обмежуватися не тільки іншими станами, а й станами обраного сусідства для кожного конкретного значення параметра.

Сусідство на упорядкованій параметричній множині зазвичай називається маскою.

Не зважаючи на те, що система з поведінкою, яка визначається за формулою (2.9), параметрично інваріантно описує обмеження на зміні представляючої системи, вона не містить опису того, як використати це обмеження для породження даних. Для цього необхідно розбити вибіркові зміні на дві підмножини: породжуючі зміні, стани котрих породжуються з обмеження та породжуючі зміні – зміні, стани яких використовуються як умови у процесі генерації. Для системи з поведінкою (2.9) визначимо дві підмаски M_g та $M_{\bar{g}}$ маски M та отримаємо множину:

$$M_G = \left(M;\ M_g, M_{\bar{g}}\right), \tag{2.10}$$

для якої справедливі наступні умови:

$$M_g, M_{\bar{g}} \in M,\ M_g \cup M_{\bar{g}} = M,\ M_g \cap M_{\bar{g}} = \emptyset.$$

Формула (2.10) визначає маску породження, тобто розбиття маски M на породжувану підмаску M_g і породжуючу підмаску $M_{\bar{g}}$. Аналогічно розбивається множина ідентифікаторів $N_{|M|}$ на підмножини K_g та $K_{\bar{g}}$, які представляють ідентифікатори породжуваних та породжуючих змінних відповідно. Звідси функція кодування матиме наступний вигляд:

$$\lambda_g : M_g \to K_g,$$

$$\lambda_{\bar{g}} : M_{\bar{g}} \to K_{\bar{g}}, \tag{2.11}$$

за допомогою яких множини станів відповідно породжуваних та породжуючих змінних задаються:

$$G = \times_{k \in K_g} S_k,$$

$$\bar{G} = \times_{k \in K_{\bar{g}}} S_k. \tag{2.12}$$

Тоді спосіб представлення станів породжуваних змінних, які визначаються по станам породжуючих змінних, виражаються через породжуючу функцію поведінки:

$$f_{GB}: \bar{G} \times G \rightarrow \{0,1\}. \tag{2.13}$$

Враховуючи (2.10) – (2.13), отримаємо породжуючу систему з поведінкою:

$$F_{GB} = (I, M_G, f_{GB}). \tag{2.14}$$

Для описання направлених систем з поведінкою необхідно ввести розбиття вибіркових змінних на дві підмножини: вхідні змінні – змінні, які представленні середою та інші вибіркові змінні, пов'язані з даною маскою. Ці дві підмножини можна визначити, розбивши задану маску M на підмаски M_e та $M_{\bar{e}}$, які включають відповідно вхідні та вихідні змінні. Таким чином, маска для направленої системи з поведінкою матиме вигляд:

$$\widehat{M} = (M;\ M_e, M_{\bar{e}}). \tag{2.15}$$

Розбиття маски M на $M_e, M_{\bar{e}}$ виконується відносно заданого визначника входу/виходу системи та повинно задовольняти наступні умови:

$$M_e, M_{\bar{e}} \in M,\ M_e \cup M_{\bar{e}} = M \text{ та } M_e \cap M_{\bar{e}} = \emptyset.$$

Множина ідентифікаторів $N_{|M|}$ розбивається на підмножини K_e та $K_{\bar{e}}$. Звідси функції кодування матимуть наступний вигляд:

$$\lambda_e : M_e \to K_e,$$
$$\lambda_{\bar{e}} : M_{\bar{e}} \to K_{\bar{e}}, \tag{2.16}$$

та визначатимуть наступні підмножини станів:

$$E = \times_{k \in K_e} S_k,$$
$$\bar{E} = \times_{k \in K_{\bar{e}}} S_k, \tag{2.17}$$

які необхідні для направлених систем. Функція поведінки тоді матиме вигляд:

$$\hat{f}_B : E \times \bar{E} \to \{\mathbf{0}, \mathbf{1}\}. \tag{2.18}$$

Враховуючи (2.15) – (2.18), можна описати направлену систему з поведінкою наступним чином:

$$\hat{F}_B = \left(\hat{I}, \hat{M}, \hat{\boldsymbol{f}}_{\boldsymbol{B}}\right). \tag{2.19}$$

Направлена породжуюча система з поведінкою визначається:

$$\hat{F}_{GB} = \left(\hat{I}, \widehat{M}_G, \hat{\boldsymbol{f}}_{\boldsymbol{GB}}\right), \tag{2.20}$$

де $\widehat{M}_G$ – породжуюча маска для направлених систем з поведінкою та визначається:

$$\widehat{M}_G = \left(M; M_e, M_g, M_{\bar{g}}\right), \tag{2.21}$$

причому $\{M_e, M_g, M_{\bar{g}}\}$ – розбиття M; $\hat{f}_{GB}$ – породжуюча функція для направлених систем з поведінкою, яка визначається наступним чином:

$$\hat{f}_{GB}: E \times G \times \bar{G} \rightarrow \{\mathbf{0}, \mathbf{1}\}. \tag{2.22}$$

Відомо, що побудована породжуюча система з поведінкою є ймовірнісною математичною моделлю досліджуваної складної системи [9].

Для того, щоб визначити оптимальну математичну модель, необхідно побудувати множину Y_r систем з поведінкою, які представляють задану систему даних D та мають деякі підходящі додаткові властивості. Множина таких систем Y_r є підмножиною множини Y усіх систем з ймовірнісними функціями поведінки, які сумісні з системою даних. Причому підмножина Y_r повинна задовольняти наступним умовам:

а) підмножина Y_r множини Y визначений в автоматизованій системі як вибір за замовчуванням;

б) неузгодженість між відповідними змінними заданої системи даних та системою з поведінкою з множини Y_r повинна бути мінімальною;

в) степінь невизначеності при породженні даних системи з поведінкою з множини Y_r повинен бути незначним;

г) система з Y_r повинна бути якнайбільш простою;

д) перевага умови б) умовам в), г).

Системи з поведінкою, які задовольняють цим умовам (ці умови прийнято називати потребою відповідності [9]), однозначно визначаються своїми масками. Зведення множини Y до підмножини Y_r відповідно до умови а) можна представити як обмеження на множину допустимих масок. Для цього необхідно визначити маску найбільшу допустиму маску, яка обмежує глибину маски (кількість стовбців в масці M). Подібна маска матиме вигляд повної матриці з $\boldsymbol{n}$ кількістю строк та визначеною за формулою (2.23) кількістю стовбців. Глибина маски M визначається наступним чином:

$$\Delta M = 1 + max\,\rho - min\,\rho. \qquad (2.23)$$

Якщо найбільша допустима маска M задана, то, використовуючи змістовні підмаски маски M, стає можливим формування множини $\boldsymbol{Y_r}$ систем з поведінкою. Змістовна підмаска ${}^{i}M, i = \overline{2, N(n, \Delta M)}$ задовольняє умовам:

– в підмаску ${}^{i}M, i = \overline{2, N(n, \Delta M)}$ входить хоча б один елемент із строки маски ${}^{1}M$;

– в підмаску повинен бути включеним хоча б один елемент з правилом зсуву $w + \rho_2$ (крайній правий елемент).

Вказані вимоги до побудови змістовних підмасок ${}^{i}M$ необхідні для покриття заданої системи даних, тобто для того, щоб включити усі базові змінні в будь-яку систему з поведінкою множини Y_r та для передбачення і створення перешкоди дублюванню еквівалентних підмасок.

Таким чином, визначаються змістовні підмаски ${}^{i}M, i = \overline{2, N(n, \Delta M)}$ кількість яких дорівнюватиме:

$$N(n, \Delta M) = (2^{\Delta M} - 1)^n - (2^{\Delta M - 1} - 1)^n, \qquad (2.24)$$

де n – кількість базових змінних, а ΔM – глибина маски M.

Відповідно до змістовних підмасок формується множина Y_r систем з поведінкою, серед яких за степенем недетермінованості знаходиться найбільш оптимальна.

Знаходження оптимальної системи з поведінкою може проводитись за двома критеріями: впорядкування систем з поведінкою за складністю та впорядкування систем з поведінкою по нечіткості. В практичній частині необхідно визначити оптимальну математичну модель за вказаними критеріями. Цей степінь вимірюється узагальненою нечіткістю, яка супроводжує породження даних. Отже, він повинен бути визначеним через породжуючу функцію поведінки $\hat{f}_{GB}$ для направленої системи з поведінкою. Якщо функція поведінки $\hat{f}_{GB}$ представляє собою функцію розподілу ймовірностей, то міра нечіткості визначається як шенноновська ентропія [19]:

$$H(f(x) \mid x \in X) = -a \sum_{x \in X} f_B(x) log_b f_B(x). \quad (2.25)$$

Для ST-систем маски вибіркові змінні, множини станів вибіркових змінних і їх декартовій добуток C визначаються так само, як і для систем з поведінкою, за винятком наступних відмінностей:

– до ST-систем не застосовується поділ вибіркових змінних на ті, що породжуються, і ті, що породжують;

– змістовні маски в ST-системах мають додаткові обмеження. Аналогами функцій поведінки в ST-системах є функції зміни стану (або ST-функції). Для нейтральних систем вони визначені на $C^2 = C \times C$, а не на C, а для спрямованих систем на $E^2 \times \overline{E^2}$, а не на $E \times E$.

Для подальшого розгляду систем зі станами, що змінюються, визначимо наступні ST-функції:

$$f_S: C^2 \to [0, 1], \quad (2.26)$$

де $f(c, c')$ – це ймовірність стану c'наступного безпосередньо за станом c (відповідно до обраного порядку породження);

$$f_{GS}: C^2 \to [0, 1], \tag{2.27}$$

де $f_{GS}(c, c')$ – умовна ймовірність того, що при поточному стані з наступним станом буде стан c'; по цій причині будемо використовувати загальний запис $f_{GS}(c'|c)$:

$$f_{GS}: C \to C, \tag{2.28}$$

де $f_{GS}(c) = c'$, тобто наступний стан однозначно визначається поточним станом c; функція спеціального виду (2.28) може бути використана тільки до детермінованих систем. ST-функціями, що породжують, називаються функції виду f_{GS}.

ST-система визначається за формулою (2.29):

$$F_S = (I, M, f_S), \tag{2.29}$$

де I, M визначаються так само, як і для (2.9).

Формула (2.30) визначає ST-систему, що породжує:

$$F_{GS} = (I, M_G, f_{GS}), \tag{2.30}$$

де M_G визначено за формулою (2.10).

Для заданих системи даних і маски ST-функція f_S може бути визначена за допомогою повної вибірки даних аналогічно тому, як це робилося для функції поведінки f_B для визначення породжуючих систем (2.9). Єдина відмінність полягає в тому, що в результаті вибірки виходять частоти $N(c, c')$ пар послідовних станів, а не частоти $N(c)$ окремих станів.

Пара $(c, c') \in C^2$ називається переходом зі стану c в інший стан c' відносно заданого на параметричній множині порядку породження. Однією з найважливіших властивостей ST-функцій є те, що переходи в деякий стан повинні перебувати в рівновазі з переходами з цього стану. Якщо використовуються ймовірності, то для будь-якого стану $x \in C$ маємо

$$\sum_{c \in C} f_S(c, x) = f_B(x),$$

$$\sum_{c' \in C} f_S(x, c') = f_B(x), \tag{2.31}$$

отже,

$$\sum_{c \in C} f_S(c, x) = \sum_{c' \in C} f_S(x, c'), \tag{2.32}$$

що і визначає рівновагу переходів.

Стани c, c' можуть розглядатися як стани, що визначаються двома взаємопов'язаними масками M, M'. Маски пов'язані між собою простим зрушенням у відповідності з наступними правилами зсуву:

$(v_i, \rho) \in M$ тоді і тільки тоді, коли $(v_i, \rho + 1) \in M'$ (2.33)

якщо дані породжуються в порядку зростання параметра, або

$$(v_i, \rho) \in M \text{ тоді і тільки тоді, коли } (v_i, \rho - 1) \in M' \qquad (2.34)$$

якщо дані породжуються в зворотному порядку.

Маски M, M' використовуються разом для опису пар станів c, c'.

Для точного і безпомилкового породження даних, змістовні маски в ST-системах повинні задовольняти умові: для заданої маски M, якщо $(v_i, \rho_1) \in M$ та $(v_i, \rho_2) \in M$ та $\rho_1 < \rho_2$, то $(v_i, \rho) \in M$ для усіх цілих ρ, таких, що $\rho_1 \leq \rho \leq \rho_2$.

У цьому розумінні маски в ST-системах мусять бути без так званих «пробілів». Маски, що задовольняють вищенаведеній умові, називаються компактними масками.

Для обґрунтування цієї вимоги припустимо, що маска M ST-системи некомпактна. Тоді існує принаймні одна пара елементів з маски M, скажімо пара $(v_i, \rho_1) \in M$, $(v_i, \rho_2) \in M$, така, що $\rho_1 < \rho_2$,

$$(v_i, \rho_1 + 1) \notin M, (v_i, \rho_2 + 1) \notin M, \qquad (2.35)$$

та $\rho_2 \geq \rho$ для усіх $(v_i, \rho) \in M$. З (2.33) маємо

$$(v_i, \rho_1 + 1) \in M' \text{ та } (v_i, \rho_2 + 1) \in M'. \qquad (2.36)$$

Через стани s_1 та s_2, що являються компонентами c', визначимо вибіркові змінні, які базуються на елементах M'. Стани повинні бути визначенні для кожного значення по стану c, або породжуватися відповідно до

розподілу ймовірностей або можливостей $f_{GS}(c|c')$ для кожного $\boldsymbol{c}$**.** Однак жоден з цих варіантів неможливий для s_1. Згідно з (2.35) s_1 не може бути визначене по стану c. Враховуючи те, що $s_{1,t} = s_{2,t+\rho_1-\rho_2}$, воно не може коректно породжуватись при будь-якому значенні параметра t, отже s_1 визначається станом s_2 при значенні параметра $t-(\rho_2-\rho_1)$. Якщо s_1 породжується, тоді c' становиться неповним, так як раніше визначений стан не є компонентом c, отже невідомий. Враховуючи усе вищесказане, можна впевнено утверджувати, що при будь-якому значенні t стан s_1 не визначається з $\boldsymbol{c}$, та не породжується за допомогою породжуючої ST-функції.

