ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
МІНІСТЕРСТВА ОСВІТИ І НАУКИ УКРАЇНИ
Кафедра комп'ютерних інтелектуальних систем та мереж

ПЕТРОВ Олександр Віталійович

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

ДОСЛІДЖЕННЯ МОДЕЛЕЙ ВЕРИФІКАЦІЇ ДЛЯ КОМУТАЦІЇ АГЕНТІВ МУЛЬТІАГЕНТНИХ СИСТЕМ

Спеціальність 123 – Комп'ютерна інженерія
Спеціалізація – Комп'ютерні системи та мережі

Керівник: Мартинюк Олександр Миколайович,
кандидат технічних наук, доцент

Одеса – 2021

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА СТУДЕНТА

Петров Олександр Віталійович
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) ДОСЛІДЖЕННЯ МОДЕЛЕЙ ВЕРИФІКАЦІЇ ДЛЯ КОМУТАЦІЇ АГЕНТІВ МУЛЬТІАГЕНТНИХ СИСТЕМ

керівник проекту (роботи) Мартинюк Олександр Миколайович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом ректора ОНПУ від "___" 2021 _ року №_____

2. Строк подання студентом проекту (роботи) .12.2021 р.

3. Вихідні дані до проекту (роботи) Специфікації протоколів прикладного рівня розподілених інформаційних систем та MAC

4. Зміст розрахунково-пояснювальної за писки (перелік питань, які потрібно розробити) Вступ 1. Дослідження властивостей і механізмів комутації у розподілених інформаційних системах. 2. Розробка і дослідження формальної моделі комутації. 3. Побудова базової процедури верифікації комутації. 4. реалізація верифікації комутацій. Висновок

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

1. Автоматна модель NAT. 2. Модель автоматної мережі. 3. Мережева модель верифікації. 4. Приклади мережевих операцій композиції. 5. Зовнішня мережева мутація фрагментів поведінки. 6. Мережева процедура верифікації комутації. 7. Фрагмент стадій еволюції та ієрархія моделей. 8. Об'єктно-реляційна БД комплексу верифікації. 9. Схема алгоритму поведінки комутації. 10. Схема інтерактивного алгоритму вхідного контролю. 11. Складність еволюційної верифікації комутацій.

6. Консультанти розділів проекту (роботи)

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

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
	Аналіз специфікацій розподілених інформаційних систем	01.10.21	
	Аналіз автоматних і мережевих моделей аналізу	01.10.21	
	Аналіз і автоматних мережевих методів аналізу	01.10.21	
	Побудова моделі верифікації	15.10.21	
	Побудова процедури верифікації	01.11.21	
	Реалізація структур і алгоритмів верифікації	15.11.21	
	Оцінка ревлізацій базових засобів верифікації	30.11.21	
	Підготовка пояснювальної записки дипломної роботы	09.12.21	

Студент _____ Петров О. В.
(підпис) (прізвище та ініціали) .

Керівник проекту (роботи) _____ Мартыник О.М_
(підпис) (прізвище та ініціали) .

Відомість кваліфікаційної роботи

№ рядка	Найменування	Кільк.	Примітка
1	Пояснювальна записка	75	
2	Автоматна модель NAT	1	
3	Модель автоматної мережі	1	
4	Мережева модель верифікації	1	
5	Приклади мережевиж операцій композиції	1	
6	Зовнішня мережева мутація фрагментів поведінки	1	
7	Мережева процедура верифікації комутації	1	
8	Фрагмент стадій еволюції та ієрархія моделей	1	
9	Об’єктно-реляційна БД комплексу верифікації	1	
10	Схема алгоритму поведінки комутації	1	
11	Складність еволюційної верифікації комутацій	1	
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			

Зм	Ар	№ докум	Підпис	Дата	АМДР. АМ161м.1111			
Розробив		Петров О.В.			Дослідження моделей верифікації для коммутації агентів мультіагентних систем	Літера	*Лист*	Листів
Перевірив		Мартинюк О.М.					1	
						ОНПУ ІКС *каф. КІСМ АМ-161м*		

АНОТАЦІЯ

Петров О. В. Дослідження моделей верифікації для комутації агентів мультіагентних систем – кваліфікаційна робота магістра. Одеса, 2021: 80 с., 9 рис., 1 табл., 28 джерел.

Робота присвячена вдосконаленню моделей верифікації для комутації агентів мультіагентних систем (МАС).

Метою дослідження є скорочення часу верифікації для комутації за рахунок розробки моделей верифікації із зменшеною обчислювальною складністю.

Об'єктом дослідження є процес аналізу верифікації для комутації як організація експериментів поведінки для автоматних моделей МАС.

Предметом дослідження є автоматні моделі верифікації для комутації, які використовуються для аналізу МАС у цілому.

Проведено аналіз сучасного стану методів аналізу для комутації, показано існування невирішених завдань досягнення прийнятної обчислювальної складності та довжини для реальних МАС. У вирішенні завдань використовується підхід, пропонований теорією експериментів з автоматами, на рівні автоматів з визначенням обчислювальної складності і довжини, часу аналізу для комутації МАС. Побудовано спеціальні моделі верифікації для комутації у вигляді автоматних підмереж для проявлення фрагментів поведінки, відносин і операцій.

У роботі запропоновані базові процедури та алгоритми, які використовуються для підготовки забезпечення верифікації для комутації. Застосування моделей і методів дозволяє формалізувати побудову засобів верифікації для комутації у контролі МАС. Проведено апробацію розроблених засобів в рамках навчального процесу кафедри комп'ютерних інтелектуальних систем і мереж ОНПУ.

АВТОМАТ, КОМУТАЦІЯ, ВЕРИФІКАЦІЯ, ЕКСПЕРИМЕНТ РОЗПІЗНАВАННЯ.

ANNOTATION

Petrov O. V. Research of the models of verification for commutation agents of multiagent systems - qualifying work of the magister. Odessa, 2021: 80 pages, 9 pictures, 1 tables, 28 references.

The aim of the study is to reduce the verification time for switching by developing verification models with reduced computational complexity.

The object of research is the process of verification analysis for switching as the organization of behavioral experiments for automated MAC models.

The subject of the study are automatic verification models for switching, which are used to analyze the MAC as a whole.

The analysis of the current state of analysis methods for switching is carried out, the existence of unsolved problems of achieving acceptable computational complexity and length for real MAS is shown. The approach proposed by the theory of experiments with automata at the level of automata with the definition of computational complexity and length, analysis time for MAC switching is used in solving problems. Special verification models for switching in the form of automatic subnets for the manifestation of fragments of behavior, relationships and operations are built.

The paper proposes basic procedures and algorithms used to prepare for verification for switching. The use of models and methods allows to formalize the construction of verification tools for switching in MAC control. Approbation of the developed means within the educational process of the department of computer intelligent systems and networks of ONPU is carried out.

AUTOMATA, COMMUTATION, VERIFICATION, RECOGNITION EXPERIMENT.

ЗМІСТ

ВСТУП

Актуальність. Аналіз й верифікація комутацій, які е необхідними базовими механизмами розподілених інформаційних систем (РІС), зокрема мультіагентних систем (МАС), має велику важливість у підтвердженні вірогідності їх функціонування. Теорія і практика діагностики розглядає прості комутації, як схемні логічні та електричні з'єднання, та складні, коли існують визначенні протокольні умови на формати даних й процедури системного та функціонального обміну. У будь якому випадку у межах розглядаємої системи для комутації визначається відповідна їй форсмальна модель обміну, що суттєва при удосконаленні відповідних методів верифікації й контролю комутацій. Комутація також пов'язана з необхідністю доступу і розпізнавання видалених, безпосередньо недосяжних фрагментів поведінки.

Відомі формальні моделі і методи, методологічні рішення мереже-розподіленого й наскрізного аналізу комутації для канального, мережевого, транспортного та прикладного рівнів. Відповідно сформувалися різно-рівневі математичні базиси для побудови елементів такого аналізу комутацій у комп'ютерних системах, у їх складі й РІС та МАС.

Дослідження моделей і методів верифікації комутації МАС підтверджують доцільність відповідних етапів й кроків у проектуванні, верифікації, конструюванні, налагодженні МАС та РІС.

У прихованої не наочної комутації, яка часто існує за умовами функціонування конкретних МАС, необхідне виконання спеціального аналітичного поведінкового аналізу і синтезу.

Завдання дослідження формальних моделей і методів для аналізу та верифікації комутацій з їх керованностю та наочностю представляють інтерес з точки зору розробки МАС, є актуальними і нетривіальними.

Мета роботи – підвищення повноти, точності і оперативності аналізу комутації МАС за рахунок дослідження і побудови відповідних формальних структурно-поведінкових автоматних моделей її верифікації.

Поставлені завдання. Для досягнення мети вирішені завдання:

- дослідження функціональних й логічних специфікацій моделей і методів аналізу для комутації PIC та MAC, ;
- побудови, аналізу і моделювання поведінкових автоматних моделей та базових кроків верифікації комутації MAC;
- програмно-алгоритмічної реалізації базових кроків та алгоритмів верифікації комутацій MAC.

Об'єктом дослідження є процес верифікації комутацій MAC, як організація взаємодій у композиції автоматних моделей.

Предметом дослідження є моделі верифікаці комутацій і методики її аналізу для MAC.

Методи аналізу, проектування і верифікації ґрунтуються на застосуванні булевої алгебри, теорії предикатів, експериментів з автоматами, контролю і діагнозу, використовуваних як математична база побудови моделей, теорій алгоритмів, об'єктно-компонентного програмування в якості методологічної бази побудови засобів верифікації комутацій MAC.

Наукова новизна результатів полягає в удосконаленні моделей верифікації комутації MAC, заснованих на автоматному підході. У магістерській роботі отримані результати:

- Удосконалено формальну модель верифікації комутації MAC, що заснована на автоматних експериментах, має особливості спеціальних автоматних структур ідентифікації та розпізнавання властивостей взаємодії, керованості та наочності поведінки у композиції автоматних моделей. Модель верифікації комутацій призначена для формального визначення умов організації експериментів розпізнавання комутаційної поведінки у автоматної композиції, яка модулює MAC та дозволяє скоротити час контролю для методів їх верифікації.

Практичне значення отриманих результатів полягає в розробці кроків верифікації комутацій для елементів інформаційної технології контролю MAC.

Застосування моделі формалізує вимоги до кроків виконання верифікації

комутацій MAC у обраному методі, знижає час контролю MAC, підвищити його повноту.

Результати роботи планується включити в навчальну інформційну систему кафедри, яка використовується в ході освоєння студентами спеціальних дисциплін.

Згодом для поглибленого навчально-практичного аналізу верифікації комутацій MAC передбачається продовжити дослідження в розподіленої інформаційної системи кафедри КІСМ.

Пояснювальна записка складається з вступу, розділу дослідження технологій і методів аналізу комутацій у контролі MAC, розділу розробки та дослідження моделей верифікації комутацій с завданням на дослідження, розділу розробки і дослідження процедури верифікації комутацій MAC, розділу реалізації і експериментів для моделей, висновків, бібліографії в складі 32 джерел.

1. ДОСЛІДЖЕННЯ ВЛАСТИВОСТЕЙ І МЕХАНІЗМІВ КОМУТАЦІЇ У РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

1.1. Сильнозв'язані засоби транспортно-мережевого рівня

1.1.1. Інтерфейс сокетів мережевого програмування

Багато компонентів, класів та бібліотек мережевого програмування використовують інтерфейс сокетів [1]. У разі критичного розміру (для серверів) використання будь-яких VCL-компонентів (Visual Component Library) [2] небажане. Програмування на нижчому рівні вимагає великих витрат, але надає більш повний контроль за роботою, що важливо для побудови гнучких та різнопланових мережевих додатків.

Останні нововведення доступні лише на рівні Winsock2.0 [3], побудова ефективних та високопродуктивних клієнт/серверних Windows-програм неможлива без хорошого знання всіх особливостей сокетного інтерфейсу.

Зазвичай застосовуються два популярні інтерфейси прикладного мережевого програмування (API – Application Programming Interface) для додатків, що використовують протоколи стека TCP/IP:

– інтерфейс сокетів (sockets API);

– інтерфейс транспортного рівня (TLI – Transport Layer Interface) [4].

Останній, розроблений в AT&T, називають XTI (X/Open Transport Interface. XTI є розширенням TLI.

У мережах існує два різні способи передачі даних.

Перший передбачає посилку пакетів даних від одного вузла іншому (декільком вузлам) без отримання підтвердження про доставку і без гарантії, що пакети, що передаються, отримані в правильній послідовності. Приклад – протокол UDP у мережах TCP/IP [5]. Переваги датаграмних протоколів полягають у високій швидкості передачі та можливості широкомовної передачі

даних, коли один вузол відправляє повідомлення, інші їх отримують одночасно, а також груповий передачі – мультикастингу.

Другий спосіб передачі даних передбачає створення каналу передачі між двома вузлами мережі. Канал створюється засобами датаграмних протоколів, наприклад, IP, доставка пакетів у каналі є гарантованою за рахунок транспортного протоколу. Пакети завжди доходять у цілісності, правильному порядку, швидкодія виходить нижче за рахунок посилки підтверджень. Прикладом протоколів, використовують канали зв'язку, може бути протокол TCP [5]. Інтерфейс NetBIOS [5] допускає передачу даних із використанням як датаграм, і каналів зв'язку.

У передачі даних на основі перелічених протоколів (крім NetBIOS) програма верхнього рівня застосовує сокет – кінцеву точку мережного з'єднання, для створення та роботи з яким використовується спеціальний програмний інтерфейс.

Інтерфейс прикладної програми (Application Programm Interface – API) – це набір функцій для розробки прикладних програм. Використовується інтерфейс сокетів TCP/IP (TCP/IP sockets) чи інтерфейс сокетів Берклі (Berkeley sockets). Сокети реалізують взаємодії партнерів з телекомунікацій та міжпроцесних взаємодій на одному комп'ютері.

У роботі із сокетами виділяються етапи:

– сокет створюється;
– налаштовується на заданий режим роботи;
– використовується для організації обміну;
– ліквідується.

Технологія сокетів підтримує роботу з будь-якими стеками протоколів, суміщені процедури введення/виводу, використання великої кількості сервіс-провайдерів, можливість групування сокетів, це дозволяє реалізувати пріоритети та багато іншого.

Мережева взаємодія – це процес передачі даних по мережі між двома або більше комп'ютерами або процесами, а сокет - кінцевий пункт передачі даних.

Коли програми використовують сокет, для них він є абстракцією, що представляє одне із закінчень мережевої взаємодії.

Сокет – логічне поняття, що об'єднує у собі кілька елементів різної природи – структури, покажчики, записи у таблицях, приймаючі і передають буфера тощо. буд. Функція створення сокету у будь-якій операційній системі вважається ресурсомісткою, особливо для серверів із великим потоком заявок обслуговування.

З'єднання в моделі сокетів означає, що кожна мережна програма створює свої сокети. Зв'язок між сокетами орієнтований або не орієнтований на з'єднання. Коли програмі потрібен сокет, вона формує його характеристики та звертається до API, запитуючи у системи його дескриптор.

1.1.2. Інтерфейс компонентного програмування

Компонентне програмування [6] розповсюджує класи в бінарному вигляді (не у вихідному коді) та надає доступ до методів класу через певні інтерфейси, що дає сумісність компіляторів та забезпечує зміну версій класів без перекомпіляції додатків.

Роль інтерфейсів в COM важливіша за роль посередника, Інтерфейси визначаються першими і задають семантику сервісу. Класи реалізують заданий інтерфейс, забезпечуючи поліморфізм на рівні.

Існують різні технології компонентного програмування – COM (DCOM, COM+), CORBA, .Net [6-8].

Підход, який використовується в COM. Компонент зберігає (у DLL, EXE) один чи кілька класів. Клієнт знає про клас його ідентифікатор та один або кілька інтерфейсів для доступу до реалізованих даним класом методів. У реєстрі системи зберігається інформація про розташування компонента з даним класом (локально або віддалено), що дозволяє системі прозоро для клієнта перенаправляти виклики методів до потрібного компонента та повертати результати. Так забезпечується виконання принципів компонентного програмування незалежності від мови програмування та прозорості розташування сервера для клієнта.

Особливості компонентного програмування

Інкапсуляція. У COM інкапсуляція перебуває в вищому рівні ніж ООП. Між клієнтом та реалізацією класу є інтерфейси – абстрактні базові класи без елементів даних, які є прямими нащадками лише одного іншого інтерфейсу. Реалізація методів інтерфейсу виконується у класі, який є нащадком даного та, можливо, інших інтерфейсів. Компілятори з урахуванням обмежень генерують код для викликів методів інтерфейсу клієнта. Клієнт без перекомпіляції може викликати методи нової версії класу (при збереженні інтерфейсу). Нова версія може мати розширену функціональність з допомогою додавання нових інтерфейсів. Нові інтерфейси можуть використовуватися новими клієнтами, старі клієнти продовжують працювати зі старими інтерфейсами, не знаючи нових.

Наслідування інтерфейсів. На відміну від ООП, технологія компонентного програмування не підтримує спадкування реалізації класу. Наслідують лише інтерфейси. Кожен інтерфейс отримує свій унікальний ідентифікатор і може реалізовуватись у різних класах, що включаються до різних компонентів. Новий інтерфейс може успадковувати раніше описані інтерфейси. Наприклад, в COM будь-який інтерфейс повинен успадковувати стандартний інтерфейс IUnknown. Наслідування інтерфейсу, зокрема, означає, що з реалізації методів нового інтерфейсу будуть реалізовані і всі методи, описані в наслідуваному інтерфейсі. У компонентному програмуванні розробник нового компонента не використовує вихідний код раніше реалізованого компонента, але може додати функціональність старого компонента до функціональності нового за допомогою одного з двох методів: контейнеризації, коли новий компонент включає (за допомогою успадкування) у свій інтерфейс старого компонента і реалізує його методи, викликаючи методи старого компонента (делегування). Старий компонент не знає, що він працює на користь іншого компонента, новий компонент не отримує нову функціональність задарма – він посередник між клієнтом та старим компонентом, що впливає на продуктивність.

Агрегація. З використанням цього підходу старий компонент знає у тому, що його використовує інший компонент. Новий (агрегуючий) компонент не працює посередником - виклики клієнта, які стосуються старого компоненту, прямують йому. Клієнт не помічає, йому здається, що він працює з одним новим компонентом.

Поліморфізм. Якщо описаний інтерфейс, будь-яке число класів можуть реалізовувати його будь-яким способом будь-якою мовою, що підтримує COM. Однак не повинна змінюватися семантика інтерфейсу, клієнту не важливо, хто і як реалізував цікавий для нього інтерфейс.

Бінарна вистава. Компоненти поширюються і застосовують у бінарному вигляді, тобто. у вигляді "чорного ящика", що вирішує проблеми ООП і дає нові можливості (використання різних мов програмування при реалізації компонентів та їх клієнтів).

Інфраструктура для розподілених програм. Багато з інфраструктури входить у архітектуру системи з компонентним програмуванням, багато що забезпечується як додаткові послуги. Для COM при розміщенні компонента та клієнта у різних процесах (на одному або на різних комп'ютерах) формується канал передачі даних для виклику методів, передачі параметрів та повернення результатів, додаткові сервіси COM+ – безпека, транзакції, балансування завантаження серверів, асинхронний доступ до компонентів, підтримка публікації та підписки на події тощо.

1.2. Слабопов'язані засоби прикладного рівня

1.2.1. Засоби комутації протоколу HTTP

HTTP [9] – це протокол отримання різних ресурсів (HTML-документів [10]). Протокол HTTP є основою обміну даними в Інтернеті. HTTP є протоколом клієнт-серверної взаємодії, що означає ініціювання запитів до сервера самим одержувачем, зазвичай веб-браузером. Отриманий документ може складатися з

різних піддокументів, що є частиною підсумкового документа (окремо отриманого тексту, опис структури документа, зображень, відео-файлів, скриптів, ...).

Клієнти та сервери обмінюються одиночними повідомленнями (не потоком). Повідомлення клієнта – запити, повідомлення сервера – відповіді.

HTTP рахунок розширюваності удосконалювався. HTTP – протокол прикладного рівня, використовує можливості іншого протоколу транспортного рівня – TCP (TLS) – для пересилання своїх повідомлень, будь-який надійний транспортний протокол може бути використаний для доставки повідомлень. Розширюваність дозволяє використовувати HTTP не тільки для отримання клієнтом гіпертекстових документів, зображень та відео, але й для передачі вмісту серверам, наприклад, за допомогою HTML-форм. HTTP також використовується для отримання частин документа під час оновлення веб-сторінок на запит (AJAX [11]).

Складові системи, засновані на HTTP

HTTP – це клієнт-серверний протокол, тобто запити надсилаються якоюсь однією стороною – учасником обміну, або проксі. Учасником крім веб-браузера може бути пошуковий робот, який подорожує для оновлення даних індексації веб-сторінок.

Запит надсилається серверу, який обробляє його та повертає відповідь. Між запитами та відповідями можуть бути посередники – проксі для різних операцій, що працюють як шлюзи або кеш, маршрутизатори, модеми тощо. Система рівнів взаємодії приховує посередників на мережному та транспортному рівнях. У системі рівнів HTTP займає найвищий прикладний рівень. Рівні мережі (представницький, сеансовий, транспортний, мережевий, канальний та фізичний) дозволяють організувати роботу PIC та діагностику проблем, але не потрібні для розуміння HTTP.

Клієнт – учасник обміну

Учасник обміну клієнт створює запит. Сервер зазвичай цього не робить, хоча за багато років існування мережі були вигадані методи, які можуть

дозволити виконати запити з боку сервера. Для відображення веб-сторінки, браузер відправляє початковий запит для отримання HTML-документу, вивчає документ, запитує додаткові файли для відображення вмісту веб-сторінки (скрипти, CSS таблиці [12], додаткові ресурси зображень і відео-файлів ...), які є частиною документа, але розташовані в інших місцях мережі. Клієнт (браузер) з'єднує ресурси для відображення їхнього користувача у вигляді веб-сторінки. Скрипти, що виконуються браузером, можуть отримувати додаткові ресурси по мережі на наступних етапах обробки веб-сторінки, коли браузер оновлює відображення сторінки. Браузер перетворює гіперпосилання на HTTP-запити і надалі отримані HTTP-відповіді відображає у зрозумілому для користувача вигляді.

Веб-сервер

Сервер обслуговує користувача, надаючи йому документи на запит, сервер є віртуальною машиною, що повністю або частково генерує документ, хоча може бути групою серверів, між якими балансується навантаження – перерозподіляються запити різних користувачів, або складним програмним забезпеченням, що опитує інші комп'ютери (кешируючі сервери, сервери баз даних, сервери програм електронної комерції ...). Сервер не обов'язково розташований на одній машині, кілька серверів можуть бути розташовані на одній машині, за HTTP/1.1 при загальному заголовку Host вони можуть ділити IP-адресу.

Проксі

Між веб-браузером та сервером – велика кількість вузлів, що передають HTTP повідомлення, що оперують на транспортному, мережевому, канальному або фізичному рівнях, прозорих на HTTP рівні та можливо знижують продуктивність. На рівні додатків ці операції називаються проксі, прозорі чи ні (змінні запити не пройдуть через них), здатні виконувати різні функції:

- caching (кеш публічний або приватний, як кеш для браузера);
- фільтрація (як сканування антивірусу, батьківський контроль, …);

- вирівнювання навантаження (кілька серверів обслуговують різні запити);
- аутентифікація (контроль доступу до різних ресурсів);
- протоколювання (дозвіл на зберігання історії операцій).

Основні аспекти HTTP

Простота. HTTP/2 ввів інкапсуляцію HTTP-повідомлень у кадри, але HTTP залишився простий і зручний. HTTP-повідомлення можуть читатися та розумітись, легко тестуватися, мати зменшену складність для нових користувачів.