Загалом, у роботах Дж. Кліра [9] центральне місце відводиться описанню систем з поведінкою, якими через процедуру абстрагування між вищезазначеними епістемологічними рівнями представлення даних моделюються системи реального світу з ціллю передбачення їх розвитку, виділення і управління факторами, що впливають на нього. Актуальність, наукова та практична цінність запропонованої Дж. Кліром системної методології полягає в тому, що методи системології можуть використовуватися в підтримці прийняття рішень в різних предметних областях. Увесь процес підйому по епістомологічним рівням системи розуміється у контексті деякого методологічного алгоритму опису, математичного моделювання та дослідження складної системи [9, 11]. Цей алгоритм базується на точній системі абстракцій в результаті створення структури загальних системних понять, актуальних для різних додатків.

Викладена методика може застосовуватися для вивчення об’єктів різної природи та структурної складності.

3 ПРИНЦИПИ АВТОМАТИЗАЦІЇ ПРОЦЕСУ МОДЕЛЮВАННЯ СИСТЕМИ ЗІ СТАНАМИ, ЩО ЗМІНЮЮТЬСЯ

В третьому розділі розглядаються етапи проектування та побудови математичної моделі системи зі станами, що змінюються, на ЕОМ та приводиться технічний опис програмного продукту «CSB++», який був розроблений для вирішення проблеми автоматизації усього процесу моделювання системи.

3.1 Етапи проектування математичної моделі

Загальний процес побудови моделі системи зі станами, що змінюються, складається з трьох основних етапів:

а) складання математичної моделі;

б) побудова моделюючого алгоритму;

в) розробка програмного продукту, що реалізує алгоритм.

Кожен з етапів при необхідності можна розбити на кілька складових кроків для більш докладної оцінки або обговорення. У даній кваліфікаційній роботі зупинимося на розгляді тільки основних.

Перший етап полягає в складанні математичної моделі об'єкта дослідження для того, щоб за допомогою математичних формул описати найбільш підходящі для цілі дослідження властивості даного об'єкта. Позначаються ті змінні, які необхідні для поточного дослідження, формується теоретична основа для формування множини станів об'єкта. Математична модель досліджується, коригується або відхиляється, якщо дослідник знаходить докази того, що вона не здатна належним чином дати відповіді на питання дослідження. Якщо математична модель визнається адекватною, тоді на її основі будується моделюючий алгоритм.

Моделюючий алгоритм – це зрозуміла інтерпретація детально розписаних дій, необхідних для побудови коректної технічної імітації моделі. Алгоритм повинен мати лише абстрактне уявлення про модель, без глибокої конкретизації, що дозволить адаптувати його під різні ситуації, допустимі дослідженням. При цьому він не повинен викривляти основні властивості об'єкта. Спільність і рівень абстракції описуваного в ньому об'єкта визначаються обмеженнями дослідження, зокрема, фінасновимі і часовими рамками. Алгоритм не залежить ні від ЕОМ, на якій будується імітація моделі, ні від від специфіки програмної мови, на якій ведеться розробка.

Результатом третього етапу є працездатна імітація заданих властивостей об'єкта, що описує його поведінку. Досягається це шляхом розробки програмного продукту, в основі якого закладено моделюючий алгоритм. Вихідний продукт повинен бути придатним для експериментів на ЕОМ і повинен надавати всю необхідну для дослідження інформацію, яку неможливо отримати аналітичним шляхом за обумовленим причинами. Вимоги до продукту так само диктуються ціллю дослідження з урахуванням накладених обмежень.

Після розробки програмного продукту, проводиться процес аналізу отриманих в ході його роботи системних даних і численні експерименти для визначення параметрично інваріантних властивостей змінних, що дозволяють економно представити характеристики системи. Важливим є те, що працюючи не з самим об'єктом, а лише з його імітацією, можна без зайвих витрат на натурні експерименти досліджувати необхідні властивості і змінювати вихідне уявлення на структурному рівні, пристосовуючи модель для різних ситуацій. Ще одна перевага такого підходу полягає в можливості досліджувати фізично недоступні об'єкти, спираючись лише на теоретичні уявлення про них (прикладом може послужити ядро нейтронної зірки, або чорна матерія).

Однак у такого методу дослідження систем є недоліки. Якщо об'єкт є динамічним, імітація моделі такого об'єкта може застаріти на структурному рівні, через що отримані результати не можуть вважатися адекватними в силу

їх неактуальність. Результат дослідження об'єктів, які описані лише теоретично, є лише теоретично вірними, так як можливості довести хибність цього твердження не існує.

Результати імітаційних експериментів можуть впливати на вид моделі об'єкта і, отже, на структуру моделюючого алгоритму. Наприклад, якщо в процесі експерименту з'ясується, що вихідні результати слабо залежать від того чи іншого параметра, то це може послужити причиною спрощення моделі (виняток даного параметра і відповідне зменшення розмірності моделі) і зміни моделюючого алгоритму.

Зобразимо схему імітаційного моделювання на рис. 3.1.

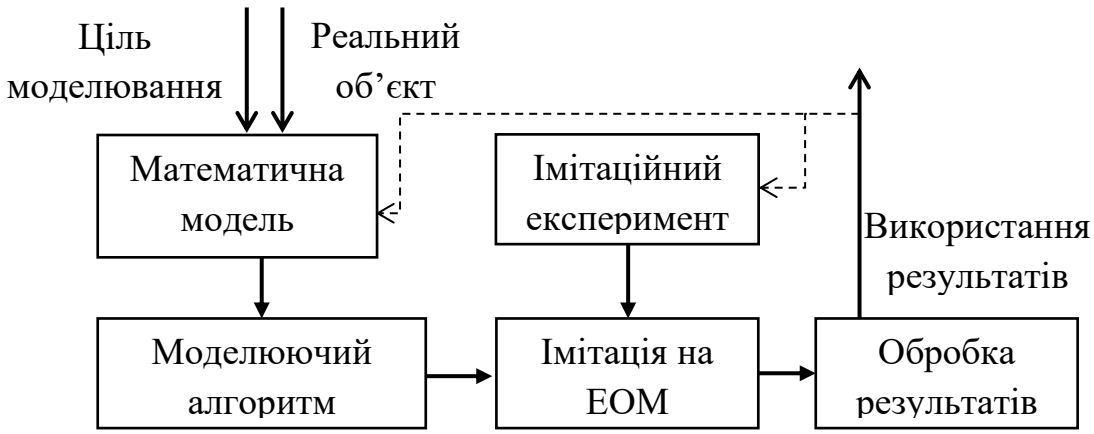


Рисунок 3.1 – Алгоритм процесу імітаційного моделювання

Реалізація методу моделювання систем залежить від фактору якісного аналізу моделі, дотримання правила збереження структури моделі при розробці алгоритму, та від технічних рішень, прийнятих в ході написання програмного продукту [8, 11, 22].

3.2 Технічний підхід до моделювання системи зі станами, що змінюються

Весь процес побудови моделі системи зі станами, що змінюються, можна представити у вигляді наступних етапів:

– визначення мети і постановка задачі дослідження;

– формалізація системи з метою побудови його математичної моделі;

– побудова математичної моделі об'єкта;

– розробка моделюючого алгоритму;

– написання програмного продукту на основі моделюючого алгоритму;

– проведення обчислень;

– аналіз результатів та їх інтерпретація;

Враховуючи специфіку задачі кваліфікаційної роботи, варто доповнити додатковими етапами: побудова альтернативної моделі складної системи та порівняльний аналіз властивостей обох моделей.

Для визначення технічних особливостей програмного продукту, який буде розроблений у результаті виконання завдання кваліфікаційної роботи, детальніше розглянемо етап побудови машинної моделі об'єкта. Перш за все потрібно зробити вибір мови програмування, на якому буде вестися розробка з урахуванням обмежень по часу, вимог продукту, написаного на цій мові (не вимагає додаткового ПЗ або апаратних змін в структурі ЕОМ), фінансових витрат (мова повинна бути безкоштовною), юридичних рамок (розробник мови програмування не повинен пред'являти права на розроблений їхньою мовою продукт, тобто програма повинна поширюватися з ліцензією вільного програмного забезпечення). Крім цього, важливим є питання оптимального використання можливостей вибраної мови для найбільш ефективного рішення поставленої задачі, завдяки чому ціль дослідження буде досягнена в достатній мірі. Технічні особливості обраної технології не повинні викривляти

результуючі характеристики системи та не надавати обмежень для досягнення цілі дослідження

Для програмного відтворення структури реальної системи на ЕОМ існує цілий ряд мов програмування, які використовують. Серед них можна визначити мови високого рівня, наприклад: Сі, Паскаль, Fortran, так як вони призначені для роботи з комплексними структурами даних. У більшості з них інтегрована підтримка строкових типів, об'єктів, операцій файлового введення або виведення тощо. Найбільш часто використовуються мови програмування, що підтримують об'єктно-орієнтований підхід (ООП). Програми на таких мовах можуть представляти сукупність об’єктів, кожен з котрих є екземпляром деякого класу. Серед таких мов можна виділити: C++, Java, C#, Python, PHP. Використовуючи принцип ООП можна використовувати для опису окремих елементів реальної системи.

Для вирішення завдання була вибрана мова PHP [16], адже вона надає найбільш підходящі під специфіку завдання технічні можливості. Для побудови структури користувальницького інтерфейсу буде використовуватись мова гіпертекстової розмітки HTML, а для опису зовнішнього вигляду документа – каскадні таблиці стилей CSS.

Сам програмний продукт буде представляти собою веб-додаток, доступ до якого можливий за допомогою веб-оглядача на будь-якому персональному комп'ютері.

Основні переваги веб-додатку:

– загальнодоступність. Можливість користуватися програмним продуктом віддалено з будь-якої точки планети;

– безпека. Дані зберігаються на віддаленому сервері, що виключає їх втрату у випадку несправності ЕОМ користувача;

– відсутність строгої вимоги до програмного забезпечення. Кожен дослідник незалежно від технічних характеристик ЕОМ може використовувати додаток. Єдина умова – наявність доступу до інтернету;

– відсутність потреби встановлювати та використовувати спеціальне ПЗ (за винятком браузера);

– кросплатформеність. Веб-додаток незалежний від операційної системи користувача. Відсутність потреби розробляти ПЗ для різних операційних систем;

– мобільність. Веб-додатком можна користуватись безпосередньо з браузера без зайвого скачування та встановлення;

– можливість налаштувати на локальну мережу. За наявності сервера дуже легко налагодити веб-додаток лише для локальної мережі;

– скриптова мова PHP розрахована для роботи з великим масивом даних, тому системи з великим числом властивостей можуть відносно швидко оброблятися;

– легке коригування роботи з базою даних;

– відсутність високого навантаження на комп'ютер користувача, так як всі процеси протікають на сервері.

Варто також враховувати недоліки такого підходу до розробки програмного продукту:

– необхідність налаштовувати і підтримувати веб-сервер, що, як правило, проводиться окремим спеціалістом;

– мова PHP не призначена для складних математичних розрахунків, отже буде довше обчислювати результати, ніж інші спеціалізовані мови;

– вимагає постійного доступу до інтернету.

Враховуючи вищеприведені переваги та недоліки, остаточним рішенням була розробка саме веб-додатку.

4 РЕАЛІЗАЦІЯ АВТОМАТИЗАЦІЇ ПОБУДОВИ МАТЕМАТИЧНОЇ МОДЕЛІ

Практична частина роботи присвячена застосуванню індуктивного методу дослідження систем зі станами, що змінюються, за допомогою розробленого програмного продукту «CSB++». Він реалізує алгоритм автоматизації процесу побудови абстрактної системи на об’єкті дослідження, формування системи даних та визначення системи зі станами, що змінюються.

4.1 Постановка задачі

Для міст з високим рівнем населення, добре розвиненою інфраструктурою і складною транспортною комунікацією питання правильного регулювання великих транспортних потоків на дорожніх розв'язках завжди було актуальним. Необхідно враховувати безліч факторів, які можуть породжуватися поза контекстом завдання, але безпосередньо впливають на неї. Наприклад, людський фактор, який найчастіше призводить до ДТП, або події природного або соціального характеру. Все це здатне порушити систему контролю за трафіком і привести до глобальних проблем як винятково міського, так і соціального плану.

Налагоджена система світлофорів дозволяє не тільки істотно оптимізувати трафік на дорогах, а й уникнути залучення людських ресурсів при виникненні складних обставин. Завдяки тому, що є можливість на технічному рівні закласти систему поведінки світлофора не тільки в різні часи доби, а й при різних обставинах, можна сказати про зв'язку таких пристроїв, як про практично самостійну складну систему управління транспортним рухом міста. Зрозуміло, така система не може на сьогоднішній день повністю гарантувати виконання поставленого завдання. Залежить це від ряду причин,

нехай то відсутність підключення до мережі електропостачання, або несправність складових частин самого пристрою.

Будемо розглядати в якості об'єкта дослідження систему роботи світлофора в різні часи доби. Беручи до уваги перераховані раніше обмеження, що накладаються на дослідження, не будемо враховувати додаткові ускладнюючі систему чинники і припустимо, що світлофор функціонує в нормальному режимі.

Позначимо напрямки, в яких рухаються транспортні потоки на перехресті: північ – південь, південь – північ, захід – схід, схід – захід. Також світлофори обладнані стрілкою лівого повороту, яка вказує в наступних напрямках: північ – схід, південь – захід, схід – південь, захід – північ.