Розширюваність. HTTP-заголовки роблять протокол легким для розширення та експериментування. Нова функціональність може бути введена угодою між клієнтом та сервером про семантику заголовка.

Нема стану, але є сесії. HTTP не має стану: немає зв'язку між двома запитами, які послідовно виконуються по одному з'єднанню. З цього випливає можливість проблем користувача, намагається взаємодіяти з певною сторінкою послідовно. Але хоча ядро HTTP не має стану, coockie зберігають сесії із збереженням стану. За рахунок розширюваності заголовків coockie додаються до робочого потоку, дозволяючи сесії на кожному HTTP-запиті ділитися деяким контекстом або станом.

З'єднання. З'єднання керується на транспортному рівні та виходить за межі HTTP. HTTP не вимагає від транспортного протоколу з'єднань, важлива надійність або відсутність втрачених повідомлень. HTTP покладається стандарт TCP, заснований на з'єднаннях, але з'єднання який завжди потрібно. HTTP/1.0 відкривав TCP-з'єднання для кожного обміну запитом/відповіддю, маючи два недоліки: відкриття з'єднання вимагає декількох обмінів повідомленнями, що повільно, хоча ефективніше при надсиланні декількох повідомлень або при регулярній відправці повідомлень. HTTP/1.1 дав конвеєрну обробку та стійкі з'єднання, що лежить в основі TCP з'єднання можна контролювати через заголовок Connection. HTTP/2 додав мультиплексування повідомлень через просте з'єднання для більшої ефективності.

Управління HTTP. HTTP потік. Коли клієнт хоче взаємодіяти з сервером, кінцевим чи проміжним, він виконує кроки:

Крок 1. Відкриття TCP-з'єднання: TCP-з'єднання використовується для надсилання запиту (або запитів) та отримання відповіді. Клієнт відкриває нове з'єднання, перевикористовується існуюче або відкриває кілька TCP-з'єднань до сервера.

Крок 2. Надсилання HTTP-повідомлення: HTTP-повідомлення (до HTTP/2) людині. У HTTP/2, прості повідомлення інкапсулюються у кадри, неможливо їхнє пряме читання, але вони залишаються такими ж.

Якщо активований конвеєр HTTP, кілька запитів можуть бути надіслані без очікування отримання першої відповіді повністю. HTTP-конвеєр погано впроваджується в мережі, де старе програмне забезпечення співіснує з сучасними версіями. HTTP-конвеєр був замінений на HTTP/2 на більш надійні мультиплексні запити у кадрі.

HTTP повідомлення. До HTTP/2 повідомлення людиночитані. У версії HTTP/2 ці повідомлення вбудовані в новий бінарний кадр, що дозволяє компресію заголовків та мультиплексування. Якщо частина оригінального повідомлення HTTP відправлена в цій версії HTTP, семантика повідомлення не змінюється і клієнт відтворює (віртуально) оригінальний запит HTTP. Це важливо для розуміння HTTP/2 повідомлень у форматі HTTP/1.1. Існує два типи HTTP повідомлень, запити та відповіді, кожен у своєму форматі.

Запити містять елементи:

- HTTP-метод, дієслово GET, POST або іменник OPTIONS або HEAD, що визначають операцію клієнта. Зазвичай клієнт хоче отримати ресурс (використовуючи GET) або передати значення HTML-форми (використовуючи POST), інші операції необхідні в інших випадках;
- шлях до ресурсу: URL ресурси позбавлені елементів, які є очевидними з контексту, наприклад без протоколу (http://), домену (тут developer.mozilla.org), або TCP порту (тут 80);

– версію HTTP-протоколу;
– заголовки (опційно), що надають додаткову інформацію для сервера;
– тіло для методів, таких як POST, що містить надісланий ресурс.

Відповіді містять елементи:

– версію HTTP-протоколу;
– HTTP код стану, що повідомляє про успішність запиту або причину невдачі;
– повідомлення стану – короткий опис коду стану;
– HTTP заголовки, подібно до заголовків у запитах;
– опціонне тіло, що містить ресурс, що пересилається.

HTTP — легкий протокол, що розширюється. Структура клієнт-сервера, із здатністю до додавання заголовків, дозволяє HTTP просуватися з можливостями Мережі. Хоча HTTP/2 додає складність, вбудовуючи повідомлення HTTP у кадри для продуктивності, базова структура повідомлень залишилася з HTTP/1.0. Сесійний потік простий, дозволяє досліджувати та налагоджувати HTTP-повідомлення.

1.2.2. Комутації WEB-сервісів

В основі WEB-сервісів [13] – Service-Oriented Architecture (SOA) [14] – парадигма організації та використання розподілених можливостей, що можуть належати різним власникам.

Базова у СОА – сутність «дії» (на противагу сутності «об'єкта» об'єктно-орієнтованої архітектури). Складовими СОА архітектури є:

– сервісні компоненти (сервіси);
– контракти сервісів (інтерфейси);
– з'єднувачі сервісів (транспорт);
– механізми виявлення сервісів (реєстри).

Сервісні компоненти (сервіси) описуються програмними компонентами, що забезпечують прозору мережеву адресацію.

Контракт сервісу (інтерфейс) забезпечує опис можливостей та якості послуг, що надаються сервісом. В інтерфейсі визначено формат повідомлень

для обміну інформацією, вхідні та вихідні параметри методів, що підтримуються сервісним компонентом.

З'єднувач сервісів забезпечує обмін інформацією між сервісними компонентами. Разом з відкритими стандартами опису інтерфейсів використання гнучких транспортних протоколів для обміну інформацією між сервісними компонентами дозволяє підвищити програмну сумісність сервіс-орієнтованої системи.

Механізми виявлення сервісів (регістри сервісів) використовуються для пошуку сервісних компонентів шуканої функціональності. Серед різних систем виявлення сервісів виділяють дві основні категорії:

– системи статичного виявлення;
– системи динамічного виявлення.

Статичні системи виявлення сервісів, наприклад, UDDI [15], зберігають інформацію про сервіси в системах, що рідко змінюються.

Динамічні системи виявлення сервісів зберігають інформацію про системи з частими появами та зникненням компонентів.

Веб-сервіси – це реалізація СОА, що найчастіше зустрічається, слабопов'язані програмні компоненти, що підтримують багаторазове використання, семантично інкапсулюють окремі функціональні можливості та програмно доступні за протоколами Інтернету, що дозволяють створювати незалежні масштабовані слабопов'язані додатки. Веб-сервіси – це незалежне від платформи та мови програмування середовище, що в основному базується на обміні повідомленнями у форматі XML-документів [16]. Передача повідомлень відбувається за допомогою протоколів HTTP, XML, XSD, SOAP, WSDL, UDDI [9, 16, 17].

Використання HTTP для передачі повідомлень важливе у розробці розподілених додатків – брандмауери та проксі-сервери пропускають HTTP-трафік, при взаємодії немає проблем, як при COM+ та CORBA. Три стандарти для підтримки представлення, пошуку та обміну інформацією між веб-сервісами – WSDL, UDDI, SOAP [17], що утворюють трикутник СОА.

Протокол SOAP (Simple Object Access Protocol) [17] служить організації взаємодії віддалених систем через асинхронний обмін XML-відформатованими документами, що складаються з конверта (обгортки), заголовка та тіла. SOAP формує базовий шар стеку протоколів веб-сервісів, забезпечуючи інфраструктуру обміну повідомленнями між ними.

Мова Web Services Description Language (WSDL) [17] представляє сервіси як абстрактні ресурси, що приймають на вхід документи певних типів та ініціюють відправлення документів інших типів. WSDL описує веб-сервіси та визначає їх розташування. WSDL написано на XML і є XML-документом, визначає обслуговування абстрактно і конкретно. На абстрактному рівні сервіс задається в термінах повідомлень, що посилаються і приймаються ним, які описуються засобами XML-Schema незалежно від транспортного протоколу. На конкретному рівні визначаються прив'язки до транспортних форматів та точок фізичного розміщення. У специфікаціях WSDL 2.0 документи WSDL Part 1: Core language, WSDL Part 2: Message exchange patterns, WSDL Part 2: Bindings.

Протокол UDDI (Universal Description Discovery & Integration) [17] – стандарт на внутрішній пристрій та зовнішні інтерфейси БД із описом сервісів. Описи БД зберігаються у вигляді XML-записів. UDDI 2.0 був призначений для каталогів електронного бізнесу. UDDI 3.0 орієнтований на корпоративне використання для побудови корпоративних систем сервіс-орієнтованої архітектури та допускає створення реєстрів декількох типів (загальнодоступний, приватний та з доступом, що розділяється). UDDI (3.01) вводить реплікацію репозиторіїв зі складними моделями підпорядкованості, побудову репозиторію з кількох вузлів та реплікацію даних між ними, глобальну унікальність записів та ключів, API публікації описів та підписки на зміни, засоби забезпечення цілісності даних, інтернаціоналізації, шифрування.

У цілому в архітектурі веб-сервісів – безліч протоколів і специфікацій, розбитих чотирма частини (процес, опис, повідомлення, зв'язок), утворюють стек протоколів, у якому кожен верхній рівень спирається нижній рівень.

У тілі міститься XML-блок з інформацією, яка має бути доставлена кінцевому адресату. Заголовок – це не обов'язковий елемент SOAP-повідомлення, за допомогою якого передаються дані, які не є основним робочим навантаженням (директиви, інформація про контекст, необхідні для обробки). SOAP-повідомлення здатне слідувати за маршрутом, що містить кілька вузлів, кожен із яких може його обробляти. Статус змін відображається у блоках заголовка повідомлення. Обидва ці розділи містяться всередині конверта. У заголовку SOAP-повідомлення можна ввести нові елементи повідомлення, не передбачені стандартом SOAP. Ці елементи відіграють утилітарну роль по відношенню до основного повідомлення, що міститься у тілі SOAP. SOAP забезпечує односторонню передачу повідомлень, щоб отримати відповідь від сервера, якому передано SOAP повідомлення, потрібно, щоб він ініціював процес передачі повідомлення та встановив з'єднання з клієнтом. Це може спричинити труднощі в умовах організації зв'язку в мережі Інтернет – більшість клієнтів не мають статичних IP-адрес, вхідні з'єднання блокуються мережевими брандмауерами. Проблема вирішена при зв'язуванні SOAP і HTTP, що реалізує паттерн поведінки запитання-відповідь (SOAP HTTP Binding).

Корпорації IBM та Microsoft спільно розробили сімейство стандартів GXA – Global XML Web Services Architecture [18] зі стандартом забезпечення безпеки веб-сервісів WS-Security, що є базою для інших технологій у сфері безпеки веб-сервісів. В рамках стандарту описуються процеси забезпечення цілісності, конфіденційності та автентичності SOAP-повідомлення, що передається в рамках уже встановлених сесій, контексту та політики безпеки. GXA визначає рамкову архітектуру, для реальних операцій вона покладається на технології PKI, Kerberos і SSL. WS-Security [19] орієнтований на комплексне вирішення завдань безпеки у взаємодії веб-сервісів, забезпечуючи визначення основних методів ідентифікації користувача, цифрові підписи, шифрування.

1.2.3. Комутація в AJAX, JSON

AJAX [11] – це абревіатура Asynchronous Javascript and XML. До AJAX і Javascript і XML існують тривалий час, тобто AJAX - це синтез технологій.

AJAX асоціюється з динамічним Web 2.0. При використанні AJAX не оновлюватимуться вся сторінка, оновлюється лише її конкретна частина. Це зручніше – не доводиться чекати, і економічніше за лімітного інтернету. Однак розробник повинен стежити, щоб користувач був у курсі того, що відбувається на сторінці, реалізується через індикатори завантаження, текстові повідомлення про обмін даними з сервером. Також не всі браузери підтримують AJAX (старі та текстові версії не підтримують), також Javascript може бути вимкнений користувачем. Тому технологія не універсальна, важливі й альтернативні методи представлення динамічної інформації на веб-сайті.

Переваги AJAX

- можливість створення зручного Web-інтерфейсу;
- активну взаємодію з користувачем;
- часткове перезавантаження сторінки замість повного;
- зручність використання.

AJAX використовує два методи роботи з веб-сторінкою: зміна Web-сторінки без її перезавантаження та динамічний доступ до сервера. Друге здійснюється декількома способами, зокрема, XMLHttpRequest [16] та використанням техніки прихованого фрейму.

Обмін даними. Для обміну даними на сторінці створюється об'єкт XMLHttpRequest, який є посередником між браузером та сервером. XMLHttpRequest дає змогу надіслати запит на сервер і отримати відповідь як різні данні. Обмін даними із сервером можливий двома способами. Перший спосіб – GET-запит, в якому звертаються до документа на сервері, передаючи йому аргументи через URL. На стороні клієнта використовують функцію escape з Javascript`у, щоб дані не перервали запит. Не рекомендується робити запити GET до сервера з великими обсягами даних. І тому існує POST-запрос.

Клієнт на Javascript повинен давати необхідну функціональність для безпечного обміну з сервером та надавати методи для обміну даними будь-яким із вищезазначених способів. Серверна частина повинна обробляти вхідні дані, і на їх основі генерувати нову інформацію (з БД…) і віддавати її назад клієнту.

Так, для запиту інформації з сервера можна використовувати GET-запит із передачею невеликих параметрів, а для оновлення інформації використовувати POST-запит, оскільки він дозволяє передавати великі обсяги даних.

AJAX використовує асинхронну передачу даних – поки йде передача, користувач може виконувати інші дії. У цей час слід сповістити користувача про обмін даними, щоб він не залишив сайт або повторно не викликав функцію, що «зависла». Індикація при обміні даними в Web 2.0 важлива, багато хто ще не звикли цим способам оновлення.

Відповідь від сервера може бути не тільки XML, можлива відповідь у вигляді звичайного тексту, або JSON (Javascript Object Notation) [20]. Якщо відповідь була отримана простим текстом, її можна відразу вивести в контейнер на сторінці. При отриманні відповіді у вигляді XML відбувається обробка XML документа на стороні клієнта і перетворення даних до XHTML. При отриманні відповіді у форматі JSON клієнт повинен виконати код (функція eval з Javascript), щоб отримати повноцінний об'єкт Javascript. JSON – це об'єктна нотація Javascript, за її допомогою можна представити об'єкт у вигляді рядка (аналогія з функцією серіалізації). При отриманні даних JSON потрібно виконати їх, щоб отримати повноцінний об'єкт Javascript і зробити з ним необхідні операції. Потрібно враховувати, що може бути переданий шкідливий код, перед виконанням отриманого коду його треба перевірити та обробити.

Також можливий «холостий» запит, у якому відповідь від сервера не надходить, лише змінюються дані за сервера. У різних браузерах даний об'єкт має різні властивості, але в цілому він збігається. На підставі стану готовності об'єкта відвідувачу можна надати інформацію про те, на якій стадії знаходиться процес обміну даними з сервером і, можливо, сповістити його про це візуально. Створення об'єкта XMLHttpRequest для кожного типу браузера є унікальним процесом, можна створювати цей об'єкт для популярних браузерів і не турбуватися про його працездатність. Створити об'єкт можна у різних місцях, якщо створити його глобально, то певний момент часу можливий буде лише

один запит до сервера. Можна створювати об'єкт завжди при запиті на сервер (це майже вирішить проблему).

Запит на сервер. Алгоритм запиту до сервера виглядає так:

Крок 1. Перевірка XMLHttpRequest.

Крок 2. Ініціалізація з'єднання із сервером.

Крок 3. Здійснення запиту серверу.

Крок 4. Обробка даних.

Для створення запиту до сервера будується функція, що по функціональності поєднує функції GET та POST запитів.

1.3. Аналіз комутації у експериментах з автоматами

Для автоматних моделей елемент часової поведінки – це стан чи перехід. Структурування елементів поведінки в автоматі дають часову модель поведінки [21], розмірність завдань її поведінкового аналізу визначається експонентними залежностями від потужності автоманих алфавітів.

Тому дослідження й практичне застосування моделей і методів ідентифікації автоматного класу суттєво основані на декомпозиції [22].

1.3.1. Комутація у мережних експериментах

У мережі, що відображає росподіл компонентів у просторі PIC, перевірки окремих автоматів, які зокрема містять перевірки елементів комутації, доповнени: а) визначенням вхідної поведінки деякого автомата, реалізованої від входів мережі; б) визначенням вихідної поведінки автомата, транспортованої до виходів мережі; в) організації перевірки на неповній реалізованій вхідній і транспортованої вихідній поведінках автомата в його мережному оточенні.

У автоматної мережі, яка коректна по Глушкову [23], компонентні автомати – автомати Мілі/Мура, їх функції переходів/виходів з визначеними елементами комутації, як й функція усієї мережі, визначається складними співвідношеннями. Помилки мережи – це помилки функцій компонентних автоматів й мережі у цілому.

1.3.2. Керованість і наочність у мережних експериментах

Керованість і наочність, або реалізація і транспортування поведінки безпосередньо пов’язані з комутацією. Для контрольного аналізу автоматних композицій важливі системи операцій, які описують базові паралельне й послідовне з'єднання, а також з'єднання зі зворотним зв'язком [24].

Мережа автоматів (СА) PIC, що прийнята як основна вхідна модель для визначення комутації та її верифікації для компонентів PIC, основана на синхронному аналізі автоматів, як однорівневих взаємодіючих об'єктів. СА визначається множиною вхідних у неї КА, структурою зв'язків СА, як зв'язків між КА й зв'язків КА із зовнішніми входами й виходами СА.

В СА виділені відносини алфавітних погоджень для входів і виходів КА й СА відповідно до структури зв'язків, що відповідають відображенням і синхронізації вхідних і вихідних алфавітів.

Попередньому аналізу стосовно реалізації й транспортування для верифікації комутації, підлягають компонентні автомати, структура зв'язків СА й відносини її алфавітних погоджень.

1.3.3. Компонентний автомат й мережева модель для комутації

Модель автомата в СА – це синхронний автомат [22-24] $A = (S, X, Y, \delta, \lambda, S_o)$, де S – множина внутрішніх станів, X і Y – вхідний і вихідний алфавіти, $\delta: S \times X \to S$ – функція переходів, $\lambda: S \times X \to Y$ (автомат Милі), або $\lambda: S \to Y$ (автомат Мура) – функція виходів, $S_o \subseteq S$ - підмножина початкових станів.

Клас дефектів, для яких розглядається верифікація, реалізація й транспортування комутації, прийнятий клас дефектів функціонального типу для автомата, як відхилення виділених відображень з еталонних автоматних функцій переходів δ і виходів λ, що не збільшують потужність множини станів S’ автомата, що перевіряється, тобто $A' = (S', X, Y, \delta', \lambda', S_o')$, $S' \leq S$ [25].

Можлива стратегія верифікації, реалізації й транспортування комутації залежить вид експериментів [26], які виконуються для кожного компонентного автомата.

Важливо отримати й проаналізувати реалізації й транспортування, що визначають множину експериментів верифікації комутацій для деякого компонентного автомата та її підмножину, яка задовольняє умовам реалізації й транспортування СА [27].

Мережа автоматів СА визначена як СА = (X, Y, A^, α^), де X і Y - загальні, зовнішні вхідні й вихідні її алфавіти, $A^ = \cup_{i\in I} A_i$ – множина компонентних автоматів, $\alpha^ = \cup_{i\in I} \alpha_i$ – множина алфавітних відображень виду $\alpha_i: X\times Y_{i'} \to X_i$, $i'\in I\backslash\{i\}$, $Y=X_k$ для деякого $k\in I$ [27, 28].

Клас помилок СА, для яких виконується верифікація [27, 28], поєднує класи постійних дефектів функціонального типу для кожного автомата A_i з множині A^ та не включає явно клас постійних дефектів топологічного типу, що представляють будь-якими відхиленнями від еталонних алфавітних відображень $\alpha_i: X\times Y_{i'} \to X_i$ з множині α^.

На попередніх етапах верифікації комутації у мережі потрібний структурно-топологічний аналіз СА та структурно-функціональний аналіз умов взаємної синхронізації.

1.4. Висновки

У першому розділі розглянути та проаналізовані основні технології та протоколи транспортно-мережевого та прикладного рівнів для специфікації, аналізу, побудові і верифікації поведінки PIC та MAC, зокрема міжкомпонентної комутації.

Розглянути стандарти, протокроли,, властивости й механізми взаємодії на основі інтерфейсу сокетів, компонентного програмування, особливості визначення та використання сокетів та програмних компонентів з акцентуванням комутацій.

У розділі також виконано аналіз сучасних прикладних технологій, Веб-технологій та протоколів прикладного рівня, властивостей та механізмів HTTP, XML, AJAX, JSON.

У верифікації комутації MAC розглянуті автоматні та мережеві моделі з урухуванням реалізації і транспортування поведінки при організації перевірки.

Зроблено висновок про наявність завдань визначення верифікації комутацій для MAC, що вимагають рішення й розвитку.

2. РОЗРОБКА І ДОСЛІДЖЕННЯ ФОРМАЛЬНОЇ МОДЕЛІ КОМУТАЦІЇ

2.1.Завдання умов й вимог роботи

2.1.1. Визначення мети й завдань

Робота має своєю ціллю удосконалення моделей верифікації комутації у автоматної поведінці для компонентів MAC.

Для досягнення цілі вирішуються завдання:

- дослідження функціональних й логічних специфікацій моделей і методів аналізу для комутації PIC та MAC;
- побудови, аналізу і моделювання поведінкових автоматних моделей та базових кроків верифікації комутації MAC;
- програмно-алгоритмічної реалізації базових кроків та алгоритмів верифікації комутацій MAC.

2.1.2. Функціональні, інформаційні й часові умови

Рост важливості структурно-функціонального проектування для складних PIC і MAC вимагає використання ефективних моделей формальних високорівневих специфікацій, аналізу, синтезу, верифікації і контролю. Розподілення у часі і просторі взаємодії подій і процесів поведінки компонентів MAC ускладнює її аналіз й контроль.

Функціональні, інформаційні та часові вимоги визначають можливість формальних специфікацій комутаційної поведінки та її верифікації для основних синхронізованих властивостей і механізмів взаємодій транспортно-мережевого та прикладного рівнів, що можливе за допомогою автоматних моделей та їх мережевих композицій.

Це обумововило обрання автоматних моделей і їх композицій у мережах, з абстрактними алфавитами, синхронним часом, паралелизмом, як вхідних до верифікації комутації агентів MAC, яка у завданнях работи виконується на базі спеціального автоматного подання поведінки PIC і MAC, зокрема для комутації агентів MAC, як Веб-сервісу, експериментах розпізнавання для автоматів.

Вхідними даними функціонування засобів верифікації можуть бути:

- компонентна структура МАС як Веб-сервісу з акцентованою комутацією;
- функціональні специфікації загальних і компонентних комутаційних функцій МАС;
- формати загальних і компонентних інформаційних комутаційних об'єктів МАС;
- функціональні специфікації компонентних інтерфейсів комутації в структурі МАС;
- функціональні специфікації базових сценаріїв роботи МАС.

Вихідними даними функціонування засобів верифікації можуть бути:

- формальні моделі верифікації комутації автоматного класу у складі механізмів МАС, як Веб-сервісів;
- інтуїтивні інтерактивні і формальні кроки побудови перевірок верифікації комутації у складі механізмів МАС.

Функціями засобів верифікації комутації агентів, що забезпечуються сукупністю моделей і кроків, можуть бути:

- побудова моделей і кроків верифікації для властивостей і механізмів МАС, як Веб-сервісів;
- побудова елементарних структурних і функціональних перевірок верифікації для властивостей і механізмів МАС, як Веб-сервісів.

Теорії автоматів, експериментів з автоматами, графів і алгоритмів використано як математичний й методологічний інструментарій.

2.2.Побудова моделей верифікації для комутацій

Для побудови формальної автономної моделі верифікації комутації агента в якості вхідної системи алгебри прийнята модель компонента PIC або МАС, як автомат, розширений інтервалами і недетермінізмом. Процедури верифікації базуються на методах теорії поведінкових автоматних експериментів з

еволюційною ортимизацией, тобто, у роботі будуються спеціальні еталонні опорні ідентифікатори і примітиви верифікації, з урахуванням яких еволюційно формуються верифікаційні фрагменти – автоматні експерименти як особин еволюційних популяцій. Формування складає основі операцій верифікаційних і еволюційних об'єктів.

2.2.1 Вхідні базові моделі

Вхідна базова модель аналізу, що визначає автоматну структуру поведінки агента, зокрема його комутації, може бути представлена орграфом у вигляді трійки $G = (S,V,\delta)$,, де S – безліч вершин, V – безліч дуг, δ – відображення виду $\delta: S \times V \to S$.

Орграф дозволяє представляти всі основні топологічні структури-псевдографи – зв'язані та незв'язані ланцюги, дерева, гамаки, сітки, слабо зв'язані та сильнопов'язані орграфи.

Моделі та методи теорії загального графового аналізу становлять остовну частину методів верифікації, мають відомі ефективні реалізації, у зв'язку з чим у даній роботі не представлені.

Як відзначено, як вхідна модель верифікації прийнята часовий недетермінований автомат Милі (ВНДА) $a = (S,X,Y,T,Pb,\delta_T,\lambda_T,S_0)$ (див. рис. 2.1) із властивостями інтервалів часу й недетермінованості для функцій переходів і виходів який дозволяє специфікувати розширені поведінкові властивості й механізми агентів МАС. Тут, по-перше, S,X,Y – алфавіти станів, вхідна й вихідний, $|S|=n$, $|X|=m$, $|Y|=l$, по-друге, $T \subset N$ – безліч тимчасових відрізків відповідним інтервалам, по витіканню яких відбувається перемикання в чергові стани функції переходів δ_T і під час яких відбувається дія вихідних сигналів функції виходів λ_T автомата Милі, по-третє, $Pb \subset [0;1] \subset D$ – безліч імовірнісних коефіцієнтів діапазону $[0;1]$, по-четверте, $\delta_T: S \times X \times T \times Pb \to S$ – тимчасова недетермінована функція переходів, по-п'яте, $\lambda_T: S \times X \times T \times Pb \times S \to Y$ – тимчасова недетермінована функція виходів для переходів, по-шосте, $S_0 \subseteq S$ – підмножина початкових станів.

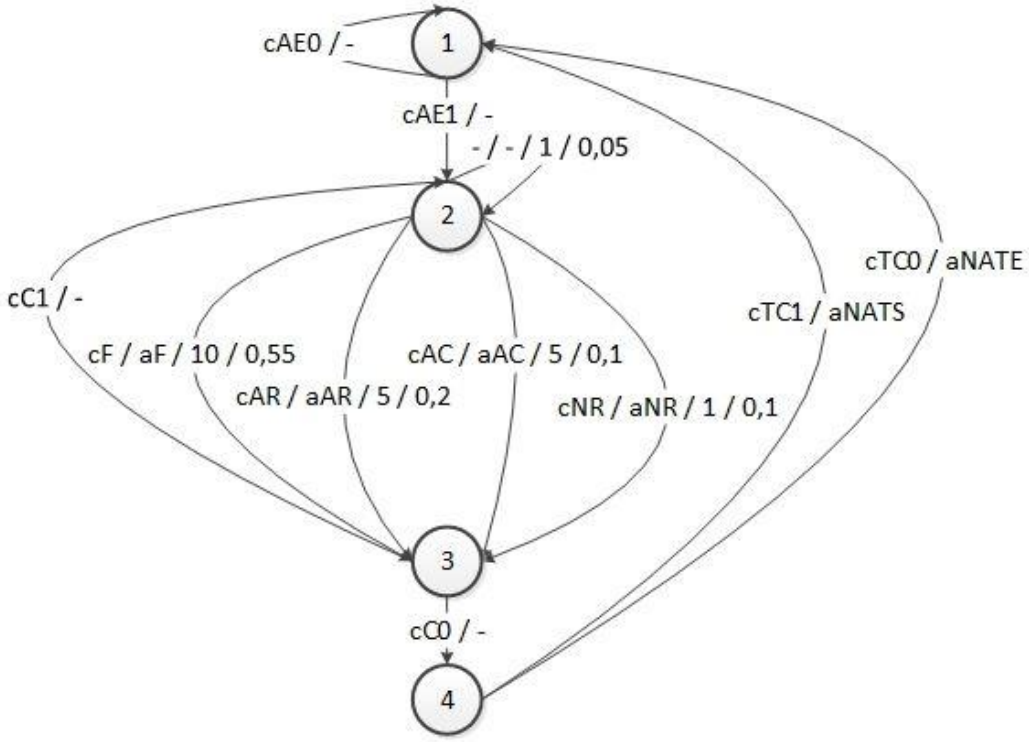


Рисунок 2.1 – ВНДА *a* для NAT

Для *a* і його деякої четвірки $(s,x,t,pb) \in S \times X \times T \times Pb$ можливий вивід з деякою ймовірністю $pb \in Pb_{s,x,t}$, що $\exists s' \in S(s' = \delta_T(s,x,t,pb)$ або $s' \in \delta_T(s,x,t,Pb_{s,x,t}))$, де $\Sigma_{pb \in Pbs,x,t}\, pb = 1$. Це представляється у формі існування з імовірністю *pb* деякого недетермінованого переходу-п'ятірки (s,x,t,pb,s') тривалістю $t \in T$. Якщо для (s,x,t,pb,s') вірно з деякою ймовірністю $pb' \in Pb_{s,x,t'}$, що $\exists y \in Y(y \in \lambda_T(s,x,t,pb,s',pb'))$, де $\Sigma_{pb \in Pbs,x,t'}\, pb=1$, то серед вихідних сигналів переходу (s,x,t,pb,s',t',pb') є вихід *y* з інтервалом дії $t' \in T$. Інтервал t' дії виходу *y* уміщається в інтервал *t* переходу (s,x,t,pb,s') – існують $t'',t''' \in T \cup \{e\}$ такі, що $t=t''\,\%\,t'\,\%\,t'''$. Дана подія інтерпретується як існування з імовірністю $pb \times pb'$ об'єднаного недетермінованого переходу у вигляді вісімки $(s,x,t,pb,s',t''t't''',pb',y)$.

Знижуючи складність верифікаційного аналізу з невеликою втратою його спільності можливо прийняти рівними дискретний час очікування нового стану на переході й час дії вихідної оцінки $t=t'(t''=t'''=0)$. Також можна вважати, що

недетермінізм переходу має ймовірність r появи пари (s',y). Таким чином, перехід є шістка (s,x,t,r,s',y). Автомат a для зниження його розмірності наведений до часткового напівавтомата за рахунок формування вхід-вихідного алфавіту $U=X\times Y$, у якому безліч породжуваних слів $W=U^*\cup\{e\}$ поповнено «нульовим» порожнім символом «e». У цьому випадку також модифікуються функції переходів-виходів δ_T, λ_T:

$$a = (S,U,T,Pb,\Delta_T,S_0), \tag{2.1}$$

тут

– $\Delta_T: S\times U\times T\times Pb\to S$ – тимчасова, недетермінована функція переходів-виходів;

– інші компоненти представлені вище.

Для a ступінь часткової визначеності в порівнянні з a нижче саме в силу зазначеного спрощення й переходу до алфавіту U.

У переході (s,x,t,pb,s',y) автомата a для автомата a очевидні перестановка й угруповання компонентів для приведення у відповідність алфавіту U і об'єднаної функції Δ_T. Якщо $u=(x,y)$, то для деякої четвірки $(s,u,t,pb)\in S\times U\times T\times Pb$ вірно $\exists s'\in S\ (s'\in\Delta_T(s,u,t,pb))$, це представляється як існування деякого недетермінованого переходу – п'ятірки (s,u,t,pb,s'). В автоматах a й a визначені підмножини алфавітів $X'\subseteq X$, $Y'\subseteq Y$, $U'\subseteq U$, для яких є зовнішнє спостереження й керування. Ці алфавіти поповнені «нульовим» символом «e», «невідомим» символом «χ» і байдужним символом «~», внаслідок формуються зовнішні алфавіти для автоматів a й a. Очевидно, що наочність/керованість для X', Y' вище, ніж для U' через можливість розгляду X', Y' окремо. У цих алфавітах із символами «χ» і «~» для відповідно неспостережуваних (невизначених) і байдужних символів можлива специфікація зовнішнього поводження, по якому можна робити виводи про відповідність моделей, що *перевіряють*, $a^\wedge$ *і* $^\wedge\wedge$ вимогам еталонних *моделей* а *й* . Без великої втрати спільності можна обмежитися моделями $a^\wedge$ і a.

Тегда клас верифікуємих властивостей *Pr* еталонного *a*, відповідно до якого верифікується що перевіряє $a^\wedge$ і в цілому вдосконалюється модель верифікації, можна обмежити відповідністю функцій переходів-виходів Δ_T еталонного $a=(S,U,T,Pb,\Delta_T,S_0)$ і $\Delta_T^\wedge$ *що* перевіряє $a^\wedge=(S^\wedge,U^\wedge,T^\wedge,Pb,\Delta_T^\wedge,S_0^\wedge)$ з обмеженням $|S^\wedge|\leq|S|$ для зменшення розмірності відповідності Δ_T і $\Delta_T^\wedge$.

Тоді клас можливих помилок для $a^\wedge$ можна представити статичною частиною – перекручуваннями в переходах і виходах у функції $\Delta_T^\wedge$, динамічною частиною – перекручуваннями інтервалів $T^\wedge$ в $\Delta_T^\wedge$, а також недетермінованою частиною – перекручуваннями ймовірностей $Pb^\wedge$ в $\Delta_T^\wedge$.

Слід зазначити, що перевірка интервальных властивостей $T^\wedge$ функції $\Delta_T^\wedge$ на відповідність властивостям *T* функції Δ_T може вироблятися цілеспрямовано або у фоновому режимі, зокрема, у ході перевірки переходів-виходів функції $\Delta_T^\wedge$ на відповідність переходам-виходам функції Δ_T, як не перевищення $t_j^\wedge\leq t_j$ взаємно відповідних інтервалів $\forall t_j^\wedge\in T^\wedge$ і $\forall t_j\in T$.

Верифікація в перевірці недетермінованості $Pb^\wedge$ функції $\Delta_T^\wedge$ заснована на використанні в моделі верифікації:

– статистики, для якої поточна розмірність безлічі перевірок на кожному з переходів-виходів відображається у відносних величинах у лічильниках $pb_{j\Delta T}^{stat}\in Pb$, що модифікують черговою розміткою $pb_{j\Delta T}^{stat}(time+1)$ поточну розмітку *r* переходу *(s,u,t,r,s')* або модифікувати поточну розмітку $pb_{j\Delta T}^{stat}(time)$ переходу $(s,u,t,pb_{j\Delta T}^{stat}(time),s')$ до чергового виду розмітки $pb_{j\Delta T}^{stat}(time+1)$ для переходу $(s,u,t,pb_{j\Delta T}^{stat}(time+1),s')$ на наступному кроці. Тобто можливий поточний статистичний аналіз варіантів недетермінізму на відповідність емпіричним статистичним даним, накопиченим на підставі функціонування агентів МАС.

– директив при уведенні детермінізму в Δ_T за рахунок поточних пріоритетів перевірок на кожному з переходів-виходів, що представляють у відносних величинах у пріоритетних змінних $pb_{j\Delta T}^{priort}\in Pb$, що модифікують черговою розміткою $pb_{j\Delta T}^{priort}(time+1)$ для поточної пріоритетної розмітки *r*

переходу (s,u,t,r,s') або для поточної розмітки $pb_{j\Delta T}^{priort}(time)$ переходу $(s,u,t,pb_{j\Delta T}^{priort}(time),s')$ до чергового виду $pb_{j\Delta T}^{priort}(time+1)$ для переходу $(s,u,t,pb_{j\Delta T}^{priort}(time+1),s')$ наступного кроку. Тобто, можлива поточна модифікація пріоритетного вибору переходів-виходів в Δ_T.

У першому випадку трансформація для (2.1) розширює функцію переходів-виходів Δ_T модифікацією статистики появи переходів Δ_T^{stat} в $a_U^{stat}=(S,U',T,Pb^{stat},\Delta_T^{stat},S_0)$, *тут* $\Delta_T^{stat}:S\times U'\times T\times Pb^{stat}\to S$ – тимчасова недетермінована функція переходів-виходів з відносними двокомпонентними лічильниками статистики pb^{stat} з $Pb^{stat}\subset[0;1]^2$ для кожного з недетермінованих переходів, перший компонент $pb_1=i_{H1}(pb^{stat})$ визначає еталонну ймовірність переходу, другий компонент $pb_2=i_{H2}(pb^{stat})$ – досягнуту його поточну відносну статистику.

Другий компонент лічильника pb^{stat} - це дріб $pb_2=r_1/r_2$, тут r_1 – число виконаних переходів-п'ятірок (s,u',t,pb^{stat},s'), r_2 – число всіх виконаних переходів на основі вхідних трійок (s,u',t) функції Δ_T^{stat}, що породжують розгалуження недетермінізму із приватними рішеннями виду (s,u,t,pb^{stat},s'). У побудові перевірок для кожної п'ятірки (s,u',t,pb^{stat},s') переходу в деякий новий стан $s'=\Delta_T^{stat}(s,u',t,pb^{stat})$ модифікується другий компонент $pb_2=i_{H2}(pb^{stat})$ для числа поточних статистичних перевірок у лічильнику цього переходу, що зберігає відносні значення після переходу виду $pb_2'=(r_1+1)/(r_2+1)$ і фіксує поточне наближення до граничного значення першого компонента $pb_1=i_{H1}(pb^{stat})$.

Друга трансформація для (2.1) модифікує її функцію переходів-виходів Δ_T пріоритетами-семафорами переходів: $a^{prior}=(S,U',T,Pb^{prior},\Delta_T^{prior},S_0)$, тут $\Delta_T^{prior}:S\times U'\times T\times Pb^{prior}\to S$ – функція детермінізації з директивними пріоритетами-семафорами вибору варіантів дозволу недетермінізму, $pb^{prior}\in Pb^{prior}\subset[0;1]^2$ для недетермінованих переходів, перший компонент $pb_1=i_{H1}(pb^{prior})$ містить еталонний відносний пріоритет переходу, другий компонент $pb_2=i_{H2}(pb^{prior})$ – поточний відносний пріоритет.

Другий компонент із pb^{prior} має вигляд правильного дробу $pb_2=pb_1-(r_1/r_2)$, тут r_1 і r_2 мають той же зміст, як і для статистичної моделі: r_1 – число виконаних переходів-п'ятірок $(s,u,\ t,pb^{prior},s')$, r_2 – число виконаних переходів для вхідних трійок (s,u',t) функції Δ_T^{prior}, що має розгалуження недетермінізму із приватними рішеннями (s,u,t,pb^{prior},s'), pb_2 – еталонне число пріоритету в інтервалі $[0;1]$, не суперечній умові $\Sigma_{pb^{prior}\in Pbs,x,t}\, pb^{prior}=1$.

При побудові перевірок для кожної п'ятірки (s,u',t,pb^{prior},s') переходу в деякий новий стан $s'=\Delta_T^{prior}(s,u',t,pb^{prior})$ змінюється другий компонент $pb_2=iн_2(pb^{prior})$ для числа поточних пріоритетних перевірок у відносному лічильнику цього переходу, після переходу він змінює в компоненті pb_2' значення на $(r_1+1)/(r_2+1)$, директивно задає нове число пріоритету в інтервалі $[0;1]$, що не порушує умову $\Sigma_{pb^{prior}\in Pbs,x,t}$ $pb^{prior}=1$ і фіксує поточне наближення до граничного значення першого компонента $pb_1=iн_1(pb^{prior})$ лічильника.

2.2.2 Визначення вхідої мережевої моделі

Для мережної моделі верифікації вводяться різні типи цільових об'єктів, у перевірці СА для МАС на основі поведінкових експериментів як об'єкти визначені фрагменти верифікації для компонентних автоматів мережі, у ній реалізовані й комутації, що транспортують при використанні.

Мережа автоматів (див. рис. 2.2) для подання розподіленої МАС визначена як:

$$na=(X,\ Y,\ A,\ \xi), \tag{2.2}$$

тут

– $X,\ Y$ визначені як загальні вхідний і вихідний алфавіти мережі;

– $A=\cup_{i\in I}\ a_i$ визначено як безліч автоматів мережі na одного із представлених вище видів $a_i=(S_i,X_i,Y_i,T_i,Pb_i,\delta_i,\lambda_i,S_{oi})$ з їхніми безлічами вхід-вихідних слів зареєстрованого поводження $W''=\cup_{i\in I}\ W_i''$ у раніше уведених алфавітах $U_i''\subseteq U_i'\times T_i\times Pb_i$, які структуровані мережею na, з урахуванням початкових станів і недетермінованості поводження компонентних автоматів;

– $\xi=\cup_{i\in I}\ \xi_i$ визначено як безліч алфавітних, ймовірносно-тимчасових відображень для структури зв'язків у мережі *na*:

$$\xi_{i''}: x_{i',\in I\setminus\{i\}}(Y_{i\wedge}\times T_{i\wedge}\times Pb_{i\wedge})\to(X_{i\wedge\wedge}\times T_{i\wedge\wedge}\times Pb_{i\wedge\wedge}) \text{ або } \xi_{i\wedge\wedge}: x_{i\wedge\in I\setminus\{i\}}(Y_{i\wedge}'')\to(X_{i\wedge\wedge}''),$$

тут

а) $i\wedge\in I\wedge\subseteq\subseteq I\setminus\{i\}\cup\{0\}$, $i\wedge\wedge\in I\cup\{i+1\}$, $Y_0=X$, $Y=X_{n+1}$;

б) $(x_{i\wedge\wedge},t_{i\wedge\wedge},pb_{i\wedge\wedge})=\xi_{i\wedge\wedge}((y_{i\wedge},t_{i\wedge},pb_{i\wedge}),\ldots,(y_{i\wedge},t_{i\wedge},pb_{i\wedge}),\ldots,(y_{n\wedge},t_{n\wedge},pb_{n\wedge}))$,

причому $t_{i\wedge\wedge}=\cap_{i\wedge\in I\wedge}t_{i\wedge}$ і $pb_{i\wedge\wedge}=\Pi_{i\wedge\in I\wedge}pb_{i\wedge}$.

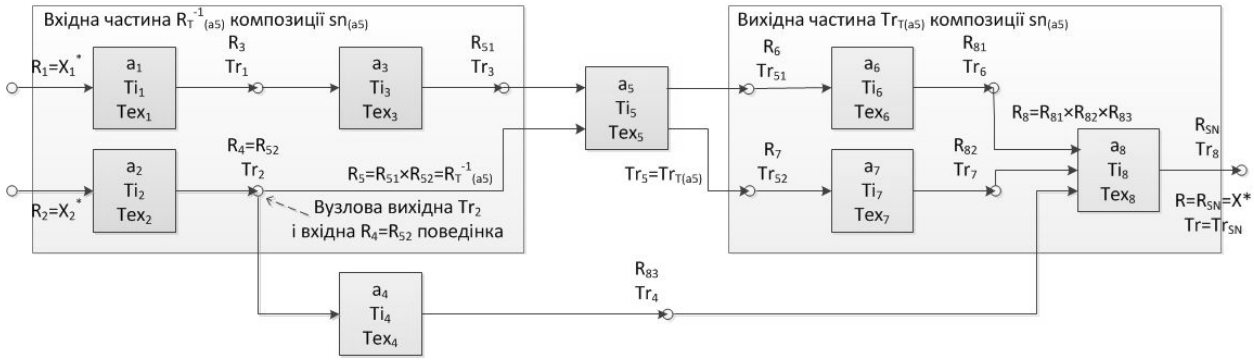


Рисунок 2.2 – Автоматна мережа з автоматом a_i для верифікації комутації

2.2.3. Побудова мережевої моделі верифікації

Компоненти формальної мережної моделі верифікації мають свою інтерпретацію розмірності й повноти перевірки безлічі компонентних автоматів $A=\cup_{i\in I}\ a_i$ (об'єктів) мережі *na*. Мережна модель верифікації для мережі *na* визначена, як алгебраїчна система:

$$tn=(Ta,\ aR_T^{-1}{}_{(A)},\ aTr_{(A)},\ Sg_{tn},\ Re,\ Tre), \qquad (2.3)$$

тут

– $Ta=\cup_{i\in I}\ ta_i$ визначено як безліч компонентних моделей верифікації для компонентних автоматів $A=\cup_{i\in I}\ a_i$ мережі *na*;

– $aR_T^{-1}{}_{(A)}=\cup_{i\in I}\ aR_T^{-1}{}_{(ai)}$, $aTr_{(A)}=\cup_{i\in I}\ aTr_{(ai)}$ визначені як безлічі реалізованих і напівавтоматів топологічних вузлів, що транспортують, *мережі* na, зокрема, входів і виходів компонентних автоматів з $A=\cup_{i\in I}\ a_i$, для будь-якого

$i \in I$ $pr_1 W_i'' \leq aR_T^{-1}{}_{(ai)} \subseteq X_i''^*$, $pr_2 W_i'' \leq aTr_{(ai)} \subseteq Y_i''^*$. Вузли представляють також виходи реалізуючих (від входів мережі *na* для $aR_T^{-1}{}_{(A)}$) пдиереж $T^{-1}(A) = \cup_{i \in I} T^{-1}(a_i)$ і входами транспортуючих (до виходів мережі *na* для $aTr_{(A)}$) підиереж $T(A) = \cup_{i \in I} T(a_i)$). За допомогою безлічей реалізованих і напівавтоматів топологічних вузлів, що транспортують, *мережі* na визначаються безлічі вхідних *реалізованих* $_{a(A)}? \cup_{i \in I} aR_{ai}$ і вихідних що транспортують $aTr = \cup_{i \in I} aTr_{ai}$ слів поводження для кожного $a_i \in A$ у його алфавіті U_i'';

– $Sg_{tn} = \{\circ^{Y''}, \circ^{X''}, \times^{Y''}, \times^{X''}, *^{Y''}, *^{X''}\}$ визначена як сигнатура мережних операцій прямій і зворотної композиції поводження: послідовної, паралельної, зі зворотним зв'язком;

– $Re = \cup_{i \in I} Re_{(ai)}$, $Tre = \cup_{i \in I} Tre_{(ai)}$ визначені як безлічі реалізованих і транспортованих фрагментів поводження – особин при застосуванні еволюційної оптимізації, що визначає стратегію композицій.

Верифікація поводження, у тому числі комутаційного, для кожного $a_i \in A$, представленого безліччю вхід-вихідних послідовностей W_i'', базується на його апріорній інформації про алфавіти U_i'', зокрема, розділених вхідному X_i і вихідному Y_i, функціях Δ_T^{stat}, δ_i^{stat}, λ_i^{stat}, безлічі ідентифікаторів Ti_i, у тому числі початкових Ti_{Hi} і кінцевих Ti_{Ki} $(Ti_{Hi} \cup Ti_{Ki} \subseteq Ti_i)$, відносинах сумісності σ_i, розрізнення η_i, невизначеності τ, квазіпорядку ν, на безлічі Ti_i. У мережі *na* для довільного $a_i \in A$ його представлена безліч вхід-вихідних послідовностей W_i'' обмежено мережею *na* по двох умовах:

– $np_1 W_i'' \subseteq L(aR_T^{-1}{}_{(ai)})$ визначена як безліч вхідних послідовностей $iн_1 W_i''$, які включаються в автоматну безліч слів (мова $L(aR_T^{-1}{}_{(ai)})$) напівавтомата $aR_T^{-1}{}_{(ai)}$) і реалізуються із загальних входів мережі *na* на виході подсети $T^{-1}(a_i)$ – вході a_i;

– $np_2 W_i'' \subseteq L(aTr_{(ai)})$ визначена як безліч вихідних послідовностей $iн_2 W_i''$, які включаються в автоматну безліч слів (мова $L(aTr_{(ai)})$) напівавтомата $aTr_{(ai)}$) і транспортуються до загальних виходів мережі *na* із входу подсети $T(a_i)$ – виходу a_i.