В якості змінних виступатимуть чотири напрями руху транспорту на регульованому перехресті зі світлофором. Для кожного з них світлофор має три стани: червоний, жовтий, зелений. Всі спостереження були виконані протягом рівно 24 годин, тобто параметром в даному дослідженні буде виступати час t. Існує три основних періоди в добі (день, ніч і година пік), на протязі кожного з яких світлофори функціонують по різному. Кожен період складається з інтервалів, при зміні яких змінюються стани об'єкта. Тривалість інтервалів визначається індивідуально для кожного з періодів. Зазначимо кількісні характеристики інтервалів часу відповідно t_1, t_2, t_3, t_4 для дня:

$$0 \le \omega^{-1}(t_1) < 20, 20 \le \omega^{-1}(t_2) < 30,$$
$$30 \le \omega^{-1}(t_3) < 50, 50 \le \omega^{-1}(t_4) < 60,$$

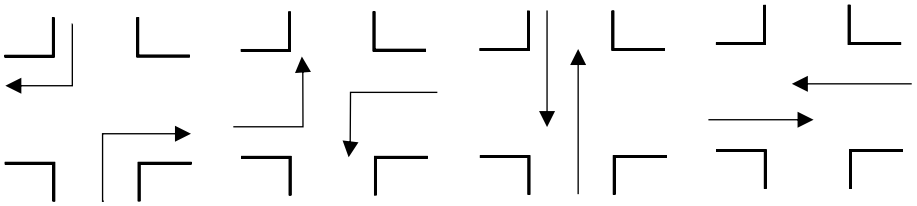
$t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}$ для години пік:

$$0 \le \omega^{-1}(t_5) < 15, 15 \le \omega^{-1}(t_6) < 25,$$
$$25 \le \omega^{-1}(t_7) < 55, 55 \le \omega^{-1}(t_8) < 65,$$
$$65 \le \omega^{-1}(t_9) < 80, 80 \le \omega^{-1}(t_{10}) < 90,$$
$$100 \le \omega^{-1}(t_{11}) < 110, 110 \le \omega^{-1}(t_{12}) < 120,$$

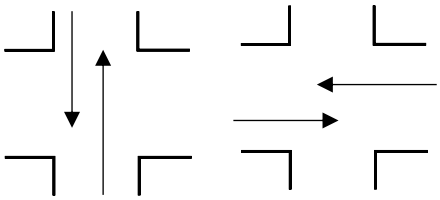
та $t_{13}, t_{14}, t_{15}, t_{16}$ для ночі:

$$0 \leq \omega^{-1}(t_{13}) < 30, 30 \leq \omega^{-1}(t_{14}) < 40,$$
$$40 \leq \omega^{-1}(t_{15}) < 50, 50 \leq \omega^{-1}(t_{16}) < 60.$$

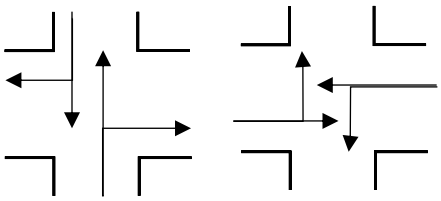
Зобразимо схематично реальні ситуації на перехресті для кожного з визначених періодів на рис. 4.1.



а) Схема роботи світлофору на перехресті вдень



б) Схема роботи світлофору на перехресті у час пік



в) Схема роботи світлофору на перехресті уночі

Рисунок 4.1 – Схема роботи світлофора у визначений період

Наведений опис об'єкта дослідження, визначені стани, умови їх виникнення та зміни служать фундаментом для подальшої побудови математичної моделі ST-системи і автоматизації даного процесу. Наведений

результат представимо у вигляді фрагмента діаграми, що характеризує переходи між станами системи та їх ймовірностями.

4.2 Визначення вихідної системи

Першим етапом будь-якого емпіричного дослідження є визначення вихідної системи, складовими якої є властивості об'єкта О. Фактично, в даному дослідженні об'єктом є перехрестя, яке регулюється світлофорами. Властивостями є напрями руху, тому, спираючись на формулу опису системи на об'єкті (2.1), доречно ввести такі позначення цих властивостей:

$$a_1: \text{ПнПд} - \text{ПдПн},$$
$$a_2: \text{СЗ} - \text{ЗС}. \quad (4.1)$$

Беручи до уваги детальний опис станів об'єкта дослідження, визначимо їх для a_1, a_2:

$$A_1 = A_2 = \{\text{зелений}, \text{жовтий}, \text{червоний}\}, \quad (4.2)$$

Позначимо напрямки, для регулювання яких пристосована стрілка лівого повороту:

$$a_3: \text{ПнС} - \text{ПдЗ},$$
$$a_4: \text{СПд} - \text{ЗПн}. \quad (4.3)$$

Стани для напрямів a_3, a_4:

$$A_3 = A_4 = \{\text{зелений}, \text{жовтий}, \text{червоний}\}. \quad (4.4)$$

На основі (4.2) – (4.5) маємо сукупність властивостей та визначених для кожного з них станів:

a_1: ПнПд − ПдПн, $A_1 = \{\text{зелений}, \text{жовтий}, \text{червоний}\}$,

a_2: СЗ − ЗС, $A_2 = \{\text{зелений}, \text{жовтий}, \text{червоний}\}$,

a_3: ПнС − ПдЗ, $A_3 = \{\text{зелений}, \text{жовтий}, \text{червоний}\}$,

a_4: СПд − ЗПн, $A_4 = \{\text{зелений}, \text{жовтий}, \text{червоний}\}$. (4.5)

Параметром, щодо якого відбуваються переходи з одного стану в інший, виступає час. Різні періоди складаються з наборів інтервалів, які оголошують порядок породження змінних. Спостереження проводяться на протязі доби (86 400 с). Визначимо, скільки буде тривати нічний період роботи світлофорів і скільки інтервалів буде здійснено за цей час.

Регулювання світлофором у нічному режимі триває з 23:00 до 06:00 –7 годин (25 200 с). Протяжність повного проходження світлофором нічного циклу дорівнює 60с. Тоді отримаємо 420 періодів повного проходження циклу зміни станів світлофора, з чого слідує, що увесь нічний період буде остаточно завершено на 25 200 секунді, після чого режим роботи світлофора автоматично змінюється на денний. Визначивши подібним чином кількість періодів за даний період та час пік, отримаємо базу:

b_1 − час зміни станів світлофора та стрілки лівого повороту,

$$B_1 = \{[0,20), \dots, [25190,25200), \dots, [86390,86400)\}. \quad (4.6)$$

Множини властивостей $a_i, i = \overline{1,4}$ з їх проявами $A_j, j = \overline{1,4}$ представлені у вигляді (4.1) та (4.6) і формують першу примітивну систему на об’єкті.

Наступним кроком є побудова конкретної представляючої системи $\dot{I}$, компонентами якої є змінні $\dot{v}_i, i = \overline{1,4}$ та параметр $\dot{w}_1$. Множинами станів змінних та параметра є $\dot{V}_i, \dot{W}_1$ відповідно. За допомогою чіткого каналу спостережень Q (2.3) відбувається їх впровадження. Функції $o_i, i = \overline{1,4}$ мають наступний вигляд:

$$o_1(\text{зелений}) = o_2(\text{зелений}) =$$
$$= o_3(\text{зелений}) = o_4(\text{зелений}) = \text{з},$$
$$o_1(\text{жовтий}) = o_2(\text{жовтий}) =$$
$$= o_3(\text{жовтий}) = o_4(\text{жовтий}) = \text{ж},$$
$$o_1(\text{червоний}) = o_2(\text{червоний}) =$$
$$= o_3(\text{червоний}) = o_4(\text{червоний}) = \text{ч}. \quad (4.7)$$

Для відображення бази необхідно представити параметр як функцію виду (4.8):

$$s_1([0,20)) = t1, \dots, s_{1680}([25190,25200)) = t1680,$$
$$\dots$$
$$s_{1920}([28790,28800)) = t1920,$$
$$\dots$$
$$s_{2401}([36000,36015)) = \text{t2401}, \quad (4.8)$$

…

$$s_{4080}\big([61190,61200)\big) = \text{t4080},$$

…

$$s_{4560}\big([68390,68400)\big) = \text{t4560},$$

…

$$s_{5760}\big([86390,86400)\big) = \text{t5760}.$$

Канал спостереження (4.7) – (4.8) задає розбиття заданої множини проявів властивостей і параметра. На основі цього розбиття можна представити конкретну представляюючу систему $\dot{I}$.

Конкретні спостережувані змінні $\dot{v}_i$ і її значення $\dot{V}_i$ тоді матимуть вигляд (4.9):

$$\begin{aligned}
&\dot{v}_1 = \text{ПнПд} - \text{ПдПн},\ \dot{V}_1 = \{\text{з}, \text{ж}, \text{к}\},\\
&\dot{v}_2 = \text{СЗ} - \text{ЗС},\ \dot{V}_2 = \{\text{з}, \text{ж}, \text{к}\},\\
&\dot{v}_3 = \text{ПнС} - \text{ПдЗ},\ \dot{V}_3 = \{\text{з}, \text{ж}, \text{к}\},\\
&\dot{v}_4 = \text{СПд} - \text{ЗПн},\ \dot{V}_4 = \{\text{з}, \text{ж}, \text{к}\}.
\end{aligned} \tag{4.9}$$

Конкретна база (параметр) $\dot{w}_j$ і множина її значень $\dot{W}_j$ матиме вигляд (4.10):

$$\begin{aligned}
\dot{w}_1 &= \text{час},\\
\dot{W}_1 &= \{t1, t2, \dots, t5760\}.
\end{aligned} \tag{4.10}$$

Після того, як був позначений канал спостереження, формується канал абстрагування (2.5) для представлення вихідних властивостей і їх станів на

базі параметра в абстрактному вигляді для подальшого формування системи даних. Цей етап характеризує побудову узагальненої представляючої системи I. Канал абстрагування визначає перехід від $\dot{I}$ до I. Визначемо $\gamma_i: \dot{V}_i \to V_i$:

$$\gamma_1(\text{з}) = \gamma_2(\text{з}) = \gamma_3(\text{з}) = \gamma_4(\text{з}) = 0,$$
$$\gamma_1(\text{ж}) = \gamma_2(\text{ж}) = \gamma_3(\text{ж}) = \gamma_4(\text{ж}) = 1,$$
$$\gamma_1(\text{ч}) = \gamma_2(\text{ч}) = \gamma_3(\text{ч}) = \gamma_4(\text{ч}) = 2, \quad (4.11)$$

та $\omega_j: \dot{W}_j \to W_j$:

$$\omega_1(t1) = 1,$$
$$\omega_2(\text{t2}) = 2,$$
$$\dots$$
$$\omega_{5760}(\text{t5760}) = 5760. \quad (4.12)$$

Через канал абстрагування визначаємо множини значень узагальнених спостережуваних змінних v_i (4.13):

$$V_1 = \{0,1,2\},$$
$$V_2 = \{0,1,2\},$$
$$V_3 = \{0,1,2\},$$
$$V_4 = \{0,1,2\}, \quad (4.13)$$

та множину значень узагальненого параметру w_j:

$$W_1 = \{1,2,\dots,5760\}. \qquad (4.14)$$

Отримавши узагальнену представляючу систему I (4.13) – (4.14) можна вважати етап формування вихідної системи завершеним.

Поведінку світлофора необхідно дослідити протягом доби, а кількість його режимів роботи дорівнює трьом (ніч, день, час пік), причому кожен такий період розбивається на інтервали наступним чином: нічний режим продовжується 420 хвилин, розбивається на чотири інтервали, що охоплюють 60 секунд, тобто загальна кількість інтервалів у нічному режимі роботи світлофора дорівнює $4 \times 420 = 1680$. Аналогічно визначається кількість інтервалів для режиму дня – 4000, та для режиму роботи у час пік – 960. Отримаємо кінцеву довжину параметричної множини, яка становить 1680 + 4000 + 960 = 5760 елементів. Оскільки параметрична множина досить громіздка, аналітичний метод визначення системи з станами, які змінюються, вважається недоцільним по відношенню до затрачених ресурсів та незручної інтерпретації результату користувачеві. Для того, щоб мати зручний інструмент, який дозволить без зайвих витрат побудувати вихідну систему досліджуваного об'єкта з можливістю задавати властивості і їх стани динамічно, був розроблений відповідний функціонал в загальному програмному продукті «CSB++» (complex system building++). Виходячи з поставленого завдання дослідити конкретний об'єкт у вигляді перехрестя з чотирма світлофорами, немає сенсу реалізовувати більш гнучкий функціонал для адаптації алгоритму під об'єкти іншого роду.

Інтерфейс програми на рис. 4.3 – 4.4 є формою введення даних і надає право вибору станів властивостей a_1, a_2, a_3, a_4 із заданої множини для конкретного параметра t. Користувач має можливість задати який колір має той чи інший світлофор на перехресті в кожен з інтервалів кожного періоду часу.

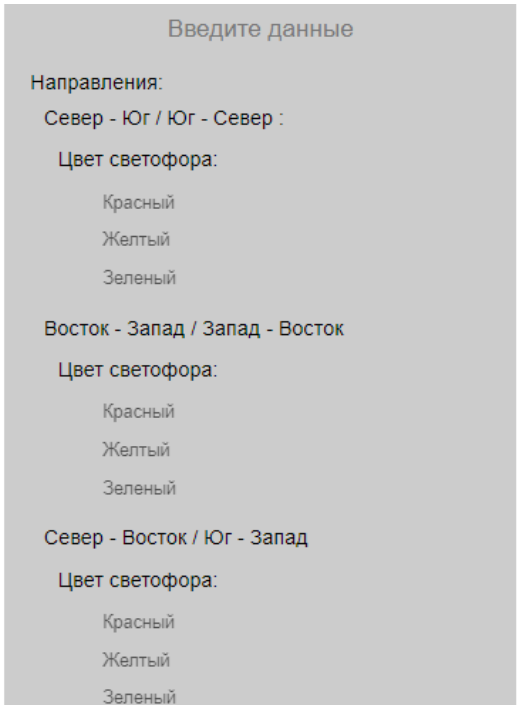


Рисунок 4.3 – Форма вводу властивостей a_1, a_2, a_3, a_4 та їх станів

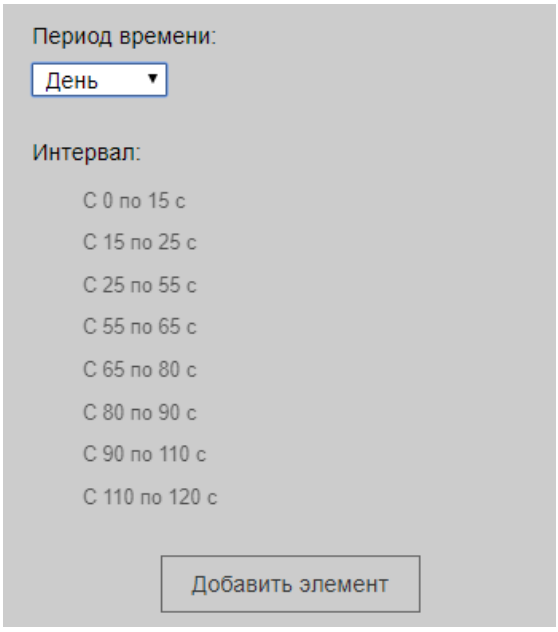


Рисунок 4.4 – Форма вводу параметра t

Нагадаємо, що спостереження проводяться протягом доби, а загальна кількість інтервалів, що позначають переходи з одного стану в інший, вважається рівним 5760. Зрозуміло, користувачеві не потрібно вводити усі інтервали вручну, так як поведінка світлофорів закономірно повторюється з кожним новим циклом в межах конкретного періоду доби. Програма

самостійно заповнить всю систему на основі лише одного разу введених даних.