Мережна модель верифікації *tn*, по-перше, бере в якості вхідних дані моделі верифікації $Ta=\cup_{i\in I} ta_i$ поводження автоматів $a_i \in A$, у тому числі їхні примітиви верифікації $Tp=\cup_{i\in I} Tp_i$ на основі ідентифікаторів $Ti=\cup_{i\in I} Ti_i$ для опорних станів автоматів $a_i \in A$; по-друге, надалі обмежує $Ta=\cup_{i\in I} ta_i$ зазначеними умовами реалізуємості in_1W_i'' і транспортуємості in_2W_i''. Формальна модель верифікації $ta_i \in Ta$ (2.2, 2.3) компонентного автомата $a_i \in A$ дана в складі мережної моделі верифікації *tn* (2.9).

Тобто, є у мережній моделі верифікації *tn* на додаток до безлічі компонентних моделей верифікації $Ta=\cup_{i\in I} ta_i$ для $a_i \in A$ є визначення й використання умов реалізуємості in_1W_i'' і транспортуэмості in_2W_i'' фрагментів верифікації $Tf=\cup_{i\in I} Tf_i$ для автоматів $a_i \in A$, зокрема, примітивів $Tp=\cup_{i\in I} Tp_i$ у середовищі мережі *na*. Тому виконується побудова для кожного автомата $a_i \in A$ компонентних моделей реалізації з $a=\cup_{i\in I} aR_i$ і транспортування з $aTr=\cup_{i\in I} aTr_i$, які використаються для побудови реалізованих $aR_T^{-1}{}_{(A)}=\cup_{i\in I} aR_T^{-1}{}_{(ai)}$ на їхніх входах і що транспортують $aTr_{(A)}=\cup_{i\in I} aTr_{(ai)}$ з їхніх виходів вузлових напівавтоматів у мережній моделі верифікації *tn*.

Для компонентних моделей верифікації $Ta=\cup_{i\in I} ta_i$ автоматів $a_i \in A$, які взаємодіють у відповідності зі структурою й відображеннями мережної моделі тестування *tn*, виконується декомпозиція верифікації в мережі *na* у цілому з поліноміальним зниженням її *NP-складності*.

Таким чином, відповідно до мережної моделі верифікації *tn* для мережі *na*, моделі представлені на:

– першому рівні компонентної моделі верифікації $ta_i \in Ta$ для довільного окремого компонентного автомата a_i мережі *na*;

– другому рівні кооперативної моделі верифікації $cTa_i=(ta_i, aR_T^{-1}{}_{(ai)}, aTr_{(ai)})$ для компонентного автомата a_i, що сформована на основі компонентної моделі тестування ta_i у мережному оточенні його реалізуючої $T^{-1}(a_i)$ і транспортуючої $T(a_i)$ подсетей мережі *na*, що включає:

а) внутрішню модель верифікації ta_i для автомата a_i;

б) зовнішні для входу й виходу автомата a_i реалізований $aR_T^{-1}{}_{(ai)}$ і що транспортує $aTr_{(ai)}$ напівавтомати;

– третьому рівні мережної моделі верифікації $tn=(Ta,aR_T^{-1}{}_{(A)},aTr_{(A)},Sg_{tn}, Re,Tre)$, що сформована на основі кооперативних моделей верифікації другого рівня безлічі $CTa=\cup_{i\in I}\ cTa_i$ для компонентних автоматів з $A=\cup_{i\in I}\ a_i$ першого рівня.

Кооперативні моделі верифікації $CTa=\cup_{i\in I}\ cTa_i$ і в їхньому составі моделі верифікації $Ta=\cup_{i\in I}\ ta_i$ компонентних автоматів з $A=\cup_{i\in I}\ a_i$ відповідно до їх взаємодії в мережі na можуть бути використані для побудови різних, відмінних від простої безлічі CTa, кооперацій верифікації другого рівня $CTn=\cup_{j\in J}\ Tn_j$ для забезпечення паралелізму верифікації.

2.2.4. Вхідна редукція поведінки коммтації з Y''-мінімізацією

При побудові кооперативних моделей верифікації для зменшення розмірності верифікації автоматів з $A=\cup_{i\in I}\ a_i$ застосовані автоматні редукції – звуження довільного автомата a_i по його вхідному X_i'' або вихідному Y_i'' алфавітах, це може дозволити виконання мінімізації одержуваних напівавтоматів ai'' і ai'' з метою зниження складності аналізу.

Вихідна модель $ai''=\Downarrow_Y(a_i)$ для розглянутого окремо автомата a_i визначає його регулярну вихідну безліч реалізованих вихідних слів Ri'' у вихідному алфавіті Y_i'' при абстрагуванні від вхідного алфавіту X_i''. Вихідна модель – це напівавтомат виду $ai''=(S_i,Y_i'',\delta_i'')$, що будується при звуженні $\Downarrow_Y$ розмітки автоматних переходів виду $((x,y),t,pb)$ до виду (y,t,pb) у розширеному алфавіті $Y_i''\subseteq Y_i''\times T_i\times Pb_i$, функція переходів для ai'' має вигляд $\delta_i'':Si''\times Y_i''\to Si''$, де $S^{Y''}\subseteq S$.

При порівнянні з вихідним автоматом a_i недетермінізм і надмірність напівавтомата ai'' може збільшитися, отже, можлива Y''-мінімізація числа станів S_i для одержуваного при звуженні напівавтомата ai'' у результаті може бути отриманий мінімізований вихідний напівавтомат $ai''^{min}=(Si'',Y_i'',\delta_i'')$.

Таким чином, операція Y''-мінімізації $ai''^{min}=Y''\text{-}min(ai'')$ для ai'' базується на операції мінімізації детермінованого автомата $min(a)$, що зберігає

недетермінізм вихідного a_i і одержуваного після звуження напівавтомата ai'', а також має особливості узгодження відповідності $t_{i\wedge\wedge}=\cap_{i\wedge\in I\wedge} t_{i\wedge}$ і $pb_{i\wedge\wedge}=\Pi_{i\wedge\in I\wedge} pb_{i\wedge}$. для переходів, що *ототожнюють,* $(s_{i\wedge}1,yi'',s)$, $(s_{i\wedge}2,yi'',s)$ з минимизируемых *станів* $s_{i\wedge}1,s_{i\wedge}2(^{Si}$''.

При $s_{i\wedge1},s_{i\wedge1}\in Si''$ і $\delta^{Y''}{}_i(s_{i\wedge1},iн_1Y'')\sim\delta^{Y''}{}_i(s_{i\wedge2},iн_1Y'')$, де «~» – відношення умовної еквівалентності станів $s_{i\wedge1}$, $s_{i\wedge1}$, якщо $\delta^{Y''}{}_i(s_{i\wedge1},iн_1Y'')=\delta^{Y''}{}_i(s_{i\wedge2},iн_1Y'')$ або $\delta^{Y''}{}_i(\delta^{Y''}{}_i(s_{i\wedge1},iн_1Y''),iн_1Y'')\sim\delta^{Y''}{}_i(\delta^{Y''}{}_i(s_{i\wedge2},iн_1Y''),iн_1Y'')$ виходить:

$$Y''\text{-}min(s_{i\wedge1}, s_{i\wedge2})=\begin{cases} s_{i\wedge1}, \text{ і } \delta^{Y''}{}_i(s_{i\wedge1},Y'')\sim\delta^{Y''}{}_i(s_{i\wedge2},Y''), \\ \text{якщо } t_{i\wedge\wedge}=\cap_{i\wedge\in I\wedge} t_{i\wedge}\neq\varnothing \;\&\; pb_{i\wedge\wedge}=\Pi_{i\wedge\in I\wedge} pb_{i\wedge}\neq 0. \\ \{s_{i\wedge1}, s_{i\wedge2}\} \text{ інакше.} \end{cases} \quad (2.4)$$

Аналіз підмножин станів для мінімізації, що виконується для напівавтомата ai'' в операції Y''-*min* відповідно до модифікованого базового методу мінімізації, приводить до мінімізованого ai''^{min}.

Умови реалізації на вході автомата a_i мережі *па* визначаються безліччю його вхідних слів $R_i^{X''}$ в алфавіті $X_i''=\alpha_i(X''\times Y_{i\wedge}'')$, які реалізуються мережею автоматів *па* з її входів. Нехай X_i* – повна вхідна безліч автомата a_i, розглянутого окремо, $R_i^{X''}\subseteq X_i$*.

При визначенні $R_i^{X''}$ для автомата $a_i\in A$ потрібні моделі, прямій і зворотний структурно-функціональний аналіз у мережі автоматів *па*.

Зв'язку від входу автомата $a_i\subseteq A$ до загальних входів *па* згідно всім зв'язками $\cup_{i\in I}\ \xi_i^{-1}$ визначають подсеть $T^{-1}(a_i)$ з автоматів $A_{T^{-1}(ai)}\subseteq A$, її вихідний алфавіт дає вхідний алфавіт X_i'' автомата a_i відповідно до мережного відображення $\xi_i(Y_{T^{-1}(ai)})=\xi_i(X''\times Y_{i'}'')=X_i''$, $i'=I\backslash\{i\}$.

Вхідний алфавіт мережі автоматів *па* породжує рівність $X''=X'''\times X_{T^{-1}(ai)}''$, тут $X_{T^{-1}(ai)}''$ *що* впливає й X''' не впливає на подсеть $T^{-1}(a_i)$ вхідні подалфавиты мережі *па*.

Подсеть $T^{-1}(a_i)$ породжує пряме $\xi':X_T{}^{-1}{}_{(ai)}{}''^{*}\rightarrow R^{Y''}{}_T{}^{-1}{}_{(ai)}$ і зворотне $\xi'^{-1}:R^{Y''}{}_T{}^{-1}{}_{(ai)}\rightarrow X_T{}^{-1}{}_{(ai)}{}''^{*}$ відповідності для безлічі вхідних слів $X_T{}^{-1}{}_{(ai)}{}''^{*}$ і реалізованої регулярної безлічі вихідних слів $R^{Y''}{}_T{}^{-1}{}_{(ai)}$, $R^{Y''}{}_T{}^{-1}{}_{(ai)}=Ri''$.

Подсеть $T^{-1}(a_i)$ будує безліч $Ri''=R^{Y''}{}_T{}^{-1}{}_{(ai)}$ від входів $X''{}_T{}^{-1}{}_{(ai)}$ подсети до її виходу $Y''{}_T{}^{-1}{}_{(ai)}$ – входу X_i'' автомата a_i або у зворотному порядку – від виходу $Y''{}_T{}^{-1}{}_{(ai)}$ подсети – входу X_i'' автомата a_i до входів $X''{}_T{}^{-1}{}_{(ai)}$ подсети.

Ri'' будується в прямому русі у відображеннях ξ' від входів $X''{}_T{}^{-1}{}_{(ai)}$ подсети до її виходу $Y''{}_T{}^{-1}{}_{(ai)}$ – входу X_i'' автомата a_i на базі операцій послідовної, паралельної композиції й композиції зі зворотним зв'язком з автоматами з $a_{i\wedge}\in A_T{}^{-1}{}_{(ai)}\cup\{a_i\}$, розширеннями алфавітів X_i'' і Y_i'', а також Y''-мінімізацією, зв'язаних в $T^{-1}(a_i)$ структурою зв'язків $\cup_{i\in I}\xi_i$.

При прямому русі ξ' для двох суміжних автоматів a_i і a_{i+1} у ланцюжку наступний a_{i+1} автомат приймає умови просування ξ' від попереднього a_i у вигляді регулярної безлічі реалізованих слів R_i (напівавтомата aR_i) в алфавіті Y_i'' і нічим не обмежений – має повна власна безліч всіх можливих вхідних слів $X_{i+1}''^{*}$ в алфавіті X_{i+1}'', у такій композиції як вхідна безліч реалізованих слів дає $R_i=R_i\cap X_{i+1}''^{*}$ для a_{i+1} при алфавітній умові $X_{i+1}''=Y_i''$.

В основі модифікованої послідовної автоматної композиції «°"» – звичайна послідовна автоматна композиції «°» при прямому просуванні у відображеннях ξ' підмережі $T^{-1}(a_i)$:

$$a_{km}=(a_k{}^{\circ''}a_m)=\begin{cases}(a_k\circ a_m),\ \text{якщо } (in_1Y_k''=Y_k)\cap(in_1X_m''=X_m)=Y_{km}\neq\varnothing\ \&\\ \quad\&\ \forall y_{km}''\in Y_{km}''(y_{km}''=(y_{km},t_{km},pb_{km})(t_{km}=t_k\cap t_m\neq\varnothing\ \&\\ \quad\&\ pb_{km}=pb_k\cdot pb_m\neq 0)),\\ \text{інакше не визначена}\end{cases}\qquad(2.5)$$

Тоді модифікована послідовна автоматна Y''-композиція «$\circ^{Y}$"» визначається на основі композиції «°"» с застосуванням внутрішньої Y''-мінімізації:

$$a_k^{Yk''}{}_m = (a_k \circ^{Y''} a_m) = (Y''\text{-}min(\Downarrow_Y(a_k)) \circ'' a_m) = (Y''\text{-}min(ak'') \circ'' a_m) = (ak''^{min} \circ'' a_m) \quad (2.6)$$

При впливі на автомат $a_k^{Yk''}{}_m$ операцій звуження $\Downarrow_Y$ і Y''-мінімізації формується напівавтомат $a_{km}^{Ym''min} = Y''\text{-}min(\Downarrow_Y(a_k^{Yk''}{}_m))$, що дає регулярну безліч $R^{Y''}{}_{ak \circ am}$ вихідних слів в алфавіті Y_m'', реалізованих послідовною композицією $a_k \circ a_m$. Як наслідок, послідовність наведених операцій представляється суперпозицією (підстановкою):

$$a_{km}^{Ym'' min} = Y''\text{-}min(\Downarrow_Y(Y''\text{-}min(\Downarrow_Y(a_k)) \circ'' a_m) \quad (2.7)$$

В основі модифікованої паралельної композиції «$\times''$» – звичайна паралельна автоматна композиція «×» при просуванні у підмережі $T^{-1}(a_i)$:

$$a_{km} = (a_k \times'' a_m) = \begin{cases} (a_k \times a_m), \text{ якщо } (ін_1 Y_{km}'' = ін_1 Y_k'' \times пр_1 Y_m'') \& (ін_1 X_{km}'' = \\ = ін_1 X_k'' \times пр_1 X_m'') \ \& \ \forall x_{km}'' \in X_{km}''(x_{km}'' = (x_{km}, t^x{}_{km}, pb^x{}_{km}) \ \& \\ \& \ (x_{km} = (x_k, x_m) \ \& \ t^x{}_{km} = t^x{}_k \cap t^x{}_m \neq \varnothing \ \& \ pb^x{}_{km} = pb^x{}_k \cdot pb^x{}_m \neq 0 \ \& \\ \& \ \forall y_{km}'' \in Y_{km}''(y_{km}'' = (y_{km}, t^y{}_{km}, pb^y{}_{km}) \ \& \ (y_{km} = (y_k, y_m) \ \& \\ \& \ (y_{km} = (y_k, y_m) \ \& \ t^y{}_{km} = t^y{}_k \cap t^y{}_m \neq \varnothing \ \& \ pb^y{}_{km} = pb^y{}_k \cdot pb^y{}_m \neq 0)), \\ \text{інакше не визначена} \end{cases} \quad (2.8)$$

Тоді модифікована паралельна автоматна Y''-композиція «$\times^{Y}$''» визначається на основі модифікованої паралельної автоматної композиції «$\times''$» с застосуванням Y''-мінімізації:

$$a_{km}^{Ym''min} = (a_k \times^{Y''} a_m) = (Y''\text{-}min(\Downarrow_Y(a_k \times'' a_m))) = (Y''\text{-}min(\Downarrow_Y(a_{km}))) = (Y''\text{-}min(akm'')) \quad (2.9)$$

Одержуваний напівавтомат $a_{km}^{Ykm''min}$ визначає регулярна безліч $R^{Y''}{}_{ak \times am}$ реалізованих композицією $a_k \times a_m$ вихідних слів в алфавіті Y_{km}''.

Модифікована операція *Y''*-композиції зі зворотним зв'язком виконується при прямому просуванні у відображеннях ξ' подсети $T^{-1}(a_i)$, операція можлива в контурах зворотного зв'язка (КЗС) при присутності вихідних алфавітів автоматів КЗС у їхніх функціях переходів-виходів. Щоб виключити утворення зовнішньої пам'яті при відсутності такої мети, виконується вимога побудови правильної схеми Глушкова, коли в КЗС присутня хоча б один автомат Мура, для НДВА це може бути реалізоване за допомогою тимчасових інтервалів.

На основі послідовної автоматної композиції «$\circ''$» і базової автоматної композиції зі зворотним зв'язком «*» визначається модифікована операція автоматної композиції зі зворотним зв'язком «$*''$», застосовувана при прямому просуванні у відображеннях ξ' подсети $T^{-1}(a_i)$ для:

– вхідного (для КЗС) автомата a_k із вхідними алфавітами:

а) X_{k1} (поза КЗС);

б) $X_k \cap (X_{k1}'' \times Y_m'') \neq \varnothing$ (усередині КЗС);

– вихідного (для КЗС) автомата a_m з вихідним алфавітом Y_m''.

Якщо t^{pres} – інтервал дії поточної події, t^{next} – інтервал дії наступної події, $t^{nextnext}$ – інтервал дії последуючої події, то модифікована операція композиції зі зворотним зв'язком «$*''$» має вигляд:

$$a_{km}=(a_k *'' a_m)=\begin{cases} (a_k * a_m), \text{ якщо } (i_{H_1} Y_k''=Y_k) \cap (i_{H_1} X_m''=X_m)=Y_{km} \neq \varnothing \;\&\; \\ \quad \&\; (i_{H_1} Y_m''=Y_m) \times (i_{H_1} X_{k1}''=X_{k1}) \cap X_k=X_{km} \neq \varnothing \;\&\; \\ \quad \&\; \forall y_{km}'' \in Y_{km}''(y_{km}''=(y_{km}, t^y_{km}, pb^y_{km}) \;\&\; (t^y_{km}=t^y_k \cap t^x_m \neq \varnothing \;\&\; \\ \quad \&\; pb^y_{km}=pb^y_k \cdot pb^x_m \neq 0)) \;\&\; \forall x_{km}'' \in X_{km}''(x_{km}''=(x_{km}, t^x_{km}, \\ \quad pb^x_{km}) \;\&\; (t^x_{km}=t^{xnext}_k \cap t^y_m \neq \varnothing \;\&\; pb^x_{km}=pb^x_k \cdot pb^y_m \neq 0 \;\&\; \\ \quad \&\; t^x_{km}=t^{xpres}_k \cap t^y_m=\varnothing \;\&\; t^x_{km}=t^{xnextnext}_k \cap t^y_m=()), \\ \text{інакше не визначена,} \end{cases} \quad (2.10)$$

Тоді на основі модифікованої автоматної композиції «*"» зі зворотним зв'язком із застосуванням внутрішньої *Y''*-мінімізації визначена модифікована автоматна *Y''*-композиція «$*^{Y}$"» зі зворотним зв'язком:

$$a_k{}^{Yk''}{}_m = (a_k *^{Y''} a_m) = (Y''\text{-}min(\Downarrow_Y(a_k)) *'' a_m) = (Y''\text{-}min(ak'') *'' a_m) = (ak''^{min} *'' a_m) \quad (2.11)$$

При застосуванні операцій звуженні $\Downarrow_Y$ і зовнішньої *Y''*-мінімізації до автомата $a_k{}^{Yk''}{}_m$ виходить напівавтомат $a_{km}{}^{Ym''min} = Y''\text{-}min(\Downarrow_Y(a_k{}^{Yk''}{}_m))$, що визначає регулярну безліч $R^{Y''}{}_{ak*am}$ вихідних слів в алфавіті Y_m'', реалізованих композицією $a_k * a_m$.

Нарешті, як і для послідовної композиції, послідовність наведених операцій представляється суперпозицією:

$$a_{km}{}^{Ym''min} = Y''\text{-}min(\Downarrow_Y(Y''\text{-}min(\Downarrow_Y(a_k)) *'' a_m) \quad (2.12)$$

2.2.5. Вихідна редукція поведінки комутації з *X''*-мінімізацією

Скорочена модель $a_i^{X''}$ автомата a_i визначає його регулярну безліч вхідних слів $R_i^{X''}$ у вхідному алфавіті X_i'' при абстрагуванні від його вихідного алфавіту Y_i''. Для безлічі вхідних слів, що транспортують, які можливі для автомата a_{igTr},, виконується $Tr^{X''} \subseteq R_i^{X''}$.

Входная скорочена модель $a_i^{X''} = (S_i, X_i'', \delta_i{}^{X''})$ автомата a_i – це напівавтомат виду, що будується при звуженні $a_i^{X''} = \Downarrow_X(a_i)$ вхідної розмітки автоматних переходів виду $((x,y),t,pb)$ до виду (x,t,pb) у розширеному алфавіті $X_i'' \subseteq X_i'' \times T_i \times Pb_i$, тоді виходить функція переходів виду $\delta_i'' : Si'' \times X_i'' \to Si''$, де $S^{X''} \subseteq S$. Після одержання напівавтомата ai'' можливе виконання операції *X''*-мінімізації станів S_i з побудовою мінімізованого вихідного напівавтомата $a_i^{X''min} = (S_i^{X''}, X''_i, \delta^{X''}{}_i)$.

Операція *X''*-мінімізації $a_i^{X''min} = X''\text{-}min(a_i^{X''})$ для $a_i^{X''}$, як і операція *Y''*-мінімізації:

а) заснована на базовій операції мінімізації детермінованого автомата *min(a)*, на додаток до неї зберегаюча недетермінізм автомата a_i і напівавтомата ai'', що виходить після звуження;

б) має особливості узгодження $t_{i\wedge\wedge}=\cap_{i\wedge\in I\wedge}t_{i\wedge}$ і $pb_{i\wedge\wedge}=\Pi_{i\wedge\in I\wedge}pb_{i\wedge}$, представленими раніше, для переходів, що ототожнюють, $(s_{i\wedge1},x_i'',s)$, $(s_{i\wedge2},x_i'',s)$ з мінімізуємих *станів* $s_{i\wedge1},s_{i\wedge2}\in S_i^{X''}$.

Як і Y''-$min(s_{i\wedge1}, s_{i\wedge2})$, нехай $s_{i\wedge1},s_{i\wedge1}\in Si''$ і $\delta^{X''}{}_i(s_{i\wedge1},i_{H1}X'')\sim\delta^{X''}{}_i(s_{i\wedge2},i_{H1}X'')$,«~» – умовна еквівалентність станів $s_{i\wedge1}$, $s_{i\wedge1}$, якщо $\delta^{X''}{}_i(s_{i\wedge1},i_{H1}X'')=\delta^{X''}{}_i(s_{i\wedge2},i_{H1}X'')$ або $\delta^{X''}{}_i(\delta^{X''}{}_i(s_{i\wedge1},i_{H1}X''),i_{H1}X'')\sim\delta^{X''}{}_i(\delta^{X''}{}_i(s_{i\wedge2},i_{H1}X''),i_{H1}X'')$.

$$X''\text{-}min(s_{i\wedge1}, s_{i\wedge2})=\begin{cases} s_{i\wedge1},\ \text{і}\ \delta^{X''}{}_i(s_{i\wedge1},X'')\sim\delta^{X''}{}_i(s_{i\wedge2},X''), \\ \quad \text{якщо}\ t_{i\wedge\wedge}=\cap_{i\wedge\in I\wedge}t_{i\wedge}\neq\varnothing\ \&\ pb_{i\wedge\wedge}=\Pi_{i\wedge\in I\wedge}pb_{i\wedge}\neq 0. \\ \{s_{i\wedge1}, s_{i\wedge2}\}\ \text{інакше.} \end{cases} \quad (2.13)$$

Аналіз підмножин станів для мінімізації в напівавтоматі $a_i^{X''}$ в операції X''-*min* може привести до мінімізованого $a_i^{X''min}$. В аналізі умов транспортування з виходу деякого автомата a_i мережі *na* передбачається визначення безлічі його вихідних слів $Tr_i^{Y''}$ в алфавіті Y_i'', які транспортуються мережею автоматів *na* до її виходів.

На базі представлених і додаткових моделей, зокрема, моделі перевірочних графів $g_{ai}\in G_A$, виконання прямого й зворотного структурно-функціонального аналізу в мережі автоматів *na* будується регулярна безліч вихідних слів $Tr_i^{Y''}$, які транспортуються мережею *na* з виходу деякого автомата a_i.

Від виходу автомата $a_i\subseteq A$ до загальних виходів мережі *na* у відповідності зі сполученнями-відображеннями $\cup_{i\in I}\ \xi_i$ формується підмережа $T(a_i)$ у складі автоматів з $A_{T(ai)}\subseteq A$, її вхідний алфавіт $X_{T(ai)}''$ визначений вихідним алфавітом Y_i'' автомата a_i відповідно до відображення $X_{T(ai)}''=\xi_i(Y_{T(ai)})=\xi_i(X''\times Y_i''\times(\times_{i'=I\setminus\{i\}}Y_{i'}''))$ з виділеним Y_i'', де $i'\in I'=I\setminus\{i\}$. Вихідний алфавіт мережі *na* задовольняє рівності

$Y''=X=Y'''\times Y_{T(ai)}''$, тут $Y_{T(ai)}''$ і Y''' – вихідні підалфавіти *na*, що залежить і не залежний від підмережі $T(a_i)$.

З повної безлічі вхідних слів $X_{T(ai)}''^*$ і реалізованої регулярної безлічі вихідних слів $R^Y_{T(ai)} \subseteq Y_{T(ai)}''^*$, підмережа $T(a_i)$ формує пряме $\xi': X_{T(ai)}''^* \to R^Y_{T(ai)}$ і зворотне $\xi'^{-1}: R^Y_{T(ai)} \to X_{T(ai)}''^*$ відповідності – відображення.

Відображення ξ' і ξ'^{-1} звужуються на безліч слів, що $\xi'_{Tr}: Tr_{T(ai)}{}^{X''} \to Tr_{T(ai)}{}^{Y''}$ і $\xi'^{-1}_{Tr}: Tr_{T(ai)}{}^{Y''} \to Tr_{T(ai)}{}^{X''}$, у цьому випадку $(Tr_{(ai)}{}^{X''} \to Tr_{(ai)}{}^{Y''}) \subseteq (X_{T(ai)}''^* \to R^Y_{T(ai)})$, $(Tr_{(ai)}{}^{Y''} \to Tr_{(ai)}{}^{X''}) \subseteq (R^Y_{T(ai)} \to X_{T(ai)}''^*)$.

Ппро визначення транспортування звуження ξ'_{Tr} є ін’єктивним, але не обов'язково функціональним через недетермінованість компонентних автоматів a_i мережі *na*, друге зворотне звуження ξ'^{-1}_{Tr} має зворотні властивості – воно необов'язково ін’єктивно, але функціонально.

Відображення ξ'_{Tr} і ξ'^{-1}_{Tr} у загальному випадку не функціональні в мережі *na* стосовно $Y_i''^*$, у тому числі через присутність співмножників X'' і $\times_{i' \in I'} Y_{i'}''$ у вхідному алфавіті $X'' \times Y_i'' \times (\times_{i' \in I'} Y_{i'}'')$ підмережі $T(a_i)$.

Автоматна мережа *na* не надлишкова, якщо автомат, що отриманий у композиції й еквівалентний мережі, є приведеним, у ненадлишковій мережі автомат a_i включає підавтомат без втрати інформації, тобто, має транспортуємим безліч слів $Tr_{(ai)}{}^{X''}$.

У вихідний для автомата a_i підмережі $T(a_i)$ на базі автоматних операцій послідовної, паралельної композиції й композиції зі зворотним зв'язком при прямому просуванні у відображеннях ξ'_{Tr} будується безліч $Tr_{(ai)}{}^{X''}$ від виходу Y_i'' автомата a_i – входу $X''_{T(ai)}$ підмережі $T(a_i)$ до її виходів $Y''_{T(ai)}$. Безліч $Tr_{(ai)}{}^{X''}$ також будується й при зворотному просуванні в відображеннях ξ'^{-1}_{Tr} від виходів $Y''_{T(ai)}$ підмережі до її входу $X''_{T(ai)}$ – виходу Y_i'' автомата a_i на основі тих же операцій композиції, модифікованих для автоматів $a_{i\wedge} \in A_{T(ai)} \cup \{a_i\}$ з урахуванням розширення алфавітів X_i'' і Y_i'' і X''-мінімізації й зв'язаних в $T(a_i)$ відповідно до структури відображень $\cup_{i \in I} \xi_i^{-1}$.

У зворотного просування $\xi'_{Tr}{}^{-1}$ безлічі, що транспортує, $_{Tr(ai}{}^{)X''}$ *в* $T(ai)$ є особливість - для двох суміжних автоматів аі i $_{ai+}1$ у ланцюжку такого просування попередній автомат аі, приймаючи умови просування від послідуючого $_{ai+}1$ у вигляді регулярної безлічі слів, що транспортують, $Tr_{T(ai+1)}{}^{X''}$ (напівавтомата aTr_{i+1}) в алфавіті X_{i+1}'', уже має власну регулярну вихідну безліч реалізованих слів R_i (напівавтоматом aR_i) в алфавіті Y_i''.

На основі раніше представленої модифікованої послідовної автоматної композиції «°"» визначена модифікована послідовна X''-композиція «$\circ^{X}$"» із застосуванням внутрішньої X''-мінімізації:

$$a_k{}^{Xm''}{}_m=(a_k \circ^{X''} a_m)=(a_k \circ''(X''\text{-}min(\Downarrow_X(a_m))))=(a_k \circ''(X''\text{-}min(am''))=(a_k \circ'' am''^{min}) \quad (2.14)$$

Операції звуження $\Downarrow_X$ і X''-мінімізації, застосовані до автомата $a_k{}^{Xm''}{}_m$, породжують напівавтомат $a_{km}{}^{Xk''min}=X''\text{-}min(\Downarrow_X(a_k{}^{Xm''}{}_m))$, що визначає регулярна безліч $R^{X''}{}_{ak\circ am}$ вхідних слів в алфавіті X_m'', вхідних для послідовної композиції $a_k \circ a_m$.

Тоді послідовність представлених операцій представлена суперпозицією:

$$a_{km}{}^{Xk''min}=X''\text{-}min(\Downarrow_X (a_k \circ''(X''\text{-}min(\Downarrow_X(a_m))))) \quad (2.15)$$

Для $a_{km}{}^{Xk''min}$ вірно $R^{X''}{}_{ak\circ am}=X_k''$*. Суперпозиція використається для визначення безлічі вхідних слів $Tr^{Xk''}{}_{ak\circ am}$, які транспортуються послідовною композицією $a_k \circ a_m$. Це застосування для компонентних (a_k і a_m) автоматів, що транспортують, aTr_k і aTr_m з урахуванням операцій звуження $\Downarrow_X$ і X"- мінімізації при поданні безлічі $Tr^{Xk''}{}_{ak\circ am}$ з одержанням напівавтомата $a_{km}{}^{Xk''min}$ має вигляд:

$$a_{km}{}^{Xk''min}=X''\text{-}min(\Downarrow_X (aTr_k \circ''(X''\text{-}min(\Downarrow_X(aTr_m))))) \quad (2.16)$$

На основі показаної вище модифікованої паралельної автоматної композиції «$\times$'» с застосуванням внутрішньої X''-мінімізації визначена модифікована паралельна X''-композиція «$\times^{X}$'»:

$$a_{km}^{Xkm''min}=(a_k\times^{X''}a_m)=(X''\text{-}min(\Downarrow_X(a_k\times'' a_m)))=(X''\text{-}min(\Downarrow_X(a_{km})))=(X''\text{-}min(akm''))\quad(2.17)$$

Напівавтомат $a_{km}^{Xkm''min}$ визначає регулярна безліч $R^{X''}{}_{ak\times am}$ реалізованих композицією $a_k\times a_m$ вхідних слів в алфавіті X_{km}''. У загальному випадку для $a_{km}^{Xkm''min}$ справедливо $R^{X''}{}_{ak\times am}=X_{km}''^{*}$. У цьому зв'язку особливість композиції, у тім, що вона має застосування для визначення безлічі транспортува паралельною композицією $a_k\times a_m$ вхідних слів $Tr^{Xkm''}{}_{ak\times am}$. Це застосування для однокомпонентних (a_k і a_m) внутрішніх автоматів, що транспортують, aTr_k і aTr_m з урахуванням операцій звуження $\Downarrow_X$ і X”- мінімізації при поданні безлічі $Tr^{Xkm''}{}_{ak\times am}$ з одержанням напівавтомата $a_{km}^{Xkm''min}$ дозволяє одержати:

$$a_{km}^{Xkm''min}=(X''\text{-}min(\Downarrow_X(aTr_k\times'' aTr_m)))\quad(2.18)$$

На основі послідовної X''-композиції «$\circ^{X}$'» і раніше представленої модифікованої автоматної композиції «*'» зі зворотним зв'язком із застосуванням внутрішньої X''-мінімізації визначена модифікована автоматна X''-композиція «$*^{X}$'» зі зворотним зв'язком:

$$a_k^{Xm''}{}_m=(a_k*^{X''}a_m)=(a_k*''(X''\text{-}min(\Downarrow_X(a_m))))=(a_k*''(X''\text{-}min(am''))=(a_k*''am''^{min})\quad(2.19)$$

У результаті застосування до автомата $a_k^{Xm''}{}_m$ операцій звуження $\Downarrow_X$ і X''-мінімізації будується напівавтомат $a_{km}^{Xm''min}=X''\text{-}min(\Downarrow_X(a_k^{Xm''}{}_m))$, що визначає регулярну безліч $R^{X''}{}_{ak*am}$ слів в алфавіті X_m'', вхідних для композиції зі зворотним зв'язком a_k*a_m.

Як наслідок, послідовність зазначених операцій представлена суперпозицією виду:

$$a_{km}^{Xk''min} = X''\text{-}min(\Downarrow_X (a_k *'' (X''\text{-}min(\Downarrow_X (a_m)))))) \quad (2.20)$$

Таким чином, для $a_{km}^{Xk''min}$ діє $R^{X''}{}_{ak \circ am} = X_k''*$. Суперпозиція може використатися для визначення безлічі вхідних слів $Tr^{Xk''}{}_{ak*am}$, що транспортують композицією зі зворотним зв'язком $a_k * a_m$. Для однокомпонентних (a_k і a_m) автоматів, що транспортують, aTr_k і aTr_m з урахуванням операцій звуження $\Downarrow_X$ і X''-мінімізації при поданні безлічі $Tr^{Xk''}{}_{ak*am}$ з одержанням напівавтомата $a_{km}^{Xk''min}$ ця безліч $Tr^{Xk''}{}_{ak*am}$ здобуває вид:

$$a_{km}^{Xk''min} = X''\text{-}min(\Downarrow_X (aTr_k *'' (X''\text{-}min(\Downarrow_X (aTr_m))))) \quad (2.21)$$

У підсумку, для довільного автомата $a_i \in A$ вихідна безліч слів $Tr^{Y}{}_i$, які розпізнаються мережею na:

а) регулярно й представлено в алфавіті Y_i;

б) включається у вихідний мінімізований напівавтомат виду $a^{Y}{}_i = (S^{Y}{}_i, Y_i, \delta^{Y}{}_i)$, тобто $Tr^{Y}{}_i \subseteq a^{Y}{}_i$, що відповідає автомату a_i;

в) перетинається з напівавтоматом $a^{X}{}_{T(ai)}$, що представляє вхідну безліч подсети $T(a_i)$, $Tr^{Y}{}_i \cap a^{X}{}_{T(ai)} \neq \varnothing$, а також перетинається з підавтоматом $aTr^{X}{}_{T(ai)}$, що транспортує поводження, тобто $Tr^{Y}{}_i \cap aTr^{X}{}_{T(ai)} \neq \varnothing$.

На основі звуження розмітки переходів до алфавітів $\cup_{i \in I} X_i$ при X-мінімізації вхідних напівавтоматів з безлічі $A^X = \cup_{i \in I} a^{X}{}_i$ вхідні безлічі слів $\cup_{i \in I} Tr^{X}{}_{T(ai)}$, розпізнаваних відповідним цим напівавтоматам підмережіми $\cup_{i \in I} T(a_i)$ і відповідні вхідні X-мінімізовані напівавтомати $A^X = \cup_{i \in I} a^{X}{}_i$ мають не більшу розмірність у порівнянні з еквівалентними автоматами $\cup_{i \in I} a_{(ai)}$ з повної автоматної композиції для підмереж $\cup_{i \in I} T(a_i)$.

Можна зробити вивід, що безліч вихідних слів $Tri^{"}$, які транспортуються мережею na з виходу a_i, формується в ній для перевірочних графів $g_{ai} \in G_A$ на базі послідовної («$\circ^{Y"}$», «$\circ^{X"}$»), паралельної («$\times^{Y"}$», «$\times^{X"}$»), зі зворотним зв'язком («$*^{Y"}$», «$*^{X"}$») композицій у модифікованих алфавітах X_i'' і Y_i'' для Y''-, X''-мінімізації.

Далі це дозволяє почати реалізацію еволюційної оптимізації Te_i (див. рис. 2.3) через підмережа $T^{-1}(a_i)$ і її автомат, що представляє, $aR_T{}^{-1}{}_{(ai)}$, і транспортування еволюційної оптимізації через подсеть $T(ai)$ і її автомат, що представляє, $aTr_{T(ai)}$) (див. рис. 2.2).

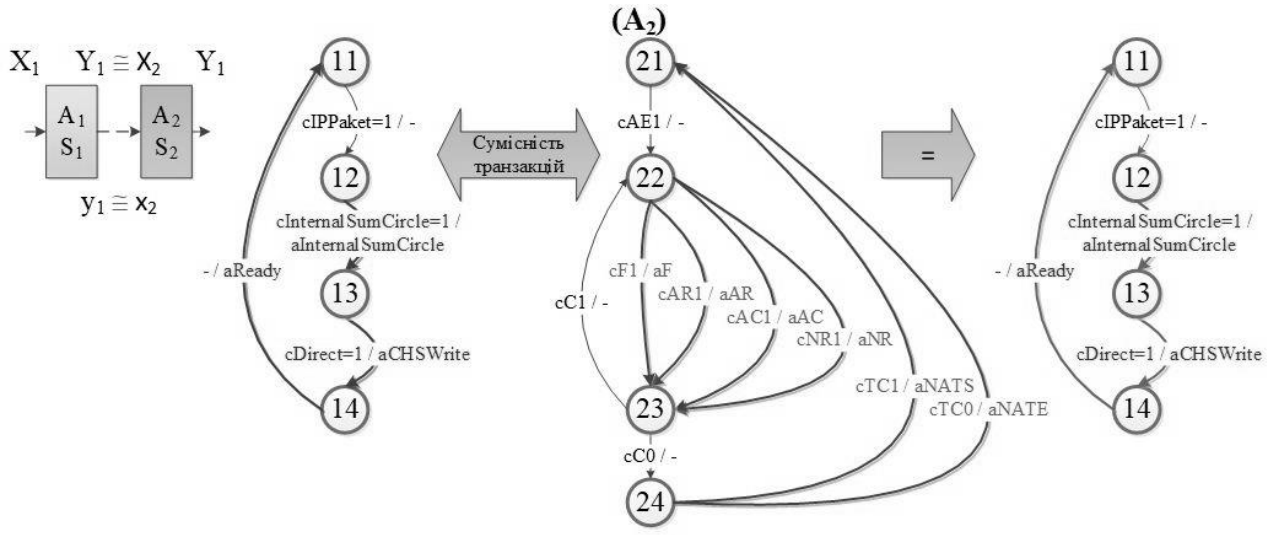


Рисунок 2.3 – Зовнішня мережева мутація для верифікуючого та транспортуючого фрагментів поведінки автоматів CHECKSUM(A_1) і NAT

2.3. Висновки

У другому розділі сформульовано завдання на дослідження верифікації комутацій агентів.

У процесі дослідження розроблено сукупність спеціальных моделей верифікації комутацій агентів, що представляють взаємодії MAC, основані на моделях мережах автоматів, з особинами визначення умов управління й спостереження комутаційної поведінки агентів у мережі MAC.

Моделі призначені для визначення умов верифікації та розробки на їхній основі методів верифікації комутацій агентів у формі автоматних примітивів, відносин і операцій над ними.

Розроблено та проаналізовано формальні моделі прояви інформації, що визначають властивості і умови розпізнавання, реалізації-транспортування комутаційної поведінки в композиції вхід-вихідних моделей автоматів.

3. ПОБУДОВА БАЗОВОІ ПРОЦЕДУРИ ВЕРИФІКАЦІЇ КОМУТАЦІЇ

3.1.Синтез мережевої процедури верифікації комутацій

Мережна процедура верифікації комутацій у мережній моделі *na* – основна складова методів верифікації для МАС, реалізована на базі коеволюцій Ce_i, Ce_{RjTi}', Ce_{RiTrk}' і тетраеволюції $Ce_{RjTiTrk}'$ мережної моделі верифікації, вузлових $aR^X_{T\text{-}1(ai)}$, $aTr^Y_{T(ai)}$ напівавтоматів і заготівлі результатів. Процедура формує мережні перевірки $TExp_{RjTiTrk}$ деякого автомата $a_i \in A$ при виконанні їхньої реалізації й транспортування в оточенні заданої автоматної мережі *na*.

3.1.1. Визначення основних задач мережевої процедури

Мережна процедура верифікації комутацій для мережі *na* побудована як зовнішня мережна тетраеволюція $Ce_{RjTiTrk}'$ на базі внутрішніх компонентних коеволюцій Ce_i і мережних коеволюцій Ce_{RjTi}', Ce_{RiTrk}' з побудовою реалізованих і що транспортують $Tp_{RjTiITrk}$, що зв'язують $Lp_{RjTiITrk}$ примітивів, верифікуючих фрагментів $Tf_{RjTITrk}$, для яких виконуються мережні операції мутації й кросінговеру, і мережних перевірок $TExp_{RjTiTrk}$ – безлічі фрагментів (в ідеалі макрофрагмента) із прийнятною повнотою $\varphi\upsilon_i$, довжиною $\lambda\varepsilon_i$ і кратністю $\theta\upsilon_i$, для яких можлива реалізація для довільного автомата $a_i \in A$ в оточенні автоматної мережі *na*

Основні в мережній процедурі - три власні завдання й одне зовнішнє компонентне завдання:

1) зовнішньої верифікації комутацій фрагментів Tf_i компонентних автоматів $a_i \in A$ за допомогою викликуваної компонентної процедури верифікації комутацій, що має в загальному випадку *NP-складність*;

2) побудови компонентних реалізованих aR_i і що транспортують aTr_i напівавтоматів, вузлових реалізуючих $T^{-1}(a_i)$ і транспортуючих $T(a_i)$ підмереж, вузлових реалізованих $aR_T{}^{-1}{}_{(ai)}$ і що транспортують $aTr_{(ai)}$ напівавтоматів, що виконується детерміноване або еволюційно й має в загальному випадку *NP-складність*;

3) зворотної реалізації мережних фрагментів верифікації Tf_{RjTi} і прямій транспортування мережних фрагментів верифікації Tf_{RiTrk}, які виконуються детерміноване або еволюційно й мають у загальному випадку *NP-складність*;

4) кооперації мережних фрагментів верифікації Tf_{RjTi}, Tf_{TiTrk}, $Tf_{RjTiTrk}$ у коеволюціях Ce_{RjTi}, Ce_{TiTrk} і тетраеволюції $Ce_{RjTiTrk}$ для безлічі автоматів $A=\cup_{i\in I} a_i$, що виконується детерміноване або эволюционно й має в загальному випадку *NP-складність*.

Зниження *NP-складності* завдань 2), 3) і 4) у більшості випадків дає пошук еволюційних методів, застосовувана в реалізованих $aR_T^{-1}{}_{(ai)}$ і що транспортують $aTr_{(ai)}$ напівавтоматах, у зворотній реалізації Tf_{rjti} і прямій транспортуванню Tf_{tiTrk} фрагментів верифікації, у коеволюціях Ce_{rjti}, Ce_{tiTrk} і тетраеволюції $Ce_{rjtiTrk}$.

3.1.2. Визначення базової мережевої процедури верифікації комутацій

При точному поданні із чотирма основними завданнями й безліччю допоміжних завдань, мережна процедура верифікації комутацій у моделі зовнішніх мережних коеволюцій Ce_{rjti}', Ce_{tiTrk}' і тетраеволюції $Ce_{rjtiTrk}'$ містить чотири стадії – препроцесорну, компонентної верифікації, мережний реалізації й мережного транспортування й наступні відповідні їхні кроки, що деталізують (див. рис. 3.1).

Крок 1. Подійно детермінованим пошуком у глибину (ширину) або еволюційним пошуком для вузлів і компонентів мережі автоматів *na*, зокрема, входів і виходів $a_i \in A$ мережі *na*, у наявних результатах, які раніше отримані, визначаються безлічі:

– структурних, вузлових зворотних реалізуючих (до входів *na*) $T^{-1}(a_i)$ і прямих транспортуючих (до виходів *na*) $T(a_i)$ підмереж;

– компонентних реалізованих aR_i і що транспортують aTr_i напівавтоматів;

– вузлових реалізованих $aR_T^{-1}{}_{(ai)}$ і що транспортують $aTr_{(ai)}$ напівавтоматів;

– мережних фрагментів верифікації Tf_{RjTi}, Tf_{TiTrk}, $Tf_{RjTiTrk}$;

– мережних перевірок верифікації $TExp_{RjTiTrk}$.