Після введення початкових даних вони зберігаються в базі даних для подальшого використання. Як вже було зазначено раніше, однією з переваг веб-додатку, написаного на скриптовій мові PHP, є підтримка роботи з базою даних на рівні самої мови, через що на опис архітектури додатку і саму розробку функціоналу витрачається значно менше часу. Дані, які були внесені одним користувачем, можуть в подальшому використовуватися іншими для коригування, доповнення або фільтрації.

Додаток запропонує сформувати вихідну систему на об'єкті. Користувач може або погодитися, або ігнорувати пропозицію, наприклад, поки не відредагує вихідні дані перед процесом формування.:

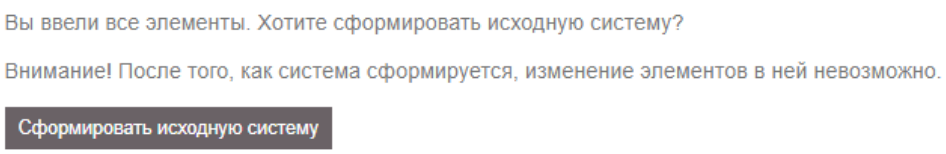


Рисунок 4.5 – Додаток розуміє, що всі необхідні дані для формування вихідної системи введені, тому пропонує її створити

Як результат вводу вихідних даних, програма, використовуючи формули (2.1) – (2.6), повністю формує вихідну систему. На основі введених даних буде сформована система даних.

4.3 Формування системи даних

Після визначення вихідної системи стає можливим збір даних. Весь процес зводиться до визначення значень відібраних змінних при певних значеннях баз і записи цих даних в деякій відповідній формі. Дані можна отримати або через спостереження (використовуючи канали спостереження),

тобто шляхом вимірювання. Якщо користувач має можливість керувати даними, то він може використати цю можливість для спостереження так званих вихідних властивостей, які будуть доповнювати систему. Як результат формується система даних D (2.7).

Беручи до уваги ту обставину, що параметричну множину умовно можна розбити на три підмножини, що явно представляють кожен з описаних раніше періодів спостереження, під час яких поведінка об'єкта істотно відрізнялося, для зручності подання системи даних, представимо її аналогічним чином, розбивши на три підсистеми, кожна з яких строго відповідає певному періоду спостереження. Позначимо ці системи як множину $D = \{D_1, D_2, D_3\}$.

Користуючись (2.7) сформуємо систему з чіткими даними, оскільки для визначення нульового епістемологічного рівня використовувався чіткий канал спостереження. Так як час спостережень за об’єктом дослідження складається з трьох множин, які визначають відмінність у поведінці виділених властивостей об’єкта, то для зручності розділимо систему даних на три підмножини: D_1, D_2, D_3, які визначають регулювання рухом на перехресті уночі, вдень та в час пік відповідно. Покажемо ці елементи у вигляді табл. 4.1 – 4.3.

Таблиця 4.1 – Система даних D_1 описує регулювання рухом вдень

$\dot{v}_j$ \ $\dot{w}_i$	t_{i+1}	t_{i+2}	t_{i+3}	t_{i+4}	t_{i+5}	t_{i+6}	t_{i+7}	t_{i+8}
СЗ – ЗС	ч	ч	з	ж	ч	ч	ч	ч
СПд – ЗПн	ч	ч	ч	ч	ч	ч	з	ж
ПнС – ПдЗ	ч	ч	ч	ч	з	ж	ч	ч
ПнПд – ПдПн	з	ж	ч	ч	ч	ч	ч	ч

Таблиця 4.2 – Система даних D_2 описує регулювання рухом у час пік

$\dot{w}_i$ / $\dot{v}_j$	t_{i+9}	t_{i+10}	t_{i+11}	t_{i+12}
СЗ – ЗС	з	ж	ч	ч
СПд – ЗПн	ч	ч	з	ж
ПнС – ПдЗ	ч	ч	ч	ч
ПнПд – ПдПн	ч	ч	ч	ч

Таблиця 4.3 – Система даних D_3 описує регулювання рухом уночі

$\dot{w}_i$ / $\dot{v}_j$	t_{i+13}	t_{i+14}	t_{i+15}	t_{i+16}
СЗ – ЗС	з	ж	ч	ч
СПд – ЗПн	ч	ч	з	ж
ПнС – ПдЗ	ч	ч	з	ж
ПнПд – ПдПн	з	ж	ч	ч

Програма «CSB++» автоматично побудує систему даних на основі сформованої в першому етапі вихідній системи і дасть вивід користувачеві дві інтерпретації: систему даних з семантикою і без семантики. Алгоритм, закладений в функціонал побудови програмним продуктом систем даних обох видів, використовує для цього формулу (2.8). Результат процедури формування систем даних за допомогою веб-додатку «CSB++» можна побачити на рис. 4.6.

Направленная система данных с четкими данными без семантики

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	1	0	0	2	1	0	0	2	1	0	0	2	1	0	0	2	1
0	0	2	1	0	0	2	1	0	0	2	1	0	0	2	1	0	0
0	0	2	1	0	0	2	1	0	0	2	1	0	0	2	1	0	0
0	0	2	1	0	0	2	1	0	0	2	1	0	0	2	1	0	0

Направленная система данных с четкими данными с семантикой

t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16
з	ж	к	к	з	ж	к	к	з	ж	к	к	з	ж	к	к
к	к	з	ж	к	к	з	ж	к	к	з	ж	к	к	з	ж
з	ж	к	к	з	ж	к	к	з	ж	к	к	з	ж	к	к
к	к	з	ж	к	к	з	ж	к	к	з	ж	к	к	з	ж

Рисунок 4.6 – Фрагменти систем даних з семантикою та без семантики відносно

Система даних без семантики матиме більший сенс для подальшого підйому по епістемологічним рівням, так як має виключно математично описані характеристики параметра і станів, в той час, як система з семантикою добре сприймається користувачем інтуїтивно.

Виходячи з цього твердження на наступному етапі буде докладно описаний процес побудови системи зі станами, що змінюються, на основі системи даних без семантики.

4.4 Побудова системи зі станами, що змінюються

Після того, як дані були визначені слід наступний етап емпіричного дослідження – обробка даних. На цьому етапі визначаються параметрично інваріантні властивості змінних v_1, v_2, v_3, v_4, що дозволить економно представити дані в деякому зручному для користувача вигляді надалі.

Для формування другого епістемологічного рівня згадаємо термін системи з поведінкою, яка визначається за формулою (2.9). По суті, це система з характеристиками загального параметрично інваріантного обмеження на розглянуті змінні узагальненої системи. Будь-яка система другого епістемологічного рівня вважається системою з поведінкою. У розділі 2 були введені системи даного рівня: спрямовані системи з поведінкою (2.20) і ST-системи (2.29).

Будь-яка ST-система може бути перетворена в ізоморфну систему з поведінкою, проте зворотне перетворення можливо лише при певному типі масок. З цього випливає, що система з поведінкою більш узагальнена, ніж ST-система. Слід так само згадати, що під час більш складних досліджень оптимальним рішенням вважається побудова на цьому етапі не ST-системи, а традиційно системи з поведінкою. Це пов'язано з тим, що ST-системи надлишкові, так як поточні і наступні стани накладаються і допустимо використання тільки масок без пам'яті. Якщо користувачеві потрібно ST-система, то його завдання вирішується за допомогою ізоморфних систем з поведінкою (при цьому, як правило, застосовується спрощення, якщо система вважається занадто складною структурно або якщо деякі змінні випадають з розгляду). Після формування відбувається перетворення з результуючих систем з поведінкою в більш зручні для сприйняття користувачем ST-системи.

Проведемо паралель між системою з поведінкою і ST-системою. Так як в програмному продукті закладено функціонал побудови системи з поведінкою, з якої можливо сформувати породжуючу систему з поведінкою,

скористаємося цим, щоб показати перевагу ST-систем в поданні результуючих даних.

Використовуючи маску M^1 та правилом зрушення $q = (0,1)$, сформуємо за допомогою «CSB++» систему з поведінкою (рис. 4.7).

S1	S2	S3	S4	S5	S6	S7	S8	$f_B(C)$
0	2	2	0	0	1	1	0	0.00017367141368531
0	1	1	0	0	0	0	0	0.00017367141368531
0	0	0	0	0	0	0	0	0.13563737408823
0	0	0	0	2	0	0	0	0.18235498436957
2	0	0	0	1	0	0	0	0.00017367141368531
1	0	0	0	0	0	0	2	0.00017367141368531
0	0	0	2	0	0	0	1	0.00017367141368531
0	0	0	1	0	2	2	0	0.00017367141368531
0	2	2	0	0	1	1	0	0.00017367141368531
0	1	1	0	0	0	0	0	0.00017367141368531
0	0	0	0	0	0	0	0	0.13563737408823
0	0	0	0	2	0	0	0	0.18235498436957
2	0	0	0	1	0	0	0	0.00017367141368531
1	0	0	0	0	0	0	2	0.00017367141368531
0	0	0	2	0	0	0	1	0.00017367141368531
0	0	0	1	0	2	2	0	0.00017367141368531
0	2	2	0	0	1	1	0	0.00017367141368531
0	1	1	0	0	0	0	0	0.00017367141368531
0	0	0	0	0	0	0	0	0.13563737408823
0	0	0	0	2	0	0	0	0.18235498436957
2	0	0	0	1	0	0	0	0.00017367141368531
1	0	0	0	0	0	0	2	0.00017367141368531
0	0	0	2	0	0	0	1	0.00017367141368531
0	0	0	1	0	2	2	0	0.00017367141368531
0	2	2	0	0	1	1	0	0.00017367141368531

Рисунок 4.7 – Фрагмент системи з поведінкою

Програма визначає множину вибіркових змінних та ймовірність, з якою стани вибіркових змінних входять у загальну множину станів. Інтерпретувати поведінку об'єкта за поточною системою не дуже зручно, адже система не відповідає на питання про породження одних змінних іншими. Досліднику безумовно буде недостатньо поточних даних для того, що простежити

переходи одного стану в інший, тому подальшим кроком є побудова породжуючої системи з поведінкою (2.19).

У завданні не вказується явний вплив об'єкта на навколишнє середовище. Так само передбачається, що під час побудови системи даних користувач не буде керувати ними для визначення вихідних змінних, тому структурно нейтральна і спрямована системи з поведінкою в поточному прикладі вважаються однаковими, адже множина вихідних змінних буде порожньою. Побудуємо породжуючу систему з поведінкою за допомогою «CSB++». Програма бере до уваги алгоритм побудови системи з поведінкою та розділяє вибіркові дані на породжувані та породжуючі. Розраховується ймовірність, з якою вони виникають у відповідних множинах. Результат представлений на рис. 4.7. Ліва таблиця формується з породжуючих змінних, а права – з породжуваних.

Згідно опису направленої системи з поведінкою та направленої породжуючої системи з поведінкою (2.19), програма розбиває множину вибіркових змінних на вхідні та вихідні. Враховуючи, що у даному завданні не передбачено явного впливу об’єкта на зовнішню середу, множина вихідних змінних буде порожньою. Отже, у даному конкретному прикладі нейтральну та направлену системи з поведінкою можна вважити однаковими за структурою.

Форма подання обмеження на вибіркові змінні для ST-систем на відміну від системи з поведінкою, визначається не на окремих станах, а на послідовних парах станів. Тобто основна відмінність полягає в тому, що в результаті повної вибірки даних виходять частоти пар послідовних станів, а не частоти окремих станів.