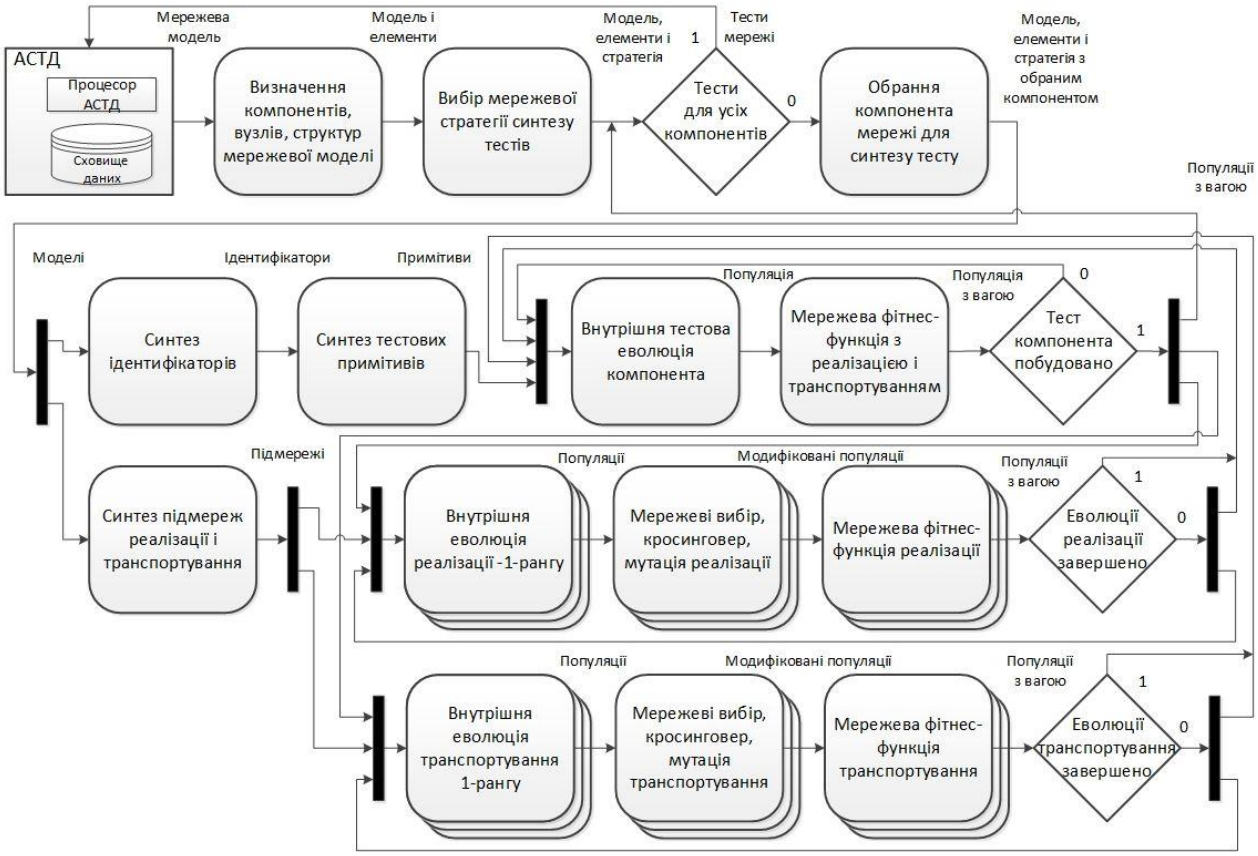


Рисунок 3.1 - Мережева процедура верифікації комутацій

Крок 2. Подійно або відповідно до завдання $Task_0$, вихідним моделям *A*, *na*, заданим параметрам, значенням повноти $\varphi\upsilon_i$, довжині $\lambda\varepsilon_i$, кратності $\theta\upsilon_i$ і попереднім оцінкам розмірності завдань і завдань *Complex(Task)*, евристичним початковим пріоритетам $Prior_0$ виконується вибір або побудова мережної стратегії, як сценарію *Script(Task)* верифікації $TExp_{na}=\cup_{i\in I}TExp_{RjTiTrk}$ для автоматів $a_i\in A$ мережі *na*.

Крок 3. Якщо перевірки верифікації $TExp_{na}=\cup_{i\in I}TExp_{RjTiTrk}$ визначені для всіх автоматів $a_i\in A$ мережі *na*, те виконується перехід у кінець до *кроку 8*, інакше виконується вибір поточного автомата $a_i\in A$, для якого відсутня перевірка верифікації $TExp_{RjTiTrk}$.

Распаралелювання після кроку 3 у кроках 4а,4б:

Крок 4а. Подійно при необхідності на основі детермінованого пошуку в глибину (ширину) або еволюційного пошуку виконується синтез або вибір

ідентифікаторів Ti_i, всіх можливих або перших зв'явившихся для кожного стану $s \in S_i$ розглянутого автомата $a_i \in A$.

Крок 4б. Подійно при необхідності детермінованим пошуком у глибину (ширину) або еволюційним пошуком виконується початковий синтез або вибір структурних, вузлових зворотних реалізуючих (до входів *па*) $T^{-1}(a_i)$ і прямих транспортуючих (до виходів *па*) $T(a_i)$ підмереж;

Крок 5а. Событийно при необхідності на основі детермінованого пошуку в глибину (ширину) або еволюційного пошуку виконується початковий синтез або вибір примітивів верифікації Tp_i і сполучних Lp_i примітивів, всіх можливих або перших появившихся для кожного стану $s \in S_i$ автомата $a_i \in A$.

Распаралелювання після кроку 4б у кроках 5ба, 5бб

Крок 5ба. Для нульового етапу $k'=0$ реалізуючої еволюції $Re_{jk'}$ рангу «$r_j=r_i-1$» для кожного поточного $a_j \in \xi^{-1}(a_i) \subseteq A$ рангу «r_j» підмережі $T^{-1}(a_i)$ детерминированно будується початкова популяція $R_{+jk'}=\varnothing$, для якої визначається початкова безліч чергових нових особин-фрагментів $R_{+jk'}'=Rf_{jk'} \cup RLf_{jk'}$, де $Rf_{jk'}=Rp_j$ – початкові реалізуючі фрагменти й $RLf_{jk'}=RLp_{jk'}=Pr_j$ – початкові сполучні фрагменти, з яких подійно при необхідності еволюційно в реалізуючій еволюції $Re_{jk'}$ виконується визначення початкових вузлових реалізованих $aR^{Y}_{T^{-1}(ai)k'}$ напівавтоматів і модифікованих реалізуючих популяцій $R_{+j+1k'}$ на базі:

а) визначення безлічей компонентних реалізованих *aRj* напівавтоматів рангу r_j, що відповідає подсети $T^{-1}(a_i)$;

б) визначення безлічей реалізованих фрагментів-особин $Rf_{jk'}$ популяцій $R_{+jk'}$, які відповідають компонентним реалізованим напівавтоматам $aR_j^{Y}{}_{k'}$ рангу r_j, $Rf_{jk'} \subseteq L(aR_j^{Y}{}_{k'})$;

в) якщо ранг $r_j<r_i-1$, у коеволюціях виду $Ce_{RjRj+1k'}$ між фрагментами-особинами всіх пар суміжних реалізуючих популяцій $R_{+jk'}$ рангу r_j і $R_{+j+1k'}$ рангу r_j+1 виконується послідовність еволюційних мережних операцій і функцій:

1) фітнес-функції φ'; вибору для кросінговера σ_κ';
2) кросінговера κ';

3) вибору для мутації σ_{μ}';

4) імунітету ϕ';

5) мутації μ';

с побудовою модифікованої популяції $R_{+j+1k'}$, що реалізується з популяції $R_{+jk'}$, після цього виконується:

1) присвоєння $r_j=r_j+1$;

2) повернення в початок поточні п. *5в*);

г) якщо «$r_j>r_{min}$», то модифікації рангів як послідовність дій «$r_{j\wedge}=r_j-1$», «$r_{i\wedge}=r_j$», «$r_j=r_{j\wedge}$», інакше, якщо «$r_j=r_{min}$», те завершення *кроку 5ба*;

д) паралельно-ветвящогося рекурсивного виклику *кроку 5ба* для нульового етапу реалізуючих еволюцій $Re_{jk'}$ молодшого суміжного рангу «r_j» для кожного поточного $a_j\in\xi^{-1}(a_{i\wedge})\subseteq A$ підмережі $T^{-1}(a_i)$ при $k'=0$ за умови попередньої модифікації рангів.

Крок 5бб. Для нульового етапу $k'=0$ транспортуючої еволюції $Tre_{kk'}$ рангу «$r_k=r_i+1$» для кожного поточного $a_k\in\xi(a_i)\subseteq A$ рангу «r_k» подсети $T(a_i)$ детерминированно будується початкова популяція $Tr_{+kk'}=\varnothing$, для якої визначається стартова безліч нових особин-фрагментів $Tr_{+kk'}'=Trf_{kk'}\cup TrLf_{kk'}$, тут $Trf_{kk'}=Trp_k$ – початкові транспортуючі фрагменти й $TrLf_{kk'}=TrLp_{kk'}\subseteq g(a_k)$ – початкові сполучні фрагменти, з яких подійно при необхідності еволюційно в транспортуючій еволюції $Tre_{kk'}$ виконується визначення початкових вузлових транспортуючих $aTr^{X}_{T(ak)k'}$ напівавтоматів і модифікованих транспортуючих популяцій $T_{+ikk'}$ на базі:

а) визначення безлічей компонентних що транспортують *aTrk* напівавтоматів рангу r_k, що відповідає підмережі $T(a_i)$;

б) визначення безлічей фрагментів-особин, що транспортують, $Trf_{kk'}$ *популяцій* $Tr_{+kk'}$, які відповідають компонентним транспортувати полуавтоматам, що, $aTr_k^{X}{}_{k'}$ рангу r_k, $Trf_{kk'}\subseteq L(aTr_k^{X}{}_{k'})$;

в) якщо ранг $r_k>r_i+1$, у коеволюціях виду $Ce_{Trk\text{-}1Trkk'}$ між фрагментами-особинами всіх пар суміжних транспортуючих популяцій $Tr_{+k\text{-}1k'}$ рангу r_k-1

і $Tr_{+kk'}$ рангу r_k виконується послідовність еволюційних мережних операцій:

1) фітнес-функції φ';
2) вибору для кросінговера $\sigma_{\kappa'}$;
3) кросінговера κ';
4) вибору для мутації $\sigma_{\mu'}$;
5) імунітету ϕ';
6) мутації μ';

при формуванні модифікованої популяції $Tr_{+k\text{-}1k'}$, що транспортує через популяцію $Tr_{+kk'}$, після чого виконується:

1) присвоєння $r_k=r_k\text{-}1$;
2) повернення в початок поточні п. *5з*);

г) якщо «$r_k<r_{max}$», то модифікації рангів як послідовність дій «$r_{k\wedge}=r_k+1$», «$r_{i\wedge}=r_k$», «$r_k=r_{k\wedge}$», інакше, якщо «$r_k=r_{max}$», те завершення *кроку 5бб;*

д) паралельно-ветвящогося рекурсивного виклику *кроку 5бб* для нульового етапу транспортуючих еволюцій $Tre_{kk'}$ старшого суміжного рангу «r_k» для кожного поточного $a_k\in\xi(a_{i\wedge})\subseteq A$ підмережі $T(a_i)$ при $k'=0$ при попередній модифікації рангів.

Сходження (одноразове) після кроків 5ба, 5бб у кроці 6а.

Крок 6а. Для нульового етапу $k'=0$ еволюції $Te_{ik'}$ рангу «$r_i=0$» для поточного $a_i\in A$ детерминированно будується початкова популяція $T_{+ik'}=\varnothing$, для якої визначається стартова безліч нових особин-фрагментів $T_{+ik'}'=Tf_{ik'}\cup Lf_{ik'}$, тут $Tf_{ik'}=Tp_i$ – початкові фрагменти верифікації – примітиви й $Lf_{ik'}=Lp_{ik'}=Pr_i$ – початкові сполучні фрагменти – примітиви, з яких подійно при необхідності эволюционно будується безліч фрагментів верифікації $Tf_{RjTiTrkk'}$ початкової популяції $T_{+RjTiTrkk'}$ тетраеволюції верифікації $Ce_{RjTiTrkk'}$, що обмежено безлічами вузлових реалізованих $aR^{Y}{}_{T}{}^{-1}{}_{(ai)k'}$ і що транспортують $aTr^{X}{}_{T(ai)k'}$ напівавтоматів, їм відповідних реалізованих

$T_{+RjTik'}$ і що транспортують $T_{+TiTrkk'}$ популяцій отриманих на базі додаткових вхідних збіжних паралельних процесів:

а) на базі знайдених раніше повністю або частково в п. 5ба безлічей вузлових реалізованих $aR^{Y}{}_{T}{}^{-1}{}_{(aj)k'}$ напівавтоматів і модифікованих реалізуючих популяцій $R_{+jk'}$ рангу $r_j=r_i-1$ виконання в коеволюціях виду $Ce_{RjTik'}$ між фрагментами-особинами всіх пар суміжних популяцій реалізуючої $R_{+jk'}$ рангу r_j і послідовності верифікації $T_{+ik'}$ рангу r_i еволюційних мережних операцій:

1) фітнес-функції φ';
2) вибору для кросінговера σ_κ';
3) кросінговера κ';
4) вибору для мутації σ_μ';
5) імунітету ϕ';
6) мутації μ';

с побудовою реалізованого вузлового напівавтомата $aR^{Y}{}_{T}{}^{-1}{}_{(ai)k'}$ і модифікованої популяції верифікації $T_{+RjTik'}$, що реалізується з боку модифікованої реалізуючої популяції $R_{+jk'}$;

б) на базі знайдених раніше повністю або частково в п. 5бб безлічей вузлових що транспортують $aTr^{X}{}_{T(ak)k'}$ напівавтоматів і модифікованих транспортуючих популяцій $Tr_{+kk'}$ рангу $r_j=r_i+1$ виконання в коеволюціях виду $Ce_{TiTrkk'}$ між фрагментами-особинами всіх пар суміжних популяцій послідовності верифікації $T_{+ik'}$ рангу r_i і транспортуючої $Tr_{+kk'}$ рангу r_k послідовності еволюційних мережних операцій:

1) фітнес-функції φ';
2) вибору для кросінговера σ_κ';
3) кросінговера κ';
4) вибору для мутації σ_μ';
5) імунітету ϕ';
6) мутації μ';

с побудовою вузлового напівавтомата, що транспортує, $aTr^{X}_{T(ai)k'}$ і модифікованої популяції $T_{+TiTrkk'}$, що транспортує через модифіковану *популяцію* $Tr_{+kk'}$;

в) на базі певних п. п. а), б) модифікованих реалізованого $aR^{Y}{}_{T}{}^{-1}{}_{(ai)k'}$ і що транспортує $aTr^{X}_{T(ai)k'}$ напівавтоматів, верифікуючих реалізованої $T_{+RjTik'}$ і що транспортує $T_{+TiTrkk'}$ популяцій побудова безлічі фрагментів верифікації $Tf_{RjTiTrkk'}$ у популяції $T_{+RjTiTrkk'}$ тетраеволюції верифікації $Ce_{RjTiTrkk'}$ событийно при мері необхідності еволюційно, зокрема, як $Tf_{RjTiTrkk'} = T_{+RjTik'} \cap T_{+TiTrkk'}$;

с подальшою модифікацією загального поточного етапу «*k'=k'+1*» для всіх поточних паралельних процесів.

Крок 6ба. Для поточного етапу *k'* реалізуючої еволюції $Re_{jk'}$ рангу «$r_j=r_i-1$» для кожного поточного $a_j \in \xi^{-1}(a_i) \subseteq A$ рангу «r_j» підмережі $T^{-1}(a_i)$ будується поточна популяція $R_{+jk'}$ на базі попередньої популяції $R_{+jk'-1} \subseteq R_{+jk'}$, з особин-фрагментів якої подійно при необхідності еволюційно в реалізуючій еволюції $Re_{jk'}$ виконується визначення поточних вузлових реалізованих $aR^{Y}{}_{T}{}^{-1}{}_{(ai)k'}$ напівавтоматів і модифікованих реалізуючих власної $R_{+jk'}$ і суміжної мережний $R_{+j+1k'}$ популяцій на основі процесів:

а) побудови безлічей нових реалізованих фрагментів-особин $Rf_{jk'}$ популяцій, що *модифікують,* $_{R+jk}$' і самих *популяцій* $_{R+jk}$', що відповідають компонентним реалізованим *напівавтоматам* $_{aRjYk}$' *рангу* rj (тобто $Rf_{jk'} \subseteq L(aR_j{}^{Y}{}_{k'})$) на основі виконання для поточних фрагментів-особин $Rf_{jk'}$ реалізуючих популяцій $R_{+jk'}$ рангу r_j послідовності еволюційних компонентних операцій і функцій:

1) фятнес-функцяї φ;
2) вибору для кросінговера σ_κ;
3) кросінговера κ;
4) вибору для мутації σ_μ;
5) імунітету ϕ;

6) мутації μ;

б) якщо ранг $r_j<r_i-1$, у коеволюціях виду $Ce_{RjRj+1k'}$ між фрагментами-особинами всіх пар суміжних реалізуючих популяцій $R_{+jk'}$ рангу r_j і $R_{+j+1k'}$ рангу r_j+1 виконується послідовність еволюційних мережних операцій і функцій:

1) фітнес-функції φ';

2) вибору для кросінговера $\sigma_{\kappa'}$;

3) кросінговера κ';

4) вибору для мутації $\sigma_{\mu'}$;

5) імунітету ϕ';

6) мутації μ';

с побудовою модифікованої популяції $R_{+j+1k'}$ рангу r_j+1, котра реалізується з популяції $R_{+jk'}$ рангу r_j, після чого виконується:

1) зсув поточного рангу присвоєнням «$r_j=r_j+1$»;

2) повернення в початок поточні п. *6ба*-а);

в) якщо «$r_j>r_{min}$», то модифікації рангів у вигляді послідовності дій «$r_{j\wedge}=r_j-1$», «$r_{i\wedge}=r_j$», «$r_j=r_{j\wedge}$»,

інакше,

якщо «$r_j=r_{min}$» і одночасно для всіх реалізуючих популяцій $R_{+jk'}$ виконується хоча б одне із двох умов:

по-перше, $R_{+jk'}=R_{+jk'-1}$;

по-друге, досягнуті критерії повноти $\varphi\upsilon_{jk'}(T_{+jk}{}^f)\geq\varphi\upsilon_{jk'}{}^{end}$, довжини $\lambda\varepsilon_{jk'}(T_{+jk}{}^f)\leq\lambda\varepsilon_{jk'}{}^{end}$, кратності $\theta\upsilon_{jk'}(T_{+jk}{}^f)\leq\theta\upsilon_{jk'}{}^{end}$,

те завершення *кроку 6ба*:

в) модифікація приватного поточного етапу $k'=k'+1$ розвитку реалізуючих еволюцій $Re_{jk'}$;

г) паралельно-ветвящогося рекурсивного виклику *кроку 6ба* для етапу k' реалізуючих еволюцій $Re_{jk'}$ молодшого суміжного рангу «r_j» для кожного поточного $a_j\in\xi^{-1}(a_{i\wedge})\subseteq A$ підмережі $T^{-1}(a_i)$ при попередній модифікації рангів (п. *6в*) і етапів (п. *6ба-г*).

Крок 6бб. Для поточного етапу k' транспортуючої еволюції $Tre_{kk'}$ рангу «$r_k=r_i+1$» для кожного поточного $a_k \in \xi(a_i) \subseteq A$ рангу «r_k» підмережі $T(a_i)$ будується поточна популяція $Tr_{+kk'}$ на основі попередньої популяції $Tr_{+kk'-1} \subseteq Tr_{+kk'}$, з особин-фрагментів якої подійно при необхідності еволюційно в транспортуючій еволюції $Tre_{kk'}$ виконується визначення поточних вузлових що транспортують $aTr^{X}_{T(ai)k'}$ напівавтоматів і модифікованих транспортуючих власної $Tr_{+kk'}$ і суміжної мережний $Tr_{+k-1k'}$ популяцій на базі процесів:

а) побудови безлічей нових фрагментів-особин, що транспортують, $Trf_{kk'}$ популяцій, що модифікують, $Tr_{+kk'}$ і самих популяцій $Tr_{+kk'}$, які відповідають компонентним транспортувати пілуавтоматам, що, $aTr_k{}^{X}{}_{k'}$ *рангу* r_k (тобто $Trf_{kk'} \subseteq L(aTr_k{}^{X}{}_{k'})$) на базі виконання для поточних фрагментів-особин $Trf_{kk'}$ транспортуючих популяцій $Tr_{+kk'}$ рангу r_k послідовності еволюційних компонентних операцій і функцій:

1) фітнес-функції φ;
2) вибору для кросінговера σ_κ;
3) кросінговера κ;
4) вибору для мутації σ_μ;
5) імунітету ϕ;
6) мутації μ;

б) якщо ранг $r_k > r_i+1$, у коеволюціях виду $Ce_{Trk-1Trkk'}$ між фрагментами-особинами всіх пар суміжних транспортуючих популяцій $Tr_{+k-1k'}$ рангу r_k-1 і $Tr_{+kk'}$ рангу r_k виконується послідовність еволюційних мережних операцій:

1) фітнес-функції φ';
2) вибору для кросінговера σ_κ';
3) кросінговера κ';
4) вибору для мутації σ_μ';
5) імунітету ϕ';
6) мутації μ';

с побудованим модифікованої популяції $Tr_{+k\text{-}1k'}$ рангу $r_k\text{-}1$, що транспортується через популяцію $Tr_{+kk'}$ рангу r_k, після цього виконується:

1) зсув поточного рангу присвоєнням «$r_k=r_k\text{-}1$»;

2) повернення в початок поточні п. *6бб*-а);

в) якщо «$r_k<r_{max}$», то модифікації рангів у вигляді послідовності дій «$r_{k\wedge}=r_k+1$», «$r_{i\wedge}=r_k$», «$r_k=r_{k\wedge}$»;

інакше,

якщо «$r_k=r_{max}$» і одночасно для всіх транспортуючих популяцій $Tr_{+kk'}$ виконується хоча б одне із двох умов:

по-перше, $Tr_{+kk'}=Tr_{+kk'\text{-}1}$;

по-друге, досягнуті встановлені критерії повноти $\varphi\upsilon_{kk'}(T_{+kk'}{}^{f})\geq\varphi\upsilon_{ik}{}^{end}$, довжини $\lambda\varepsilon_{kk'}(T_{+kk'}{}^{f})\leq\lambda\varepsilon_{kk'}{}^{end}$, кратності $\theta\upsilon_{kk'}(T_{+kk'}{}^{f})\leq\theta\upsilon_{kk'}{}^{end}$,

те завершення *шагаё 6бб*:

г) модифікація приватного поточного етапу $k'=k'+1$ розвитку транспортуючих еволюцій $Tre_{kk'}$;

д) паралельно-ветвящогося рекурсивного виклику п. 6бб для етапу k' транспортуючих еволюцій $Tre_{kk'}$ старшого суміжного рангу «r_k» для кожного поточного $a_k\in\xi(a_{i\wedge})\subseteq A$ підмережі $T(a_i)$ при попередній модифікації рангів.

Сходження після кроків 6а (одноразово), 6ба, 6бб у кроці 7.

Крок 7. Для поточного етапу k' еволюції $Te_{ik'}$ рангу «$r_i=0$» для поточного $a_i\in A$ детерминированно будується поточна популяція $T_{+ik'}$ на базі попередньої популяції $T_{+ik'\text{-}1}\subseteq T_{+ik'}$, з особин-фрагментів якої подійно при необхідності еволюційно будується безліч фрагментів верифікації $Tf_{RjTiTrkk'}$ поточної популяції $T_{+RjTiTrkk'}$ тетраеволюції верифікації $Ce_{RjTiTrkk'}$, обмежене безлічами вузлових реалізованих $aR^{Y}{}_{T}{}^{-1}{}_{(ai)k'}$ і що транспортують $aTr^{X}{}_{T(ai)k'}$ напівавтоматів, їм відповідних реалізованих $T_{+RjTik'}$ і що транспортують $T_{+TiTrkk'}$ популяцій отриманих на основі додаткових вхідних збіжних паралельних процесів:

а) побудови безлічей нових фрагментів-особин верифікації $Tf_{ik'}$ популяцій, що модифікують, $T_{+ik'}$ і самих популяцій $T_{+ik'}$ на базі виконання для поточних фрагментів-особин $Tf_{ik'}$ популяцій верифікації $T_{+ik'}$ *рангу* ri послідовності еволюційних внутрішніх компонентних операцій:

1) фітнес-функції φ;
2) вибору для кросінговера σ_κ;
3) кросінговера κ;
4) вибору для мутації σ_μ;
5) імунітету ϕ;
6) мутації μ;

б) на базі визначених раніше повністю або частково в п. *7а-а* популяцій верифікації, що модифікують, $T_{+ik'}$, а також на кроці 6ба безлічей вузлових реалізованих $aR^{Y}{}_{T}{}^{-1}{}_{(aj)k'}$ напівавтоматів і модифікованих реалізуючих популяцій $R_{+jk'}$ *рангу* rj=ri-1 виконання в коеволюціях *виду* $Ce_{RjTik'}$ між фрагментами-особинами всіх пар суміжних популяцій реалізуючої $R_{+jk'}$ рангу r_j і верифікуючої $T_{+ik'}$ *рангу* r_i послідовності еволюційних мережних операцій:

1) фітнес-функції φ';
2) вибору для кросінговера σ_κ';
3) кросінговера κ';
4) вибору для мутації σ_μ';
5) імунітету ϕ';
6) мутації μ';

с побудовою реалізованого вузлового напівавтомата $aR^{Y}{}_{T}{}^{-1}{}_{(ai)k'}$ і модифікованої популяції верифікації $T_{+RjTik'}$, що реалізується з боку модифікованої реалізуючої популяції $R_{+jk'}$;

в) на основі знайдених раніше повністю або частково в п. *7а-а* популяцій, що модифікують, $T_{+ik'}$,а також *на* кроці 6бб безлічей вузлових що транспортують $(aTr^{X}{}_{T\ ak)k'}$ напівавтоматів і модифікованих транспортуючих популяцій $Tr_{+kk'}$ *рангу* $rj=ri$+1 виконання в коеволюціях виду $Ce_{TiTrkk'}$ між

фрагментами-особинами всіх пар суміжних популяцій верифікації $T_{+ik'}$ *рангу* r_i і транспортуючої $Tr_{+kk'}$ рангу r_k послідовності мережних операцій:

1) фитнесс-функции φ';
2) вибору для кроссинговера σ_{κ}';
3) кроссинговера κ';
4) вибору для мутації σ_{μ}';
5) імунітету ϕ';
6) мутації μ';

с побудовою вузлового напівавтомата, що *транспортує,* $_{aTrXT(ai)}$ k' і модифікованої *популяції* $_{T+TiTrkk}$', що транспортується через модифіковану *популяцію* $_{Tr+kk}$';

г) на базі знайдених п. п. а), б), в) модифікованих реалізованого $aR^{Y}{}_{T}{}^{-1}{}_{(ai)k'}$ і що транспортує $aTr^{X}{}_{T(ai)k'}$ напівавтоматів, верифікуючих реалізованої $T_{+RjTik'}$ і що транспортує $T_{+TiTrkk'}$ популяцій побудова безлічі фрагментів верифікації $Tf_{RjTiTrkk'}$ у популяції $T_{+RjTiTrkk'}$ тетраеволюції верифікації $Ce_{RjTiTrkk'}$ подійно при необхідності еволюційно, зокрема, як $Tf_{RjTiTrkk'} = T_{+RjTik'} \cap T_{+TiTrkk'}$;

д) якщо для популяції верифікації $T_{+ik'}$ виконується хоча б одне з умов:

по-перше, $T_{+ik'}=Tr_{+ik'-1}$;

по-друге, досягнуті встановлені критерії повноти $\varphi\upsilon_{ik'}(T_{+ik}{}^{f}) \geq \geq \varphi\upsilon_{ik'}{}^{end}$, довжини $\lambda\varepsilon_{ik'}(T_{+ik}{}^{f}) \leq \lambda\varepsilon_{ik'}{}^{end}$, кратності $\theta\upsilon_{ik'}(T_{+ik}{}^{f}) \leq \theta\upsilon_{ik'}{}^{end}$,

те завершення *кроку 7*.

с подальшої модифікацією загального поточного етапу «*k'=k'+1*» для всіх поточних паралельних процесів і переходом до *кроку 6ба*.

Крок 8. Завершення еволюції.

Фітнес-функції мережевої еволюційної процедури, також на основі використання властивостей сумісності та ідентифікації, критеріїв повноти $\varphi\upsilon_i$, довжини $\lambda\varepsilon_i$ та кратності $\theta\upsilon_i$ верифікації, знаходять попередження та екстремуми всіх формованих фрагментів Tf_i, Tf_{rjti}, Tf_{tiTrk}, $Tf_{rjtiTrk}$ и популяції T_{+i}, T_{+rjti}, T_{+tiTrk}, $T_{+rjtiTrk}$

загалом у просторі значень приватних критеріїв та загального адитивно-мультиплікативного критерію фітнес-функцій.

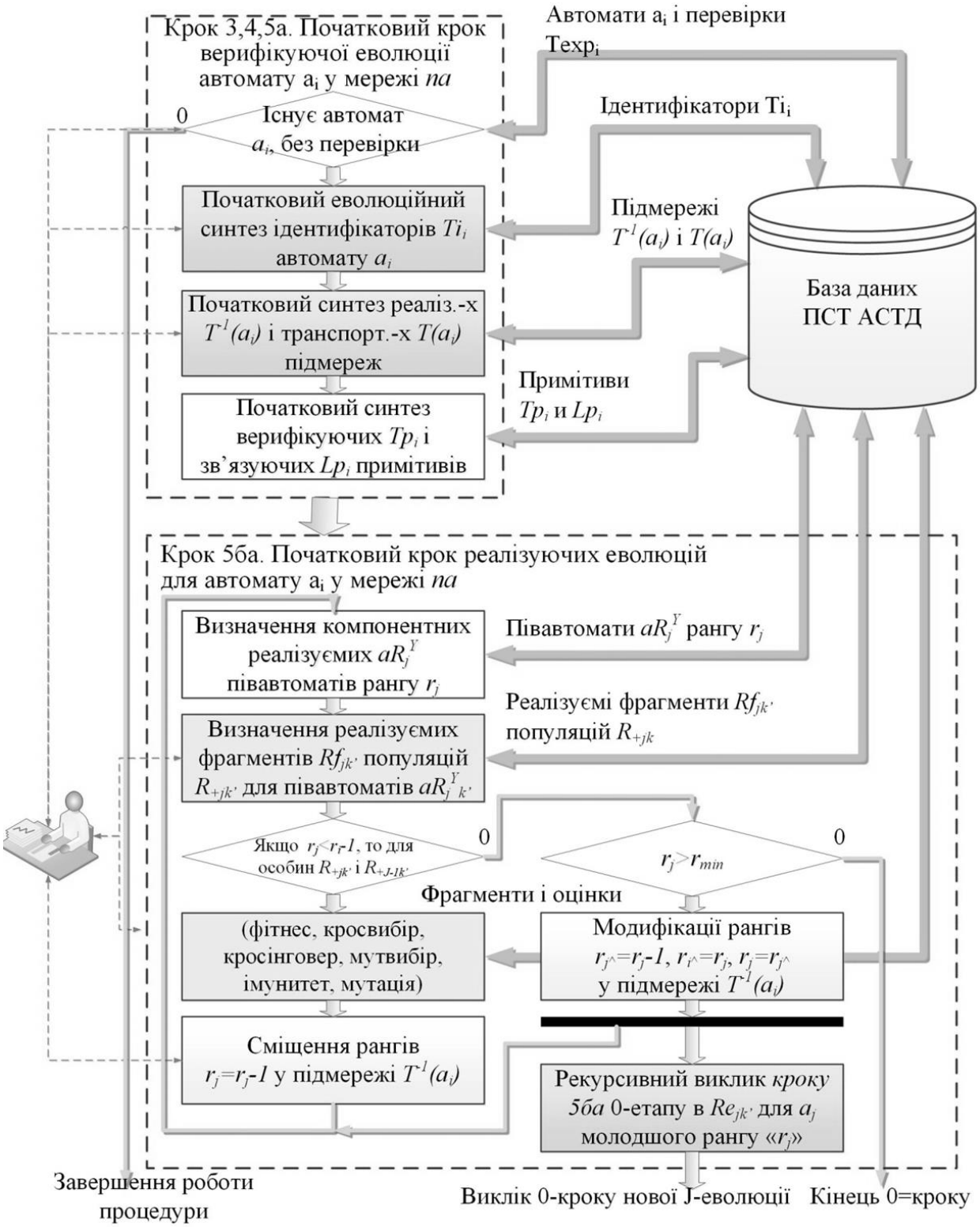


Рисунок 3.2 – Початковий фрагмент стадій верифікуючої та реалізуючої еволюцій мережевої процедури

Мережеві приватні процедури на вирішення чотирьох завдань дозволили з допомогою декомпозиції поліноміально знизити NP-сложность синтезу поведінкових перевірок верифікації як автоматних експериментів $TExp_i$..

Її зменшення досягається пошуком еволюційних підходів, що використовується як базовий для вирішення задач в еволюції Te_i, внутрішньої коеволюції Ce_i, зовнішніх реалізуючих Re_j і транспортних Tre_k еволюціях, зовнішніх коеволюціях Ce_{RjTi}, Ce_{TiTrk} і тетраеволюції $Ce_{RjTiTrk}$.

3.2. Аналіз результатів побудови процедури верифікації

Мережний метод верифікації комутацій на базі мережний тетраеволюці\ верифікації будує перевірку кінцевої довжини, що поліноміально залежить від розмірності вхідної мережної моделі й ступеня її декомпозиції k, довжина не менше $k((m+1)(n/k)^2+((m+2)(n/k)+1)((n/k)-1)$, не більше $k((m-1+((n/k)((n/k)--1))/2)(n/k)+(m(n/k)^2((n/k)-1))/2+(m-1)((n/k)-1)((n/k)-2)+2^{n/k-1})$.

У мережній процедурі верифікації комутацій на базі еволюційного походу реалізоване й імунізоване поводження, що транспортує, Tf_{jIk} будується у відкритих просторах поводження тетраеволюцій $Ce=\cup_{i\in I}\ Ce_{jik}$, усіх $a_i \in A$, сумарно $(k^2((n/k)(2m+1)+m((n/k)-1)(2(n/k)+l)+((lm)^{(n/k)}2^{(n/k)-1})(2(n/k)+3)-3+ +2^{(n/k)-1}(4m+(n/k))))/log_2(2nm/k)$ умовних операцій зі станами пам'яті, обчислювальної складності, що знижена до $log_2(2nm/k)$ раз при порівнянні з вычислителной складністю побудови поводження верифікації у мережній детермінованій процедурі за рахунок кращої збіжності еволюцій (див. рис. 3.3, 3.4). Тут $k=|I|$ – коефіцієнт декомпозиції моделі tn, або число компонентів мережі na.

а) б)

Рисунок 3.3 – Залежність обчислювальної складності еволюційної верифікації від ступеня декомпозиції

3.3. Висновки

Мережева процедура верифікації комутацій МАС у розширеному класі мережевих помилок, що має особливості побудови популяцій перевірок, що реалізуються, транспортуються фрагментів у композиції еволюцій та багаторазового використання вузлової поведінки верифікації на основі мережевої моделі верифікації, дозволяє скоротити обчислювальну складність верифікації до $log_2(2nm/k)$ у порівнянні з мережевою детермінованою процедурою.

При допустимому паралельному представленні та реалізації мережевої еволюційної процедури, доки екземпляр фрагмента Tf_{tik} або популяції T_{+ik} еволюції Te_{ik} модифікується, він не зв'язується в чергових операціях, функціях, кроках еволюції. Te_{ik}.

4. РЕАЛІЗАЦІЯ ВЕРИФІКАЦІЇ КОММУТАЦИЙ

Верифікація комутацій агентів МАС є важливою часткою аналізу взаємодій, її реалізація представлена у вигляді, наборів, послідовностей і структур і алгоритмів аналізу комутацій й використовується для:

- верифікації проектів МАС на відповідність специфікаціям у процесі структурно-алгоритмічного проектування;
- перевірки реалізацій засобів верифікації комутацій на відповідність верифікованим проектам МАС при їх виготовленні;
- перневірки екземплярів реалізацій МАС при експлуатації.

Застосування засобів верифікації комутацій у спеціальному комплексі верифікації зменшує час налагодження проектів, вихідного контролю й відновлення працездатності МАС.

4.1.Побудова блок схеми комплексу верифікації

4.1.1. Основні блоки і функції

Режими роботи засобів комплексу верифікації комутацій

Сеанс роботи користувача з комплексом верифікації містить реєстрацію, визначення умов роботи, завантаження програмного, інформаційного забезпечення, ініціалізацію програм, вибір режиму й сценарію роботи.

Засоби й програми комплексу верифікації комутацій потрібні при підготовці контрольного забезпечення для перевірки проектів МАС, їх верифікації, контролю реалізацій і є визначальними програмами.

Програми верифікації комутацій містять режими: визначення характеристичних властивостей; визначення комутацій; побудови абстрактних і мережних експериментів комутацій, реалізації й транспортування поведінки.

Режим характеристичних властивостей дозволяє визначити властивості комутацій.

Режим експериментів на основі властивостей комутацій перевіряє МАС на підставі покриття властивостей автоматної моделі.

У відповідності з вирішуваними завданями в блоці комплексу верифікації задані дев'ять основних підблоків (див. рис. 4.1.), що виделені за функціональною ознакою:

1) інтерфейсу з комплексом;
2) створення, редагування моделей верифікації;
3) верифікації спеціальних і загальних моделей;
4) вхідного контролю;
5) поновлення і створення XML- і XSL-, JSON-об'єктів;
6) підтримки фреймів;
7) бібліотеки моделей верифікації;
8) історії верифікації спеціальних і загальних моделей;
9) контекстної підказки і допомоги.

Підблоки виконують функції, що реалізуються методами і обробниками подій для об'єктів і їх властивостей у DOM блоці верифікації, взаємодіють з іншими підблоками і блоками комплексу, зовнішніми системами відповідно до власних властивостей, методів і форматів даних для їх інтерфейсів.

Підблок створення, редагування, верифікації моделей

У підблоків створення, редагування, верифікації моделей є чотири групи функцій:

- формування шаблонів;
- створення, введення та редагування;
- верифікації;
- автозаповнення по шаблонах в режимі підказки.

Кожній групі функцій відповідає окремий компонент.

Компоненти формування шаблонів, створення, введення та редагування, верифікації та автозаповнення по шаблонах в режимі підказки для об'єктів верифікації реалізують створення, редагування і автозаповнення моделей

верифікації і, у відповідності до призначення для визначених, відкритих або віртуальних вікон (фреймів), реалізують:

- формування шаблонів форм з органами управління - об'єктами DOM-документа для моделей, структурованих за типами моделей, за типами і форматам значень полів - значень властивостей об'єктів DOM;
- інтерактивні створення, введення, редагування і сортування значень полів органів управління - об'єктів DOM-документа обраних форм;
- автоматизовану верифікацію значень полів органів управління, автозаповнення в режимі підказки значень полів по шаблонам і законам функціонування моделей;
- редагування бібліотеки моделей для об'єктів верифікації в цілому.

Підблок верифікації спеціальних і загальної моделей

У підблоків верифікації спеціальних і загальної моделей є три групи функцій, окремих для автоматних, реляційних, часових моделей:

- об'єктів;
- операцій і відносин;
- синхронізації;
- логічного висновку і верифікації.

У кожній групі функцій є окремий компонент.

Компоненти об'єктів, операцій і відносин, синхронізації, логічного висновку і верифікації дають автоматизовані формування властивостей, обробників і подій об'єктів, сигнатури операцій і їх правил, відносин об'єктів і операцій, часової і подієвої синхронізації, автоматного, реляційного, графічного виведення і верифікації в відповідно до призначення зумовлених, відкритих або віртуальних вікон (фреймів),

- визначення і формальну інтерпретацію сигнатур операцій і відносин для моделей верифікації відповідно до автоматним, реляційним, графічним представленням по заданих типів і форматів значень об'єктів DOM-документа верифікації;

- визначення і формальну інтерпретацію часової, подієвої причинно-наслідкового синхронізації для моделей верифікації відповідно до автоматним, реляційним, графічним представленням по заданих типів і форматів значень об'єктів DOM-документа верифікації;
- визначення і формальну інтерпретацію процедур і кроків верифікації для моделей верифікації відповідно до автоматним, реляційним, графічним представленням по заданих типів і форматів значень об'єктів DOM-документа верифікації.

Підблок вхідного контролю

У підблоків вхідного контролю специфікацій для моделей верифікації є чотири групи функцій контролю:

- контролю вхідного алфавіту;
- контролю лексем;
- контролю синтаксису;
- контролю семантики.

Кожній групі функцій відповідає окремий компонент.

Компоненти контролю вхідного алфавіту, контролю лексем, контролю синтаксису і контролю семантики дають автоматизований вхідний контроль значень об'єктів, властивостей, обробників, подій, відносин, моделей верифікації і, відповідно до призначення визначених, відкритих або віртуальних вікон (фреймів), реалізують :

- автоматичний контроль символів, що вводять значень полів органів управління DOM-документа для об'єктів, властивостей, обробників, подій, відносин - значень властивостей об'єктів DOM-документа для моделей верифікації на приналежність заданому алфавітом;
- автоматичний контроль лексем, які подаються значеннями полів органів управління для об'єктів, властивостей, обробників, подій, відносин - значеннями властивостей об'єктів DOM-документа;
- автоматичний контроль синтаксису для значень наборів полів органів управління для об'єктів, властивостей, обробників, подій, відносин -

структур поведінки моделей верифікації відповідно операціям, законам обраного формалізму автоматного, реляційного, графічного представлення;

- автоматизований інтерактивний контроль семантики для моделей верифікації відповідно до формул і процедур обраного формалізму автоматного, реляційного, графічного представлення.

4.1.2. Інтерфейсні зв'язки

Підблоки і компоненти блоку верифікації в ході функціонування й комутації здійснюють взаємний і зовнішній обмін подіями, запитами, методами і обробниками подій, передачу управління відповідно до власних функцій, обмін вхід-вихідними даними з певними властивостями і форматами. Склад, властивості, формати і механізми спеціальних інтерфейсів для підблоків і компонентів представляють п'ять видів взаємодії й комутації:

- події, явно або неявно передаються і володіють механізмом умовчання для кожного з об'єктів DOM-документа, а також механізмом «спливання» від молодших до старших об'єктів ієрархії DOM-документа;
- запити на виконання дії, зовнішнього по відношенню до підблоків або компоненту, мають вигляд виклику методу або обробника події деякого зовнішнього об'єкта DOM-документа;
- вхід-вихідні дані, спеціалізовані для процесів інтерфейсу, створення, редагування і автозаповнення об'єктів, властивостей, обробників, подій, відносин, вхідного контролю, оновлення та створення XML-, XSL-, JSON-об'єктів, підтримки фреймів, бібліотеки дерев аналізу моделей, історії дерев аналізу, контекстної підказки і допомоги складі автоматних, табличних, реляційних, графічних специфікацій;
- методи і обробники подій блоку верифікації, спеціалізовані для кожної з представлених раніше функцій кожного з підблоків і компонентів;

– передача управління, яка має вигляд запитів з параметрами, що відображають взаємну передачу (повернення) активності з викликом представлених раніше функцій підблоків і компонентів.

4.2.Розробка базових структур і алгоритмів верифікації комутацій

4.2.1. Розробка об’єктів верифікації

Об'єктна модель

У ієрархії програмних класів моделі мережі автоматів є три рівні:

– модель мережі;

– модель автомата;

– модель виводу.

Об'єкти типу «автомат»

Властивість-об'єкт другого рівня типу «автомат» - об'єкт класу «class Automata», який представляє:

– властивість-ідентифікатор автомата - символьний ідентифікатор типу «String Automata.Id[3]», що представляє масивом трьох рядків символів «3×32Char», що визначають для автомата:

  а) ім'я макроавтомата;

  б) ім'я автомата в даному макроавтоматі;

  в) ім'я функції (функціонального, функціонально-параметричного відображення) автомата;

– властивості-об'єкти третього рівня - виводи-об'єкти автомата класу «class Automata.IOPut»;

– методи аналізу автомата, що визначають пряме, зворотне, синхронне, асинхронне, алфавітне, структурне, функціональне, параметричне, табличне, процедурне, аналітичне, текстове, бінарне відображення в змінні алфавітів типу «Automata.function(inSet)» і «Automata.function^{-1}(outSet)».

Об'єкт комутації типу «з'єднання»

Властивість-об'єкт комутації другого рівня типу «з'єднання» - це об'єкт класу «class Connect», який представляє:

- властивість-ідентифікатор з'єднання - символьний ідентифікатор типу «String Connect.Id[3]», який представляє масив трьох рядків символів «3×32Char», що визначають для мережі:
  а) ім'я шини;
  б) ім'я лінійної або ланцюга, що розгалужується, у даній шині;
  в) ім'я функції (функціонального, функціонально-параметричного відображення) ланцюга;
- властивості-об'єкти третього рівня - сукупність виводів-об'єктів автомата класу «class Automata.IOPut», доступних, за допомогою поліморфізму, або інтерфейсів з об'єктів типу «Automata»;
- методи аналізу з'єднання, що визначають пряме, зворотне, синхронне, асинхронне, алфавітне, зокрема, алфавітної арифметики, структурне, функціональне, параметричне, табличне, процедурне, аналітичне, текстове, бінарне відображення, а також повторювач у змінні алфавітів типу «Connect.function(inSet)» і «Connect.function^{-1}(outSet)».

Об'єкт типу «вивід»

Властивість-об'єкт третього рівня типу «вивід» - це об'єкт класу «class IOPut», що представляє:

- властивість-ідентифікатор виводу - символьний ідентифікатор типу «String IOPut.Id[4]», який представляє масив чотирьох рядків символів «4×32Char», що визначають для мережі):
  а) ім'я макроавтомата;
  б) ім'я автомата в даному макроавтоматі;
  в) ім'я виводу даного автомата;
  г) ім'я функції (функціонального або функціонально-параметричного відображення) виводу;
- властивості алфавітів виводу – множину припустимих алфавітів

висновку типу «array IOPut.DirAlph[12]», яка представляє масивом множин «12×32Char», що визначають для мережі алфавіт символів виводу для прямого/зворотного напрямку й высокоимпедансного стану виводу;

- властивості напрямку виводу - множину припустимих напрямків функціонування виводу типу «array IOPut.Dir[3]», яка представляє двома бітами «2Bit», що визначають для виводу автомата мережі пряме/зворотний напрямок і высокоимпедансний стан виводу;
- методи аналізу виводу, які визначають пряме, зворотне, синхронне, асинхронне, алфавітне, функціональне, параметричне, табличне, процедурне, аналітичне, текстове, бінарне відображення (й повторювач) у змінні алфавітах типу «IOPut.function(inValue)» і «IOPut.function^{-1}(outValue)».

Ієрархію і для моделей верифікації і тест-моделей наведено у рис. 4.1.

Поділ об'єктів в ієрархії, що надає поділ властивостей і оброблювачів, дає можливість оптимального аналізу алфавітів, функцій, структур і параметрів, обумовленого особливостями: властивостей аналізованих мереж і тест-моделей; методів аналізу; конкретних розв'язуваних задач аналізу.

Об’єктно-реляційна модель бази даних комплекса верифікації наведено у рис. 4.2.

4.2.2. Розробка моделей, класів та алгоритму верифікації

Алгоритм верифікації поведінки комутації після завантаження автоматних моделей мережі і їхніх характеристичних властивостей на першому етапі итеративно синтезує перевірки верифікації для кожного окремого автомату мережі.

Для цього викликається зовнішня стосовно програми верифікації процедура синтезу вхідного й вихідного дерев, на підставі яких наступна внутрішня процедура визначає реалізовану й трансльовану поведінку (див. рис. 4.3).