S1	S2	S3	S4	$f_B(g\_)$	S5	S6	S7	S8	$f_B(g)$
2	0	0	0	0.18232332002084	1	0	0	0	0.18232332002084
1	0	0	0	0.18232332002084	0	2	2	2	0.072929328008335
0	2	2	2	0.072929328008335	0	1	1	1	0.072929328008335
0	1	1	1	0.072929328008335	2	0	0	0	0.18214967876367
2	0	0	0	0.18232332002084	1	0	0	0	0.18232332002084
1	0	0	0	0.18232332002084	0	2	2	2	0.072929328008335
0	2	2	2	0.072929328008335	0	1	1	1	0.072929328008335
0	1	1	1	0.072929328008335	2	0	0	0	0.18214967876367
2	0	0	0	0.18232332002084	1	0	0	0	0.18232332002084
1	0	0	0	0.18232332002084	0	2	2	2	0.072929328008335
0	2	2	2	0.072929328008335	0	1	1	1	0.072929328008335
0	1	1	1	0.072929328008335	2	0	0	0	0.18214967876367
2	0	0	0	0.18232332002084	1	0	0	0	0.18232332002084
1	0	0	0	0.18232332002084	0	2	2	2	0.072929328008335
0	2	2	2	0.072929328008335	0	1	1	1	0.072929328008335
0	1	1	1	0.072929328008335	2	0	0	0	0.18214967876367
2	0	0	0	0.18232332002084	1	0	0	0	0.18232332002084
1	0	0	0	0.18232332002084	0	2	2	2	0.072929328008335
0	2	2	2	0.072929328008335	0	1	1	1	0.072929328008335
0	1	1	1	0.072929328008335	2	0	0	0	0.18214967876367
2	0	0	0	0.18232332002084	1	0	0	0	0.18232332002084
1	0	0	0	0.18232332002084	0	2	2	2	0.072929328008335
0	2	2	2	0.072929328008335	0	1	1	1	0.072929328008335
0	1	1	1	0.072929328008335	2	0	0	0	0.18214967876367
2	0	0	0	0.18232332002084	1	0	0	0	0.18232332002084
1	0	0	0	0.18232332002084	0	2	2	2	0.072929328008335
0	2	2	2	0.072929328008335	0	1	1	1	0.072929328008335

Рисунок 4.7 – Фрагмент породжуючої системи з поведінкою

Відповідно до заданого на параметричній множині порядку породження, можна сказати про пару станів як про перехід одного в інший. Ймовірності або можливості такого переходу повинні задовольняти умову рівноваги, позначене формулою (2.32). Таким чином, ведучи систематичну аналітику поведінки об'єкта можна на підставі даних ST-системи легко знайти проблемні

вузли, здатні вказати на нестабільність деяких структурних компонентів досліджуваного об'єкта при певному значенні параметрів. Так само на основі отриманих даних про переходи можна зробити аналіз і сформувати теоретичне уявлення про подальшу поведінку об'єкта.

Побудуємо ST-систему, використовуючи програмний продукт «CSB++».

	0000	0001	0002	0010	0011	0012	0020	0021	0022	0100	0101	0102	0110	0111	0112	0120	0121	0122	0200	0201	0202	0210	0211
0000	0.067731851	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0002	0	0.067731851	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0012	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0020	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0022	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0102	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0110	0.067558179	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0111	0.000173671	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0112	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0121	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0122	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0201	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0202	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0210	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0211	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0212	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0220	0	0	0	0	0	0	0	0	0	0	0	0	0.109412990	0	0	0	0	0	0	0	0	0	0

Рисунок 4.7 – Фрагмент ST-системи

На рис. 4.7 зображений фрагмент матриці, колонки якої представляють наступний стан, а рядки – поточний. Матриця заповнена ймовірностями того, що поточний стан перейде у конкретне наступне. Варто звернути увагу, що йде повний перебір всіх потенційно можливих значень станів, тому очевидно, що деякі ймовірні стани були відсутні під час спостережень в принципі. Імовірність переходу до таких станів нульова, тому колонки з відсутніми значеннями заповнені нулями.

Така система зручна тим, що користувач може наочно побачити переходи з одного стану в інший і ймовірність цих переходів. Однак, інтерфейс програми перевантажений через велику кількість відсутніх станів

(ймовірність переходів у такі стани дорівнюють нулю), тому бажано представити результат у вигляді діаграми.

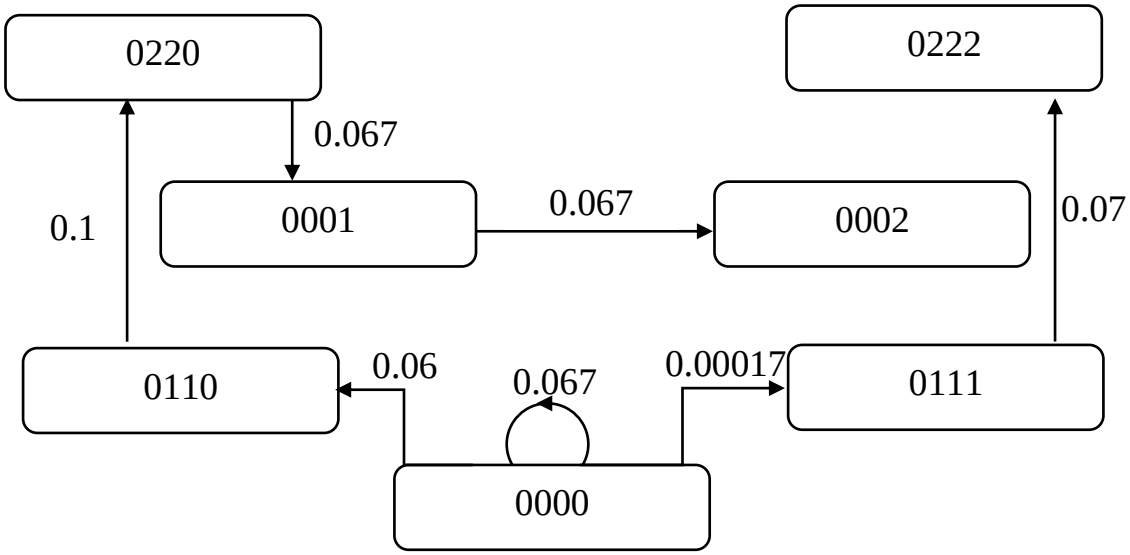


Рисунок 4.8 – Діаграма переходів певних станів ST-системи

На рис. 4.8 представлена діаграма, яка візуально інтерпретує фрагмент системи зі станами, що змінюються. На ній видно, що деякі стани можуть переходити самі в себе (стан 0000 переходить сам в себе з ймовірністю 0.067). Так само можна відзначити деякі стани, перехід в які вважається найменш вірогідними (той же стан 0000 перейде в стан 0111 з вірогідністю всього в 0.000176). Ці дані можуть в подальшому використовуватися фахівцем для аналізу ефективності світлофора на регульованому перехресті на певному інтервалі часу.

ВИСНОВКИ

Кваліфікаційна робота присвячена дослідженню питання про сутність системи зі станами, що змінюються, і проблемі автоматизації процесу індуктивного моделювання такої системи. Важливим вважається визначення переваг і недоліків побудови моделі системи зі станами, що змінюються, по відношенню до моделі систем з поведінкою.

Метою роботи є автоматизація процесу індуктивного моделювання системи зі станами, що змінюються, використовуючи сучасні принципи розробки відповідних програмних продуктів з урахуванням загальнодоступності і мобільності кінцевого програмного забезпечення.

Проведене дослідження умовно можна розбити на наступні етапи:

– розкриття теоретичної частини дослідження. Наведена формалізація складної системи, що дозволило розкрити сутність системи з станами, які змінюються. Розглянуто приклади реальних об'єктів зовнішнього світу, поведінку яких можна змоделювати за допомогою ST-систем;

– огляд індуктивного методу моделювання систем, його важливості в емпіричному дослідженні і місце в загальній класифікації найбільш застосовних методів моделювання;

– вирішення технічної частини питання про автоматизацію процесу моделювання. Докладний огляд основних етапів проектування моделюючого алгоритму і приведення доводів з приводу вибору методики і технологій для реалізації проекту з урахуванням як переваг здійсненого вибору, так і недоліків;

– розробка програмного продукту «CSB++», який вирішує поставлене перед дослідженням питання автоматизації процесу індуктивного моделювання системи зі станами, що змінюються. Аналіз та інтерпретація отриманих в результаті роботи програмного продукту «CSB++» даних та

приведення порівняльної характеристики між результатом моделювання системи зі станами, що змінюються і системою з поведінкою.

Значна частина першого розділу була присвячена опису предметної області та розкриття актуального стану поставленого питання про автоматизацію процесу моделювання складних систем. Присутні роздуми про складність визначення складних систем і наводяться спроби формалізувати поняття системи з станами, що змінюються в контексті завдань індуктивного моделювання.

У другому розділі розглядається алгоритм проведення емпіричного дослідження. Докладно описується індуктивне моделювання і наводиться поверхневий огляд альтернативних методів моделювання систем з станами, які змінюються.

Третій розділ присвячений розгляду основних етапів проектування та розробки моделюючого алгоритму. Докладно описані причини обраних для розробки програмного продукту технологій, їх роль і важливість в умовах сучасного дослідження і недоліки при проектуванні складних моделюючих алгоритмів. На теоретичній основі, приведеної в перших трьох розділах, розроблено програмний продукт «CSB++» (Complex System Building++), який на основі введених користувачем даних повністю автоматизує процес побудови вихідної системи, системи даних (з семантикою та без семантики), системи зі станами, що змінюються. На основі даних, отриманих в результаті проведення обчислювальних експериментів з розробленим програмним продуктом, наводяться аргументи на користь доречності використання ST-систем по відношенню до систем з поведінкою. Для наочності зроблених висновків отриманий результат інтерпретується для більш зручного представлення користувачу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Бусленко М. П. Моделирование сложных систем. Москва : Наука, 1968. 356 с.

2. Щедровицкий Г. П. Проблемы методологии системного исследования. Москва : Знание, 1964. 48 с.

3. Юдин А. Д., Юдин Д. В. Число и мысль. Выпуск 8. (Математики измеряют сложность). Москва : Знания, 1985. 192 с.

4. Лузина Л. И. Компьютерное моделирование : учебное пособие. Томск : Томский межвузовский центр дистанционного образования, 2001. 105 с.

5. Лефевр В. А. Конфликтующие структуры. Москва : Советское радио, 1973. 343 с.

6. Зеленский В. А. Проектирование сложных. URL: http://rtfmoodle.ssau.ru. (Дата звернення 05.04.2018).

7. Этапы и принципы моделирования объектов и систем на ЭВМ. URL: http://poznayka.org/s22171t1.html. (Дата звернення 28.03.2018).

8. Ешби Р. У. Введение в кибернетику. Москва : Издательство иностранной литературы, 1959. 422 с.

9. Клир Дж. Системология. Автоматизация решения системных задач. Москва : Радио и связь, 1990. 544 с.

10. Дхедан Х., Фомин Б. Р. Вступление в системологии: эпистемологические уровне систем : лекция. Санкт-Петербург : Санкт-Петербургский государственный электротехнический университет «ЛЭТИ», 1984. 5 с.

11. Беллман Р. Динамическое программирование. Москва : Иностранная литература, 1960. 400 с.

12. Касти Д. Большие системы. Возможности подключения, сложность и катастрофы. Москва : Мир, 1982. 216 с.

13. Хадсон П. PHP. Справочник. Санкт-Петербург : КУДИЦ-Пресс, 2006. 448 с.

14. Парето В. Социалистические системы. Москва : Директ-Медиа, 2007. 131 с.

15. Остин О. Теория графов. Москва : Наука, 1980. 336 с.

16. Клод Ш. Математическая теория связи. Москва: ИЛ, 1963. 322 с.

17. Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов. Москва : Мир, 1981. 366 с.

18. Перегудов Ф. И., Тарасенко Ф. П. Введение в системный анализ. Москва : Высшая школа, 1989. 360 с.

19. Олянич Д. Б. Теория организации: учебник. Ростов : Феникс, 2008. 408 с.

20. Пиотровский Я. Теория измерений для инженеров. Москва : Мир, 1989. 336 с.

ДОДАТОК А

Код програмного продукту «CSB++»

```
<?php
set_time_limit(0);
require_once('st-system.php');
use \Systems\StSystem;
$mysqli = connect();;
/**
 * Format and save data to cookie
 */
if (isset($_POST["yes"])) {
   $inputData = array("name" => "",
      "interval_value" => "",
      "color_1_value" => "",
      "color_2_value" => "",
      "color_3_value" => "",
      "color_4_value" => "");
   if (isset($_COOKIE["temp"])) {
      foreach ($_COOKIE["temp"] as $name => $value) {
         $name = htmlspecialchars($name);
         $value = htmlspecialchars($value);
         $inputData[$name] = $value;
      }
   }
   setcookie("interval" . $inputData['name'] . "[interval_value]", "" . $inputData['interval_value']);
   setcookie("interval" . $inputData['name'] . "[color_1_value]", "" . $inputData['color_1_value']);
   setcookie("interval" . $inputData['name'] . "[color_2_value]", "" . $inputData['color_2_value']);
   setcookie("interval" . $inputData['name'] . "[color_3_value]", "" . $inputData['color_3_value']);
   setcookie("interval" . $inputData['name'] . "[color_4_value]", "" . $inputData['color_4_value']);
}
if (isset($_POST["add-object"])) {
   mutateInputData();
}
$keptAmount = 0;
/**
 * Amount of the kept input data
 */
for ($i = 1; $i <= 3; $i++) {
   if ($i == 2) {
      for ($j = 1; $j <= 8; $j++) {
         if (isset($_COOKIE["interval" . $i . "" . $j])) {
            $keptAmount++;
         }
      }
   } else {
      for ($j = 1; $j <= 4; $j++) {
         if (isset($_COOKIE["interval" . $i . "" . $j])) {
            $keptAmount++;
         }
      }
   }
}
```

```
/**
 * Propose to save source system
 */
if ($keptAmount == 16) {
   echo "<div class='change'>";
   echo "Вы ввели все элементы. Хотите сформировать исходную систему?<br><br>";
   echo "Внимание! После того, как система сформируется, изменение элементов в ней
невозможно.<br><br>";
   echo '<input class="btn" type="submit" value="Сформировать исходную систему" name="create-base"
form="enter-form">';
   echo "</div>";
}
/**
 * Connect DB
 */
function connect()
{
   $mysqli = new mysqli("127.0.0.1", "diplom", "WDD8zh", "source_system", 3306);
   if ($mysqli->connect_error) {
      die('Ошибка подключения (' . $mysqli->connect_errno . ') ' . $mysqli->connect_error);
   }
   $mysqli->query("SET NAMES 'utf8'");
   return $mysqli;
}
function mutateInputData()
{
   $color_1 = $_POST["color-1"];
   $color_2 = $_POST["color-2"];
   $color_3 = $_POST["color-3"];
   $color_4 = $_POST["color-4"];
   if ($_POST["period"] == 1) {
      $interval = $_POST["night-interval"];
   }
   if ($_POST["period"] == 2) {
      $interval = $_POST["day-interval"];
   }
   if ($_POST["period"] == 3) {
      $interval = $_POST["rush-hour-interval"];
   }
   $enter_name = "" . $_POST["period"] . $interval;
   if (isset($_COOKIE["interval" . $enter_name . ""])) {
      setcookie("temp[name]", "" . $enter_name);
      setcookie("temp[interval_value]", "" . $interval);
      setcookie("temp[color_1_value]", "" . $color_1);
      setcookie("temp[color_2_value]", "" . $color_2);
      setcookie("temp[color_3_value]", "" . $color_3);
      setcookie("temp[color_4_value]", "" . $color_4);
      echo "<div class='change'>";
      echo "Этот элемент уже добавлен!<br><br>";
      echo "Хотите изменить параметры существующего элемента?<br><br>";
      echo '<input class="btn" type="submit" value="Да" name="yes" form="enter-form">';
      echo "</div>";
   } else {
      setcookie("interval" . $enter_name . "[interval_value]", "" . $interval);
      setcookie("interval" . $enter_name . "[color_1_value]", "" . $color_1);
      setcookie("interval" . $enter_name . "[color_2_value]", "" . $color_2);
      setcookie("interval" . $enter_name . "[color_3_value]", "" . $color_3);
      setcookie("interval" . $enter_name . "[color_4_value]", "" . $color_4);
   }
```

```
}
/**
 * Save into DB
 */
try {
   if (isset($_POST["create-base"])) {
      $mysqli->query("ALTER TABLE `intervals` AUTO_INCREMENT=1");
      $time = 0;
      $num = 0;
      function insert_into_DB($intervals, $periods, $amount)
      {
         for ($i = 1; $i <= $periods; $i++) {
            for ($j = 1; $j <= $intervals; $j++) {
               global $num;
               global $time;
               $num += 1;
               $namedInterval = "t" . $num;
               $entryArr = array("interval_value" => "", "color_1_value" => "", "color_2_value" => "", "color_3_value"
=> "", "color_4_value" => "");
               if (isset($_COOKIE["interval" . $amount . "" . $j])) {
                  //echo "interval".$i."".$j.":<br>";
                  foreach ($_COOKIE["interval" . $amount . "" . $j] as $name => $value) {
                     $name = htmlspecialchars($name);
                     $value = htmlspecialchars($value);
                     $entryArr[$name] = $value;
                  }
               }
               /**
                * Set interval for current item
                */

               if ($amount === 1) {
                  if (((int)$entryArr["interval_value"] == 1) || ((int)$entryArr["interval_value"] == 3)) {
                     $from = $time;
                     $to = $time + 20;
                     $time += 20;
                  }
                  if (((int)$entryArr["interval_value"] == 2) || ((int)$entryArr["interval_value"] == 4)) {
                     $from = $time;
                     $to = $time + 10;
                     $time += 10;
                  }
               }
               if ($amount === 2) {
                  if (((int)$entryArr["interval_value"] == 1) || ((int)$entryArr["interval_value"] == 5)) {
                     $from = $time;
                     $to = $time + 15;
                     $time += 15;
                  }
                  if (((int)$entryArr["interval_value"] == 2) ||
                     ((int)$entryArr["interval_value"] == 4) ||
                     ((int)$entryArr["interval_value"] == 6) ||
                     ((int)$entryArr["interval_value"] == 8)) {
                     $from = $time;
                     $to = $time + 10;
                     $time += 10;
                  }
                  if (((int)$entryArr["interval_value"] == 3)) {
                     $from = $time;
```

```
                $to = $time + 30;
                $time += 30;
            }
            if (((int)$entryArr["interval_value"] == 7)) {
                $from = $time;
                $to = $time + 20;
                $time += 20;
            }
        }
        if ($amount === 3) {
            if (((int)$entryArr["interval_value"] == 1)) {
                $from = $time;
                $to = $time + 30;
                $time += 30;
            }
            if (((int)$entryArr["interval_value"] == 2) || ((int)$entryArr["interval_value"] == 3) ||
((int)$entryArr["interval_value"] == 4)) {
                $from = $time;
                $to = $time + 10;
                $time += 10;
            }
        }
        /**
         * Set math parameters
        */
        for ($k = 1; $k <= 4; $k++) {
            $colors[$k - 1] = $entryArr["color_" . $k . "_value"];
            if ((int)$colors[$k - 1] === 1)
                $colors[$k - 1] = "к";
            if ((int)$colors[$k - 1] === 2)
                $colors[$k - 1] = "ж";
            if ((int)$colors[$k - 1] === 3)
                $colors[$k - 1] = "з";
        }
        /**
         * Insertion of the new system element
        */
        global $mysqli;
        $success = $mysqli->query("INSERT INTO `intervals` (`name`, `interval_from`,
`interval_to`,`color_1`,`color_2`, `color_3`, `color_4`,`color_1_math`, `color_2_math`, `color_3_math`,
`color_4_math`) VALUES('" . $namedInterval . "','" . $from . "','" . $to . "','" . $colors[0] . "','" . $colors[1] . "','" .
$colors[2] . "','" . $colors[3] . "','" . $entryArr['color_1_value'] . "','" . $entryArr['color_2_value'] . "','" .
$entryArr['color_2_value'] . "','" . $entryArr['color_4_value'] . "')");
        if ($success != true) {
            echo "<br>Проблемки с добавлением нового объекта.<br>";
        }
      }
    }
    return $num;
}
/**
 * Run saving elements for different intervals
*/
insert_into_DB(4, 420, 1, $num);
insert_into_DB(8, 30, 2, $num);
insert_into_DB(4, 120, 3, $num);
insert_into_DB(8, 210, 2, $num);
insert_into_DB(4, 120, 3, $num);
insert_into_DB(8, 150, 2, $num);
```

```
        echo "<br> Система сформирована!<br>";
    }
} catch (Exception $e) {
    var_dump($e);
    die();
}
?>
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="css/normalize.css">
    <link rel="stylesheet" href="css/styles.css">
    <script src='https://code.jquery.com/jquery-2.2.4.min.js'></script>
</head>
<body>
<div class="container">
    <section class="enter-show-point">
        <div class="enter" id="enter-block">
            <button class="enter__toggle" id="enter-toggle">Добавить объект в базу</button>
            <div class="enter__inner" style="display: none;">
                <form action="index.php" class="enter__form" method="post" id="enter-form">
                    <fieldset class="enter__form-inner">
                        <legend class="enter__form-title">Введите данные</legend>
                        <div class="enter__form-field">
                            <div class="enter__form-fieldname">Направления:</div>
                            <div class="ml-1 enter__form-fieldname">Север - Юг / Юг - Север :</div>
                            <div class="ml-2  enter__form-field" id="colors">
                                <div class="enter__form-fieldname">Цвет светофора:</div>
                                <input type="radio" class="hidden" name="color-1" id="red1" value="1">
                                <input type="radio" class="hidden" name="color-1" id="yellow1" value="2">
                                <input type="radio" class="hidden" name="color-1" id="green1" value="3">
                                <label for="red1" class="enter__form-control">Красный</label>
                                <label for="yellow1" class="enter__form-control">Желтый</label>
                                <label for="green1" class="enter__form-control">Зеленый</label>
                            </div>
                            <div class="ml-1  enter__form-fieldname">Восток - Запад / Запад - Восток</div>
                            <div class="ml-2  enter__form-field" id="colors">
                                <div class="enter__form-fieldname">Цвет светофора:</div>
                                <input type="radio" class="hidden" name="color-2" id="red2" value="1">
                                <input type="radio" class="hidden" name="color-2" id="yellow2" value="2">
                                <input type="radio" class="hidden" name="color-2" id="green2" value="3">
                                <label for="red2" class="enter__form-control">Красный</label>
                                <label for="yellow2" class="enter__form-control">Желтый</label>
                                <label for="green2" class="enter__form-control">Зеленый</label>
                            </div>
                            <div class="ml-1  enter__form-fieldname">Север - Восток / Юг - Запад</div>
                            <div class="ml-2  enter__form-field" id="colors">
                                <div class="enter__form-fieldname">Цвет светофора:</div>
                                <input type="radio" class="hidden" name="color-3" id="red3" value="1">
                                <input type="radio" class="hidden" name="color-3" id="yellow3" value="2">
                                <input type="radio" class="hidden" name="color-3" id="green3" value="3">
                                <label for="red3" class="enter__form-control">Красный</label>
                                <label for="yellow3" class="enter__form-control">Желтый</label>
                                <label for="green3" class="enter__form-control">Зеленый</label>
                            </div>
```

```
    <div class="ml-1  enter__form-fieldname">Восток - Юг / Запад - Север</div>
    <div class="ml-2  enter__form-field" id="colors">
      <div class="enter__form-fieldname">Цвет светофора:</div>
      <input type="radio" class="hidden" name="color-4" id="red4" value="1">
      <input type="radio" class="hidden" name="color-4" id="yellow4" value="2">
      <input type="radio" class="hidden" name="color-4" id="green4" value="3">
      <label for="red4" class="enter__form-control">Красный</label>
      <label for="yellow4" class="enter__form-control">Желтый</label>
      <label for="green4" class="enter__form-control">Зеленый</label>
    </div>
  </div>
  <div class="enter__form-field">
    <div class="enter__form-fieldname">Период времени:</div>
    <select class="enter__form-dropdown" name="period" id="period">
      <option value="1">Ночь</option>
      <option value="2">День</option>
      <option value="3">Час Пик</option>
    </select>
  </div>
  <div class="enter__form-intervals-wrap">
    <div class="enter__form-field  js-intervals  enter__form-field--hidden" id="intervals-1">
      <div class="enter__form-fieldname">Интервал:</div>
      <input type="radio" class="hidden" name="night-interval" value="1" id="t1-1">
      <input type="radio" class="hidden" name="night-interval" value="2" id="t2-1">
      <input type="radio" class="hidden" name="night-interval" value="3" id="t3-1">
      <input type="radio" class="hidden" name="night-interval" value="4" id="t4-1">
      <label for="t1-1" class="enter__form-control">С 0 с по 20 с</label>
      <label for="t2-1" class="enter__form-control">С 20 с по 30 с</label>
      <label for="t3-1" class="enter__form-control">С 30 с по 50 с</label>
      <label for="t4-1" class="enter__form-control">С 50 по 60 с</label>
    </div>
    <div class="enter__form-field  js-intervals  enter__form-field--hidden" id="intervals-2">
      <div class="enter__form-fieldname">Интервал:</div>
      <input type="radio" class="hidden" name="day-interval" value="1" id="t1-2">
      <input type="radio" class="hidden" name="day-interval" value="2" id="t2-2">
      <input type="radio" class="hidden" name="day-interval" value="3" id="t3-2">
      <input type="radio" class="hidden" name="day-interval" value="4" id="t4-2">
      <input type="radio" class="hidden" name="day-interval" value="5" id="t5-2">
      <input type="radio" class="hidden" name="day-interval" value="6" id="t6-2">
      <input type="radio" class="hidden" name="day-interval" value="7" id="t7-2">
      <input type="radio" class="hidden" name="day-interval" value="8" id="t8-2">
      <label for="t1-2" class="enter__form-control">С 0 по 15 с</label>
      <label for="t2-2" class="enter__form-control">С 15 по 25 с</label>
      <label for="t3-2" class="enter__form-control">С 25 по 55 с</label>
      <label for="t4-2" class="enter__form-control">С 55 по 65 с</label>
      <label for="t5-2" class="enter__form-control">С 65 по 80 с</label>
      <label for="t6-2" class="enter__form-control">С 80 по 90 с</label>
      <label for="t7-2" class="enter__form-control">С 90 по 110 с</label>
      <label for="t8-2" class="enter__form-control">С 110 по 120 с</label>
    </div>
    <div class="enter__form-field  js-intervals  enter__form-field--hidden" id="intervals-3">
      <div class="enter__form-fieldname">Интервал:</div>
      <input type="radio" class="hidden" name="rush-hour-interval" value="1" id="t1-3">
      <input type="radio" class="hidden" name="rush-hour-interval" value="2" id="t2-3">
      <input type="radio" class="hidden" name="rush-hour-interval" value="3" id="t3-3">
      <input type="radio" class="hidden" name="rush-hour-interval" value="4" id="t4-3">
      <label for="t1-3" class="enter__form-control">С 0 по 30 с</label>
      <label for="t2-3" class="enter__form-control">С 30 по 40 с</label>
      <label for="t3-3" class="enter__form-control">С 40 по 50 с</label>
```

```
                    <label for="t4-3" class="enter__form-control">С 50 по 60 с</label>
                  </div>
                </div>
                <div class="enter__form-field">
                  <input type="submit" class="enter__form-button" value="Добавить элемент" name="add-
object">
                </div>
                <div class="enter__form-field">
                  <input type="submit" class="enter__form-button" value="Показать введенные элементы"
                      name="show-objects">
                </div>
              </fieldset>
            </form>
          </div>
        </div>
        <div class="show">
          <button class="show__toggle" id="show-toggle">Показать меню</button>
          <div class="show__inner" id="show-block">
            <form class="show__form" action="index.php" method="POST">
              <input class="btn" type="submit" value="Сформировать системы данных" name="create-data-
system">
              <input class="btn" type="submit" value="Сформировать систему с поведением"
                  name="create-behavior-systems">
              <input class="btn" type="submit" value="Сформировать порождающую систему с поведением"
                  name="create-Fgb">
              <input class="btn" type="submit" value="Сформировать направленные системы с поведением"
                  name="create-Fb">
              <input class="btn" type="submit" value="Сформировать содержательные маски" name="create-
mask">
              <input class="btn" type="submit" value="Сформировать ST-Систему" name="create-st-system">
            </form>
          </div>
        </div>
      </section>
      <div class="show-canvas">
        <?php
        /**
         * Show Elements in cookie
        */
        if (isset($_POST["show-objects"])) {
          echo "<table class='show__table'>";
          echo "<tr>";
          echo "<td>Интервалы</td>";
          echo "<td>Север - Юг / Юг - Север</td>";
          echo "<td>Восток - Запад / Запад - Восток</td>";
          echo "<td>Север - Восток / Юг - Запад</td>";
          echo "<td>Восток - Юг / Запад - Север</td>";
          echo "</tr>";
          for ($i = 1; $i <= 3; $i++) {
            if ($i == 2) {
              for ($j = 1; $j <= 8; $j++) { //день содержит 8 интервалов
                echo "<tr>";
                if (isset($_COOKIE["interval" . $i . "" . $j])) {
                  if ($j == 1)
                    echo "<td>День: С 0 с по 15 с</td>";
                  if ($j == 2)
                    echo "<td>День: С 15 с по 25 с</td>";
                  if ($j == 3)
                    echo "<td>День: С 25 с по 55 с</td>";
```

```
            if ($j == 4)
               echo "<td>День: С 55 с по 65 с</td>";
            if ($j == 5)
               echo "<td>День: С 65 с по 80 с</td>";
            if ($j == 6)
               echo "<td>День: С 80 с по 90 с</td>";
            if ($j == 7)
               echo "<td>День: С 90 с по 110 с</td>";
            if ($j == 8)
               echo "<td>День: С 110 с по 120 с</td>";
            foreach ($_COOKIE["interval" . $i . "" . $j] as $name => $value) {
               $name = htmlspecialchars($name);
               $value = htmlspecialchars($value);
               if ($name != "interval_value") {
                  if (substr($name, 0, 5) === 'color') {
                     if ((int)$value === 1)
                        echo "<td>Красный</td>";
                     if ((int)$value === 2)
                        echo "<td>Желтый</td>";
                     if ((int)$value === 3)
                        echo "<td>Зеленый</td>";
                  }
               }
            }
            $keptAmount += 1;
         }
         echo "</tr>";
      }
   } else {
      for ($j = 1; $j <= 4; $j++) { //остальные периоды - 4-е
         echo "<tr>";
         if (isset($_COOKIE["interval" . $i . "" . $j])) {
            if ($i == 1 && $j == 1)
               echo "<td>Ночь: С 0 с по 20 с</td>";
            if ($i == 1 && $j == 2)
               echo "<td>Ночь: С 20 с по 30 с</td>";
            if ($i == 1 && $j == 3)
               echo "<td>Ночь: С 30 с по 50 с</td>";
            if ($i == 1 && $j == 4)
               echo "<td>Ночь: С 50 с по 60 с</td>";
            if ($i == 3 && $j == 1)
               echo "<td>Час Пик: С 0 с по 30 с</td>";
            if ($i == 3 && $j == 2)
               echo "<td>Час Пик: С 30 с по 40 с</td>";
            if ($i == 3 && $j == 3)
               echo "<td>Час Пик: С 40 с по 50 с</td>";
            if ($i == 3 && $j == 4)
               echo "<td>Час Пик: С 50 с по 60 с</td>";
            foreach ($_COOKIE["interval" . $i . "" . $j] as $name => $value) {
               $name = htmlspecialchars($name);
               $value = htmlspecialchars($value);
               if ($name != "interval_value") {
                  if (substr($name, 0, 5) === 'color') {
                     if ((int)$value === 1)
                        echo "<td>Красный</td>";
                     if ((int)$value === 2)
                        echo "<td>Желтый</td>";
                     if ((int)$value === 3)
                        echo "<td>Зеленый</td>";
```

```
                    }
                }
            }
            $keptAmount += 1;
        }
        echo "</tr>";
    }
  }
 }
}
/**
 * Build data system
*/
if (isset($_POST["create-data-system"])) {
  printDataSystem($mysqli->query("SELECT * FROM `intervals`"));
  printSemantiqueDataSystem($mysqli->query("SELECT * FROM `intervals`"));
}

/**
 * Build ST-System
*/
if (isset($_POST["create-st-system"])) {
  try {
    $queryResult = $mysqli->query("SELECT `color_1_math`, `color_2_math`, `color_3_math`, `color_4_math`
     FROM `intervals`");
    $StSystem = new StSystem($queryResult);
    $StSystem->build();
  } catch (\Exception $exception) {
    echo "Возникла ошибка при построении системы!";
    echo $exception->getMessage();
  }
}
if (isset($_POST["create-behavior-systems"])) {
  $queryResult = $mysqli->query("SELECT `color_1_math`, `color_2_math`, `color_3_math`, `color_4_math`
     FROM `intervals`");
  if ($queryResult != false) {
    $temp = array();
    $i = 0;
    $row = $queryResult->fetch_assoc();
    do {
      foreach ($row as $k => $v) {
        if ($i == 4) break;
        $temp[$k] = array();
        $i++;
      }
      foreach ($row as $k => $v) {
        switch ($k) {
          case "color_2_math":
          case "color_3_math":
          case "color_4_math":
          case "color_1_math":
            $temp[$k][] = $v - 1;
            break;
        }
      }
    } while ($row = $queryResult->fetch_assoc());
    echo "<h2>Система с поведением M<sup>1</sup> c q = (0, 1)</h2>";
    RenderBehaviorSystem($temp, 2);
  } else exit("<br>Не удалось получить данные<br>");
```

```
}
if (isset($_POST["create-Fgb"])) {
   $queryResult = $mysqli->query("SELECT `color_1_math`, `color_2_math`, `color_3_math`, `color_4_math`
       FROM `intervals`");
   if ($queryResult != false) {
      $temp = array();
      $i = 0;
      $row = $queryResult->fetch_assoc();
      do {
         foreach ($row as $k => $v) {
            if ($i == 4) break;
            $temp[$k] = array();
            $i++;
         }
         foreach ($row as $k => $v) {
            switch ($k) {
               case "color_2_math":
               case "color_3_math":
               case "color_4_math":
               case "color_1_math":
                  $temp[$k][] = $v - 1;
                  break;
            }
         }
      } while ($row = $queryResult->fetch_assoc());
      echo "<div class='df  df-sb'>";
      echo "<div>";
      echo "<h2>Нейтральная порождающая система с поведением</h2>";
      show_parageusia_system($temp, 1);
      echo "</div>";
   } else exit("<br>Не удалось получить данные<br>");
}
if (isset($_POST["create-Fb"])) {
   $queryResult = $mysqli->query("SELECT `color_1_math`, `color_2_math`, `color_3_math`, `color_4_math`
       FROM `intervals`");
   if ($queryResult != false) {
      $temp = array();
      $i = 0;
      $row = $queryResult->fetch_assoc();
      do {
         foreach ($row as $k => $v) {
            if ($i == 4) break;
            $temp[$k] = array();
            $i++;
         }
         foreach ($row as $k => $v) {
            switch ($k) {
               case "color_1_math":
                  $temp[$k][] = $v;
                  break;
               case "color_2_math":
                  $temp[$k][] = $v;
                  break;
               case "color_3_math":
                  $temp[$k][] = $v;
                  break;
               case "color_4_math":
                  $temp[$k][] = $v;
                  break;
```

```
                }
            }
        } while ($row = $queryResult->fetch_assoc());
        echo "<div class='df'>";
        buildGuideSystem($temp, 2);
        echo "</div>";
    } else exit("<br>Не удалось получить данные<br>");
}
function printDataSystem($queryResult)
{
    echo "<button class='btn' id='show-sys-1'>Направленная система данных с четкими данными без семантики</button>";
    $temp = array();
    $i = 0;
    $row = $queryResult->fetch_assoc();
    do {
        foreach ($row as $k => $v) {
            if ($i == 4) break;
            $temp[$k] = array();
            $i++;
        }
        foreach ($row as $k => $v) {
            switch ($k) {
                case "id":
                    $temp[$k][] = "<b>" . $v . "</b>";
                    break;
                case "color_1_math":
                case "color_3_math":
                case "color_4_math":
                case "color_2_math":
                    $temp[$k][] = (int)$v - 1;
                    break;
            }
        }
    } while ($row = $queryResult->fetch_assoc());
    echo '<div class="show-canvas">';
    echo "<table class='show__table' id='data-sys-1'>";
    foreach ($temp as $k => $v) {
        echo "<tr>";
        foreach ($temp[$k] as $key => $value) {
            echo "<td>" . $value . "</td>";
        }
        echo "</tr>";
    }
    echo "</table>";
    echo '</div>';
}
function printSemantiqueDataSystem($queryResult)
{
    echo "<button class='btn' id='show-sys-2'>Направленная система данных с четкими данными с семантикой</button>";
    $temp = array();
    $i = 0;
    $row = $queryResult->fetch_assoc();
    do {
        foreach ($row as $k => $v) {
            if ($i = 4) break;
            $temp[$k] = array();
            $i++;
```

```
        }
        foreach ($row as $k => $v) {
            switch ($k) {
                case "name":
                    $temp[$k][] = "<b>" . $v . "</b>";
                    break;
                case "color_1":
                    $temp[$k][] = $v;
                    break;
                case "color_2":
                    $temp[$k][] = $v;
                    break;
                case "color_3":
                    $temp[$k][] = $v;
                    break;
                case "color_4":
                    $temp[$k][] = $v;
                    break;
            }
        }
    } while ($row = $queryResult->fetch_assoc());
    echo '<div class="show-canvas">';
    echo "<table class='show__table' id='data-sys-2'>";
    foreach ($temp as $k => $v) {
        echo "<tr>";
        foreach ($temp[$k] as $key => $value) {
            echo "<td>" . $value . "</td>";
        }
        echo "</tr>";
    }
    echo "</table>";
    echo '</div>';
}
function RenderBehaviorSystem($temp, $c)
{
    echo "<table class='show__table  show__table--pm'>";
    foreach ($temp as $k => $v) {
        $count_arr = count($temp[$k]);
    }
    $count_arr_k = count($temp);
    $flag = 0;
    $j = 0;
    $s = 0;
    $count = ($count_arr - $c);
    $f = 1 / $count;
    echo "<tr>";
    while ($s <= $c * $count_arr_k) {
        if ($s == $c * $count_arr_k) {
            echo "<th>f<sub>B</sub>(C)</th>";
            break;
        }
        echo "<th>S" . ($s + 1) . "</th>";
        $s++;
    }
    echo "</tr>";
    $p = 0;
    $oka = array();
    for ($i = 0; $i < $c * ($count_arr - ($c - 1)); $i++) {
        //echo "<td><b>".$i."</b></td>";
```

```
    if ($i % $c == 0) {
        if ($flag != 0) {
            //echo "<td><b>1</b></td></tr>";
            $flag = 0;
            $j = $j - ($c - 1);
        }
        //echo "<tr>";
        $flag = 1;
    }
    $l = "";
    foreach ($temp as $key => $value) {
        $l .= $temp[$key][$j];
        //echo "<td>".$temp[$key][$j]."</td>";
    }
    $oka[$p] = $l;
    //echo "<td>".$oka[$p]."</td>";
    $j++;
    if ($i % 2 != 0) {
        $p++;
    }
}
//echo $i;
//echo $p;
$str_temp = "";
$flag = 0;
$j = 0;
$s = 0;
for ($i = 0; $i < $c * ($count_arr - ($c - 1)); $i++) {
    $Pf = 1;
    //echo "<td><b>".$i."</b></td>";
    if ($i % $c == 0) {
        if ($i != 0) {
            for ($o = 0; $o < count($oka); $o++) {
                //echo $oka[$o];
                if ($str_temp == $oka[$o])
                    $Pf++;
            }
            //$Pf *= 2;
            $Pf = $Pf / $count;
        }
        if ($flag != 0) {
            echo "<td><b>" . $Pf . "</b></td></tr>";
            $flag = 0;
            $j = $j - ($c - 1);
        }
        $str_temp = "";
        echo "<tr>";
        $flag = 1;
    }
    foreach ($temp as $key => $value) {
        $str_temp .= $temp[$key][$j];
        echo "<td>" . $temp[$key][$j] . "</td>";
    }
    if ($i % 2 != 0) {
        //echo "<td>".$str_temp."</td>";
        //echo "<td>".$str_temp."</td>";
    }
    if ($i == $c * ($count_arr - ($c - 1)) - 1) {
        if ($i != 0) {
```

```
                for ($o = 0; $o < count($oka); $o++) {
                    //echo $oka[$o];
                    if ($str_temp == $oka[$o])
                        $Pf++;
                }
                //$Pf *= 2;
                $Pf = $Pf / $count;
            }
            echo "<td><b>" . $Pf . "</b></td></tr>";
        }
        $j++;
    }
    echo "</table>";
}
function show_parageusia_system($temp, $c)
{
    echo "<div class='df'>";
    echo "<div class='fe-l; margin-right:;'>";
    echo "<h3>Порождающая система : </h3>";
    echo "<table class='show__table'>";
    //print_r($temp);
    foreach ($temp as $k => $v) {
        $count_arr = count($temp[$k]);
    }
    $count_arr_k = count($temp);
    $flag = 0;
    $j = 0;
    $s = 0;
    $count = ($count_arr - $c);
    echo "<tr>";
    while ($s <= $c * $count_arr_k) {
        if ($s == $c * $count_arr_k) {
            echo "<th>f<sub>B</sub>(g_)</th>";
            break;
        }
        echo "<th>S" . ($s + 1) . "</th>";
        $s++;
    }
    echo "</tr>";
    $p = 0;
    $oka = array();
    for ($i = 0; $i < $c * ($count_arr - ($c - 1)); $i++) {
        //echo "<td><b>".$i."</b></td>";
        if ($i % $c == 0) {
            if ($flag != 0) {
                //echo "<td><b>1</b></td></tr>";
                $flag = 0;
                $j = $j - ($c - 1);
            }
            //echo "<tr>";
            $flag = 1;
        }
        $l = "";
        foreach ($temp as $key => $value) {
            $l .= $temp[$key][$j];
            //echo "<td>".$temp[$key][$j]."</td>";
        }
//          $oka[$p] .= $l;
        $oka[$p] = $l;
```

```
    //echo "<td>".$oka[$p]."</td>";
    $j++;
    $p++;
}
//print_r($oka);
$flag = 0;
$j = 0;
for ($i = 0; $i < $c * $count; $i++) {
    $Pf = 0;
    if ($i % $c == 0) {
        if ($flag != 0) {
            if ($i != 0) {
                for ($o = 0; $o < count($oka); $o++) {
                    //echo $oka[$o];
                    if ($str_temp == $oka[$o])
                        $Pf++;
                }
                //$Pf *= 2;
                $Pf = $Pf / $count;
            }
            echo "<td><b>" . $Pf . "</b></td></tr>";
            $flag = 0;
            $j = $j - ($c - 1);
        }
        echo "<tr>";
        $flag = 1;
        $str_temp = "";
    }
    foreach ($temp as $key => $value) {
        $str_temp .= $temp[$key][$j];
        echo "<td>" . $temp[$key][$j] . "</td>";
    }
    if ($i == $c * ($count_arr - $c) - 1) {
        if ($i != 0) {
            for ($o = 0; $o < count($oka); $o++) {
                //echo $oka[$o];
                if ($str_temp == $oka[$o])
                    $Pf++;
            }
            //$Pf *= 2;
            $Pf = $Pf / $count;
        }
        echo "<td><b>" . $Pf . "</b></td></tr>";
    }
    $j++;
}
echo "</table>";
echo "</div>";
echo "<div class='fe-r'>";
echo "<h3>Порождаемая система : </h3>";
echo "<table class='show__table'>";
//print_r($temp);
$p = 0;
$oka_sec = array();
$j = $c;
$flag = 0;
for ($i = 0; $i < $count; $i++) {
    if ($i % ($c / $c) == 0) {
        if ($flag != 0) {
```

```
                $flag = 0;
                $j = $j - ($c / $c - 1);
            }
            $flag = 1;
        }
        $l = "";
        foreach ($temp as $key => $value) {
            $l .= $temp[$key][$j];
            //echo "<td>".$temp[$key][$j]."</td>";
        }
//          $oka_sec[$p] .= $l;
        $oka_sec[$p] = $l;
        //echo "<td>".$oka[$p]."</td>";
        $j++;
        $p++;
    }
    //print_r($oka);
    $j = $c;
    $flag = 0;
    echo "<tr>";
    while ($s <= 2 * $c * $count_arr_k - ($c - 1) * $count_arr_k) {
        if ($s == 2 * $c * $count_arr_k - ($c - 1) * $count_arr_k) {
            echo "<th>f<sub>B</sub>(g)</th>";
            break;
        }
        echo "<th>S" . ($s + 1) . "</th>";
        $s++;
    }
    echo "</tr>";
    for ($i = 0; $i < $count; $i++) {
        $Pf = 0;
        if ($i % ($c / $c) == 0) {
            if ($flag != 0) {
                if ($i != 0) {
                    for ($o = 0; $o < count($oka_sec); $o++) {
                        //echo $oka_sec[$o];
                        if ($str_temp == $oka_sec[$o])
                            $Pf++;
                    }
                    //$Pf *= 2;
                    $Pf = $Pf / $count;
                }
                echo "<td><b>" . $Pf . "</b></td></tr>";
                $flag = 0;
                $j = $j - ($c / $c - 1);
            }
            echo "<tr>";
            $flag = 1;
        }
        $str_temp = 0;
        foreach ($temp as $key => $value) {
            $str_temp .= $temp[$key][$j];
            echo "<td>" . $temp[$key][$j] . "</td>";
        }
        if ($i == ($count_arr - $c) - 1) {
            if ($i != 0) {
                for ($o = 0; $o < count($oka); $o++) {
                    //echo $oka[$o];
                    if ($str_temp == $oka_sec[$o])
```

```
                $Pf++;
            }
            //$Pf *= 2;
            $Pf = $Pf / $count;
        }
        echo "<td><b>" . $Pf . "</b></td></tr>";
    }
    $j++;
  }
  echo "</table>";
  echo "</div>";
}
function buildGuideSystem($temp, $c)
{
  echo "<div class='df'>";
  echo "<div>";
  echo "<table class='show__table'>";
  //print_r($temp);
  foreach ($temp as $k => $v) {
    $count_arr = count($temp[$k]);
  }
  $count_arr_k = count($temp);
  $flag = 0;
  $j = 0;
  $s = 0;
  $l = 0;
  $buffer = array();
  $generated = array();
  echo "<tr>";
  while ($s <= $c * $count_arr_k) {
    if ($s == $c * $count_arr_k) {
      echo "<th>f<sub>B</sub>(C)</th>";
      break;
    }
    /*if(($s+1)%$count_arr_k == 0){
      $s++;
      continue;
    }*/
    echo "<th>S" . ($s + 1) . "</th>";
    $s++;
  }
  echo "</tr>";
  for ($i = 0; $i < $c * ($count_arr - ($c - 1)); $i++) {
    if ($i % $c == 0) {
      if ($i != 0) {
        echo "<td><b>1</b></td></tr>";
        $j = $j - ($c - 1);
      }
      echo "<tr>";
    }
    foreach ($temp as $key => $value) {
      echo "<td>" . $temp[$key][$j] . "</td>";
    }
    if ($i == $c * ($count_arr - ($c - 1)) - 1) echo "<td><b>1</b></td></tr>";
    $j++;
  }
  echo "</table>";
  echo "</div>";
  echo "<div>";
```

```
    echo "<table class='show__table'>";
    echo "<tr>";
    $s = $count_arr_k - 1;
    while ($s <= $c * $count_arr_k) {
        echo "<th>S" . ($s + 1) . "</th>";
        if ($s + 1 == $c * $count_arr_k) {
            echo "<th>f<sub>B</sub>(C)</th>";
            break;
        }
        $s += $count_arr_k;
    }
    echo "</tr>";
    $count_gen = count($generated);
    $j = 0;
    for ($i = 0; $i < $c * ($count_arr - ($c - 1)); $i++) {
        if ($i % $c == 0) {
            if ($i != 0) {
                echo "<td><b>1</b></td></tr>";
                $j = $j - ($c - 1);
            }
            echo "<tr>";
        }
        echo "<td>" . $generated[$j] . "</td>";
        if ($i == $c * ($count_arr - ($c - 1)) - 1) echo "<td><b>1</b></td></tr>";
        $j++;
    }
    echo "</table>";
    echo "</div>";
    echo "</div>";
}
function buildGuideSystem_part_2($temp, $c)
{
    echo "<div class='df'>";
    echo "<div>";
    echo "<table class='show__table'>";
    //print_r($temp);
    foreach ($temp as $k => $v) {
        $count_arr = count($temp[$k]);
    }
    $count_arr_k = count($temp);
    $j = 0;
    $s = 0;
    $l = 0;
    $buffer = array();
    $generated = array();
    echo "<tr>";
    while ($s <= $c * $count_arr_k) {
        if ($s == $c * $count_arr_k) {
            echo "<th>f<sub>B</sub>(C)</th>";
            break;
        }
        if (($s + 1) % $count_arr_k == 0) {
            $s++;
            continue;
        }
        echo "<th>S" . ($s + 1) . "</th>";
        $s++;
    }
    echo "</tr>";
```

```
for ($i = 0; $i < $c * ($count_arr - ($c - 1)); $i++) {
   if ($i % $c == 0) {
      if ($i != 0) {
         echo "<td><b>1</b></td></tr>";
         $j = $j - ($c - 1);
      }
      echo "<tr>";
   }
   foreach ($temp as $key => $value) {
      if ($key === "color_4_math") {
         $generated[] = $temp[$key][$j];
      } else echo "<td>" . $temp[$key][$j] . "</td>";
   }
   if ($i == $c * ($count_arr - ($c - 1)) - 1) echo "<td><b>1</b></td></tr>";
   $j++;
}
echo "</table>";
echo "</div>";
$j = 0;
$l = 0;
$s = $count_arr_k - 1;
$count_gen = count($generated);
$gen_g = array();
$gen_g_ = array();
for ($i = 0; $i < $count_gen; $i++) {
   if ($i != 0 && ($i + 1) % $c == 0)
      $gen_g[] = $generated[$i];
   else
      $gen_g_[] = $generated[$i];
}
echo "<div>";
echo "<table class='show__table'>";
echo "<tr>";
while ($s <= $c * $count_arr_k) {
   if ($s + 1 != $c * $count_arr_k)
      echo "<th>S" . ($s + 1) . "</th>";
   if ($s + 1 == $c * $count_arr_k) {
      echo "<th>f<sub>B</sub>(C)</th>";
      break;
   }
   $s += $count_arr_k;
}
echo "</tr>";
if ($c == 2) {
   for ($i = 0; $i < count($gen_g_) - 1; $i++) {
      if ($gen_g_[$i] == "null") continue;
      $flag = 1;
      for ($j = $i + 1; $j < count($gen_g_); $j++) {
         if ($gen_g_[$i] == $gen_g_[$j]) {
            $flag++;
            $gen_g_[$j] = "null";
         }
      }
      echo "<tr><td>" . $gen_g_[$i] . "</td><td>" . $flag . "</td></tr>";
   }
}
if ($c == 3) {
   for ($i = 0; $i < count($gen_g_) - 1; $i += 2) {
      if ($gen_g_[$i] == "null") continue;
```

```
                $flag = 1;
                for ($j = $i + 2; $j < count($gen_g_); $j += 2) {
                    if ($gen_g_[$i] == $gen_g_[$j] && $gen_g_[$i + 1] == $gen_g_[$j + 1]) {
                        $flag++;
                        $gen_g_[$j] = "null";
                        $gen_g_[$j + 1] = "null";
                    }
                }
                echo "<tr><td>" . $gen_g_[$i] . "</td><td>" . $gen_g_[$i + 1] . "</td><td>" . $flag . "</td></tr>";
            }
        }
        echo "</table>";
        echo "</div>";
        echo "<div'>";
        echo "<table class='show__table'>";
        echo "<tr>";
        while ($s <= $c * $count_arr_k) {
            if ($s + 1 == $c * $count_arr_k) {
                echo "<th>S" . ($s + 1) . "</th>";
                echo "<th>f<sub>B</sub>(C)</th>";
                break;
            }
            $s += $count_arr_k;
        }
        echo "</tr>";
        for ($i = 0; $i < count($gen_g) - 1; $i++) {
            if ($gen_g[$i] == "null") continue;
            $flag = 1;
            for ($j = $i + 1; $j < count($gen_g_); $j++) {
                if ($gen_g[$i] == $gen_g[$j]) {
                    $flag++;
                    $gen_g[$j] = "null";
                }
            }
            echo "<tr><td>" . $gen_g[$i] . "</td><td>" . $flag . "</td></tr>";
        }
        echo "</table>";
        echo "</div>";
        echo "</div>";
    }
    ?>
  </div>
</div>
</body>
<script src="js/script.js"></script>
</html>

<?php
namespace Systems;

class StSystem
{
    protected $data = [];
    protected $system = [];
    protected $range = [];
    protected $transition = 0;
    protected $keys = [
        'color_1_math',
        'color_2_math',
```

```
    'color_3_math',
    'color_4_math',
];

/**
 * StSystem constructor.
 * @param $sourceData
 * @throws \Exception
 */
public function __construct($sourceData)
{
    $this->pullData($sourceData);
}
protected function makeRange()
{
    $variablesExpected = 4;
    $valuesExpected = 3;
    $preparedRange = array_map(function ($element) use ($variablesExpected) {
        return substr("000$element", -$variablesExpected);
    }, range(0, 2222));
    $finalRange = array_filter($preparedRange, function ($element) use ($valuesExpected, $variablesExpected) {
        return sizeof(array_filter(str_split($element), function ($digit) use ($valuesExpected) {
                return in_array((int)$digit, range(0, $valuesExpected - 1));
            })) === $variablesExpected;
    });
    $this->range = array_unique($finalRange);
}
protected function pullData($sourceData)
{
    if (!$sourceData) {
        throw new \Exception('Data is empty');
    }
    $i = 0;
    $row = $sourceData->fetch_assoc();
    do {
        foreach ($row as $k => $v) {
            if ($i == 4) break;
            $this->data[$k] = array();
            $i++;
        }
        foreach ($row as $k => $v) {
            if (in_array($k, $this->keys)) {
                $this->data[$k][] = $v;
            }
        }
    } while ($row = $sourceData->fetch_assoc());
    $this->mutateSourceData();
}
protected function mutateSourceData()
{
    $mutatedSource = [];
    foreach ($this->keys as $key) {
        foreach ($this->data[$key] as $index => $item) {
            if (!isset($mutatedSource[$index])) {
                $mutatedSource[$index] = $item - 1;
            } else {
                $mutatedSource[$index] .= ($item - 1);
            }
        }
```

```
    }
    $this->data = $mutatedSource;
}
protected function findFGs()
{
    $this->makeRange();
    $limit = sizeof($this->data);
    foreach ($this->range as $range) {
        $this->system[$range] = [];
        foreach ($this->range as $deepRange) {
            $this->system[$range][$deepRange] = 0;
            $keys = array_keys($this->data, $range);
            if (empty($keys)) {
                continue;
            }
            foreach ($keys as $key) {
                if (($key + 1) > ($limit - 1)) {
                    continue;
                }
                if($this->data[$key + 1] === $deepRange) {
                    $this->transition++;
                }
                $this->system[$range][$deepRange] += (int)($this->data[$key + 1] === $deepRange);
            }
        }
    }
    $this->calculateFGs();
}
protected function calculateFGs()
{
    $limit = sizeof($this->data) - 2;
    foreach ($this->system as $firstLevel => $item) {
        foreach ($item as $secondLevel => $coincide) {
            $this->system[$firstLevel][$secondLevel] = $this->system[$firstLevel][$secondLevel] / $limit;
        }
    }
}
protected function render()
{
    if (empty($this->system)) {
        echo '<h1>ST-система пуста</h1>';
        return;
    }
    $fields = array_keys($this->system);
    echo '<p>Переходов совершено:' . $this->transition. '</p>';
    echo "<table class='show__table  show__table--pm'>";
    foreach ($fields as $index => $field) {
        if ($index === 0) {
            echo '<tr><th></th>';
            foreach ($fields as $innerField) {
                echo '<th>' . $innerField . '</th>';
            }
            echo '</tr>';
        }
        echo '<tr><td>' . $field . '</td>';
        foreach ($this->system[$field] as $item) {
            echo '<td>' . $item . '</td>';
        }
        echo '</tr>';
```

```
        }
        echo "</table>";
    }
    public function build()
    {
        $this->findFGs();
        $this->render();
    }
}
```