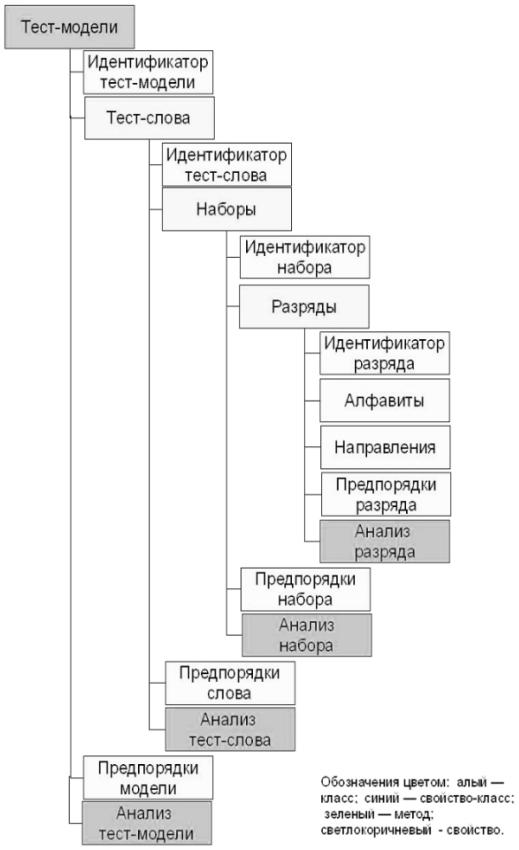


Рисунок 4.1 – Ієрархія моделей верифікації та тестування

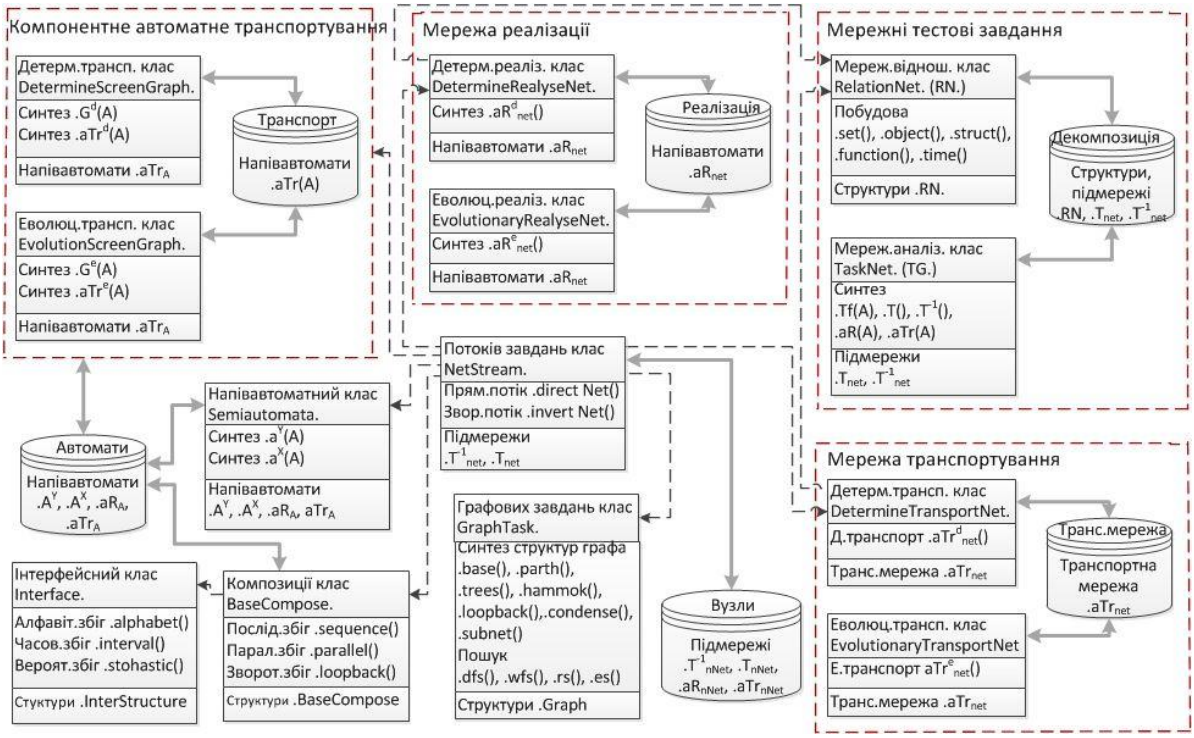


Рисунок 4.2 – Об’єктно-реляційна модель бази даних комплекса верифікації

Процедура, що викликувана на чергової ітерації, будує автоматного експерименту верифікації для реалізованої поведінці. Остання в ітерації викликувана процедура дає розпізнавання контрольного експерименту, розміщаючи його в розпізнаваної поведінці автомата.

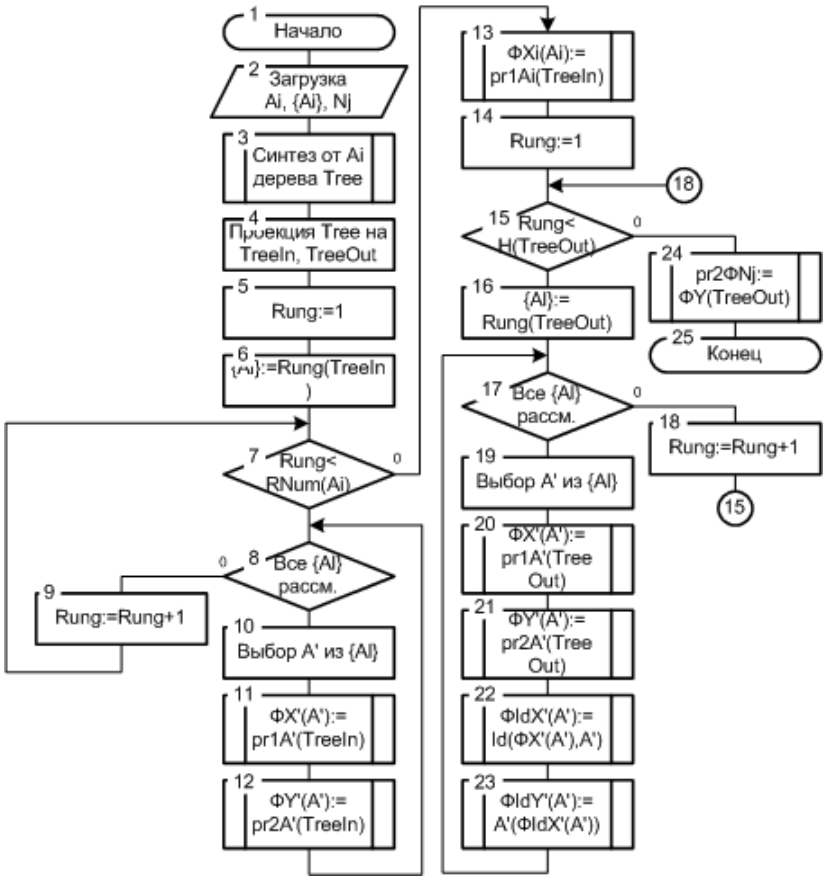


Рисунок 4.3 – Схема алгоритму реалізованої й трансльованої поведінки комутації

Далі на другому етапі ітеративно визначається повне реалізована поведінка мережі на кожному її автоматі й шукаються мінімальні покриття примітивів верифікації автомата цією поведінкою. Після знаходження локальних мінімальних покриттів виконується завдання визначення мінімальних покриттів для автоматів всієї мережі. Розподіл даних ціх завдань у реальності умовний.

4.3. Експериментальні випробування програм верифікайції поведінки

Експериментальні випробування базових програм, що реалізують розроблені моделі, кроки процедур, алгоритми й структури даних для верифікації комутацій, зроблено для мережно-транспортних й прикладних об'єктів MAC і показано у таблиці 4.1.

Для обраних об'єктів експериментальні випробування програм автоманої верифікації комутацій у порівнянні з використанням перебору у перевірці показали зниження довжини і обчислювальної складності більш ніж на порядок зі збереженням повноти й вірогідності перевірки (див. табл. 4.1).

Таблиця 4.1 – Експериментальні значення складності (часу) верифікації комутацій

Механізм	Ступінь декомпозиції	Вхідна складність	Складність
Send	5	26	2495368
Receive	4	21	2013969
Cript	3	17	14701736
Roaming	2	8	11612
Client	5	27	20751408

Застосування засобів верифікації комутацій скорочує час підготовки контролю MAC та відновлення їх працездатності за рахунок зменшення довжини перевірок.

Декомпозиція також підвищує гнучкість організації ідентифікації завдяки обліку особливостей реалізації PIC.

4.4. Висновки

У четвертому розділі розроблено базові програмні засоби верифікації комутацій при виконанні контроля MAC.

Блок-схема комплексу верифікації МАС містить базові засоби верифікації комутацій, визначено основні режими роботи, зокрема, у мережі автоматного класу, функції, блоки й компоненти, інтерфейсні зв'язки, визначено базові алгоритми та структури даних.

Експериментальні випробування програм верифікації показали, що для МАС рівня складності програмного агента при зниженні обчислювальної складності на більш, ніж порядок зниження довжини верифікації становить 10-30%. За рахунок цього скорочується час відновлення працездатності МАС.

ВИСНОВОК

У магістерській роботі вдосконалена модель верифікації для комутації агентів МАС. Модель верифікації для комутації агентів заснована на застосуванні автоматних моделей і методів експериментів аналізу.

В ході виконання роботи отримані наступні результати:

1. Проведено дослідження технологій і протоколів комутації РІС, у тому складі й на основі інтерфейсу сокетів, програмних компонентів транспортно-мережевого рівня, механізмів протоколу HTTP, технологій XML, AJAX, JSON прикладного рівня та експериментів з автоматними моделями. Зроблено висновок про можливость узагальнення композиційного автоматного підходу до верифікації для комутації агентів, що дозволяє підвищити її повноту, скоротити їх довжину і складність.

2. Удосконалено модель верифікації комутації МАС, що заснована на автоматних експериментах, відрізняється автоматними структурами ідентифікації та розпізнавання властивостей взаємодії, керованості та наочності поведінки у мережах автоматних моделей. Модель верифікації комутацій призначена для формального визначення умов організації експериментів розпізнавання комутаційної поведінки у автоматної мережі, яка модулює МАС та дозволяє скоротити час контролю для методів їх верифікації.

3. Розроблено базові кроки процедур верифікації комутації, які можуть використовуватися для підготовки контрольного забезпечення МАС.

4. Експериментальні випробування й застосування основних процедур показали, що для МАС прикладного рівня складності Веб-сервісів при зниженні обчислювальної складності аналізу на більш, ніж порядок зниження його довжини становить 10-15%.

5. Модель, метод, процедури, алгоритми прихованої поведінки РІС можуть знайти застосування в навчальному процесі кафедри КІСМ.

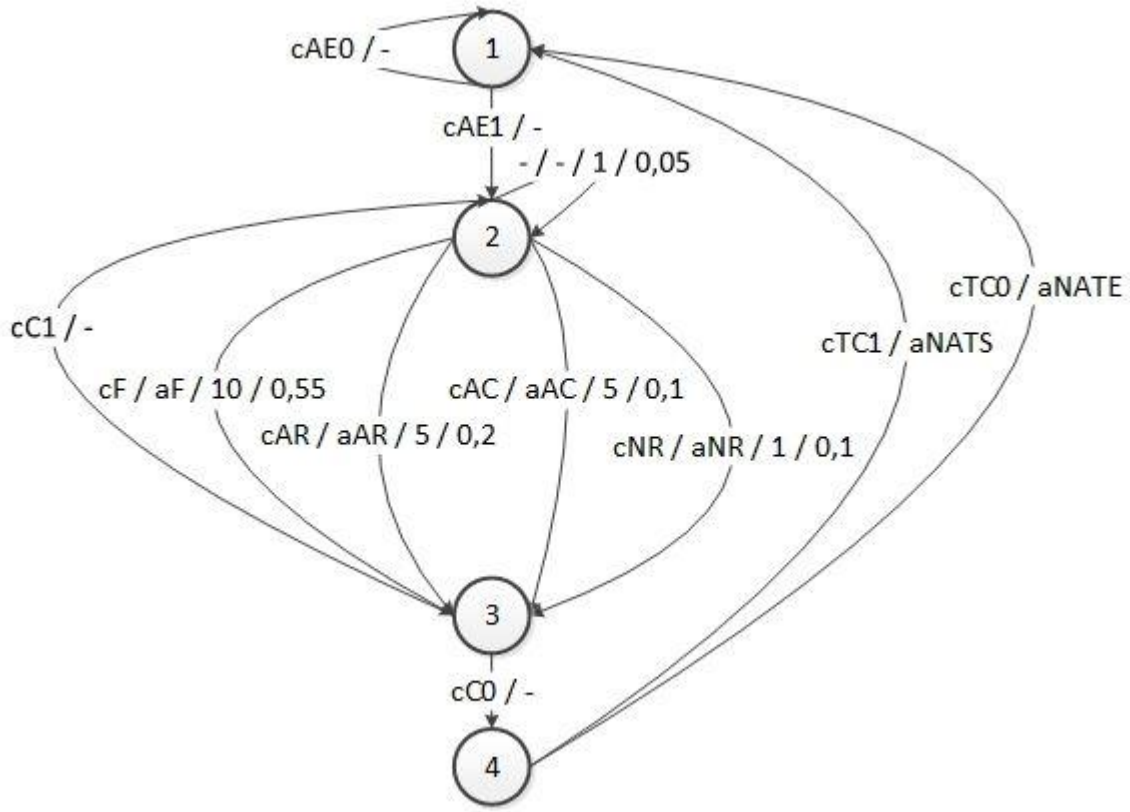
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Электронный ресурс http://unixhelp.ed.ac.uk/CGI/man-cgi?socket+2
2. Visual Component Library. https://en.wikipedia.org/wiki/Visual_Component_Library
3. Электронный ресурс https://docs.microsoft.com/en-us/windows/win32/winsock/windows-sockets-start-page-2
4. Стивенс У. Ричард Сетевое программирование UNIX. Энглвуд Клиффс, Нью-Джерси: Прентис-Холл. 1990.
5. Фейт Сидни TCP/IP. Архитектура, протоколы, реализация (включая IP версии 6 и IP Security) — 2-е изд. — М.: Изд-во «Лори», 2003. — 424 С.
6. С.В. Зыков Лекция: Компонентное программирование в .NET / Введение в теорию программирования. Объектно-ориентированный подход. intuit.ru. http://www.intuit.ru/department/se/tppobj/17/ Дата обращения: 25 октября 2010.
7. Роберт Дж. Оберг. Технология COM+. Основы и программирование = Understanding and Programming COM+: A Practical Guide to Windows 2000 First Edition. — М.: «Вильямс», 2000. — С. 480.
8. Электронный ресурс https://intuit.ru/studies/courses/571/427/lecture/9705?page=1
9. Электронный ресурс HTTP/3: the past, the present, and the future (англ.). The Cloudflare Blog (26 September 2019).
10. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2010. — 656 с.
11. Стивен Хольцнер. Ajax Библия программиста = Ajax Bible. — М.: Диалектика, 2009. — С. 553.
12. эвид Сойер Макфарланд. Новая большая книга CSS = CSS: The Missing Manual. — Санкт-Петербург: Питер, 2017. — 720 с.
13. Ouzzani, M., Bouguettaya A. Semantic Web Services for Web Databases. Springer Science+Business Media, 2011. 155 p.

14. Электронный ресурс https://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf
15. Электронный ресурс https://coderlessons.com/tutorials/akademicheskii/vyuchi-uddi/uchebnik-po-uddi
16. Дэвид Хантер, Джефф Рафтер, Джо Фаусетт, Эрик ван дер Влист, и др. XML. Работа с XML, 4-е издание = Beginning XML, 4th Edition. — М.: «Диалектика», 2009. — 1344 с.
17. Кришнамурти К., Рексфорд Дж. Web-протоколы. Теория и практика. — М.: ЗАО «Издательство БИНОМ», 2002. — 592 С.
18. Электронный ресурс https://www.developer.com/design/web-services-security-and-more-the-global-xml-web-services-architecture-gxa/
19. Электронный ресурс http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html
20. Электронный ресурс https://developer.mozilla.org/ru/docs/Learn/JavaScript/Objects/JSON
21. Брауэр В. Введение в теорию конечных автоматов. – М.: Радио и связь, 1987. – 392 с.
22. Богомолов А.М., Салий В.Н. Алгебраические основы теории дискретних систем. – М.: Наука, 1997. – 386 с.
23. Глушков В.А. Абстрактная теория автоматов // Научный журнал «Успехи математических наук», т. XVI, вып. 5 (101). – Москва. – 1961. – С.3–62.
24. Богомолов А.М., Грунский И.С., Сперанский Д.В. Контроль и преобразования дискретных автоматов. – К.: Наукова думка, 1975. – 176 с.
25. Грунский И.С., Козловский В.А. Синтез и идентификация автоматов. – К.: Наукова думка, 2004. – 246 с.
26. Hartmanis J., Stearns R.E. Algebraic structure theory of sequential machines. – Prentice – Hall, Inc.. – 1966. – С.340.
27. Баранов С.И. Синтез микропрограммных автоматов (граф-схемы и автоматы). – 2-е изд., перераб. и доп. – Л.: Энергия, Ленингр. отд-ние, 1979. – 232 с.

28. Агибалов Г.П., Евтушенко Н.В. Декомпозиция конечных автоматов. – Томск: Изд-во Томск. ун-та, 1985. – 127 с.

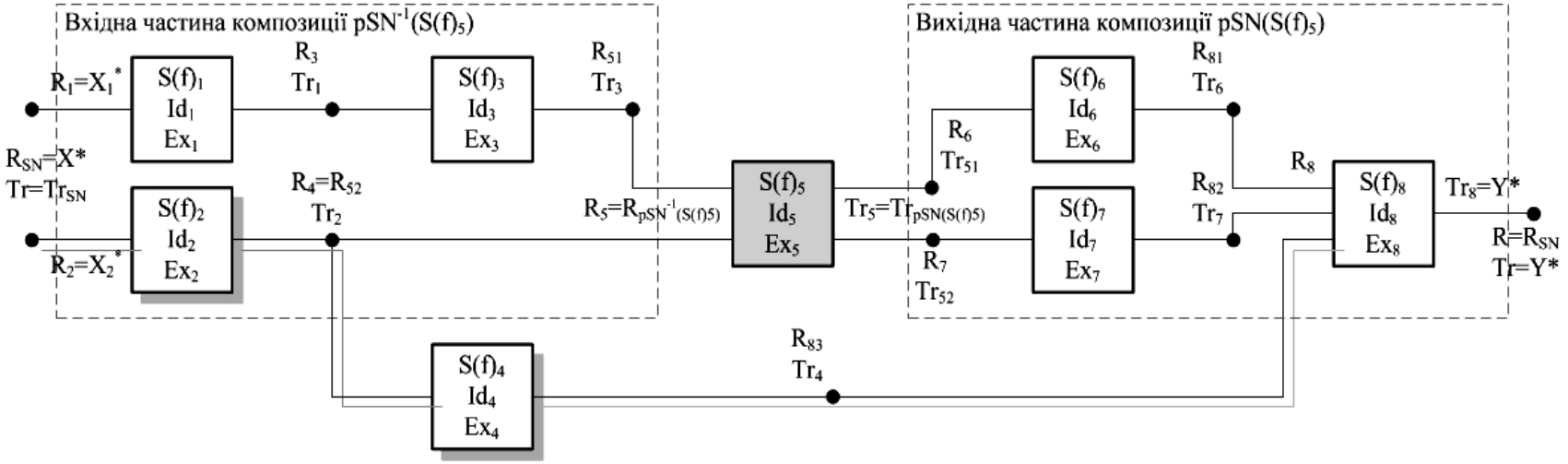
АВТОМАТНА МОДЕЛЬ NAT



$$a = (S, X, Y, T, Pb, \delta_T, \lambda_T, S_0)$$

де S,X,Y – алфавиты станів, вхідний і вихідний, $T \subset N$ – інтервали, $Pb \subset [0;1] \subset D$ – вероятностні коефіцієнті диапазону $[0;1]$, $\delta_T : S \times X \times T \times Pb \to S$ – временна недетермінована функція переходів, $\lambda_T : S \times X \times T \times Pb \times S \to Y$ – временна недетермінована функція виходів для переходів, $S_0 \subseteq S$ – начальні стани.

МОДЕЛЬ АВТОМАТНОЇ МЕРЕЖІ



$$na=(X, Y, A, \xi),$$

де X, Y - загальні вхідний і вихідний алфавіти мережі; $A=\cup_{i\in I} a_i$ – автомати мережі na з вхід-вихідними словами зарегістрованої поведінки $W''=\cup_{i\in I} W_i''$ у алфавітах $U_i''\subseteq U_i'\times T_i\times Pb_i$, структуровані мережею na; $\xi=\cup_{i\in I} \xi_i$ алфавітні, віроятнісно-временні відотображення для структур з’вязків у мережі na:

$$\xi_{i''}: x_{i'\in I\setminus\{i\}}(Y_{i\wedge}\times T_{i\wedge}\times Pb_{i\wedge})\to(X_{i\wedge\wedge}\times T_{i\wedge\wedge}\times Pb_{i\wedge\wedge}) \text{ чи } \xi_{i\wedge\wedge}: x_{i\wedge\in I\setminus\{i\}}(Y_{i\wedge}'')\to(X_{i\wedge\wedge}''),$$

МЕРЕЖЕВА МОДЕЛЬ ВЕРИФІКАЦІЇ

$$tn=(Ta,\ aR_T^{-1}{}_{(A)},\ aTr_{T(A)},\ Sg_{tn},\ Re_A,\ Tre_A),$$

де

– $Ta=\cup_{i\in I}\ ta_i$ - компонентні моделі верифікації для компонентних автоматів $A=\cup_{i\in I}\ a_i$ мережі na;

– $aR_T^{-1}{}_{(A)}=\cup_{i\in I}\ aR_T^{-1}{}_{(ai)}$, $aTr_{T(A)}=\cup_{i\in I}\ aTr_{T(ai)}$ - реалізуємі і транспортуємі півавтомати топологічних вузлів мережі na, входи і виходи компонентних автоматів з $A=\cup_{i\in I}\ a_i$, для $i\in I$ $pr_1W_i''\leq aR_T^{-1}{}_{(ai)}\subseteq X_i''^*$, $pr_2W_i''\leq aTr_{T(ai)}\subseteq Y_i''^*$, виходи реалізуючих підмереж $T^{-1}(A)=\cup_{i\in I}\ T^{-1}(a_i)$ и входами транспортирующих (к выходам сети na для $aTr_{T(A)}$) подсетей $T(A)=\cup_{i\in I}\ T(a_i)$). З їх визначаються вхідні реалізуємі $aR_{(A)}=\cup_{i\in I}\ aR_{ai}$ і вихідні транспортуємі $aTr_A=\cup_{i\in I}\ aTr_{ai}$ слів поведінки для $a_i\in A$ в алфавіті U_i'';

– $Sg_{tn}=\{\circ^{Y''},\circ^{X''},\times^{Y''},\times^{X''},*^{Y''},*^{X''}\}$ - мережеві операції п композиції поведінки: послідовної, паралельної, зі зворотним з'вязком;

– $Re_A=\cup_{i\in I}\ Re_{(ai)}$, $Tre_A=\cup_{i\in I}\ Tre_{(ai)}$ - реалізовані і транспортовані фрагменти поведінки – особини при використанні еволюційної оптимізації у стратегії композиції.

ПРИКЛАДИ МЕРЕЖЕВИХ ОПЕРАЦІЙ КОМПОЗИЦІЇ

Послідовна

$$a_{km}=(a_k \circ'' a_m)=\begin{cases}(a_k \circ a_m), \text{ якщо } (np_1Y_k''=Y_k)\cap(np_1X_m''=X_m)=Y_{km}\neq\varnothing \,\&\\ \quad \&\ \forall y_{km}''\in Y_{km}''(y_{km}''=(y_{km},t_{km},pb_{km})(t_{km}=t_k\cap t_m\neq\varnothing\,\&\\ \quad \&\ pb_{km}=pb_k\cdot pb_m\neq 0)),\\ \text{иначе не определена}\end{cases}$$

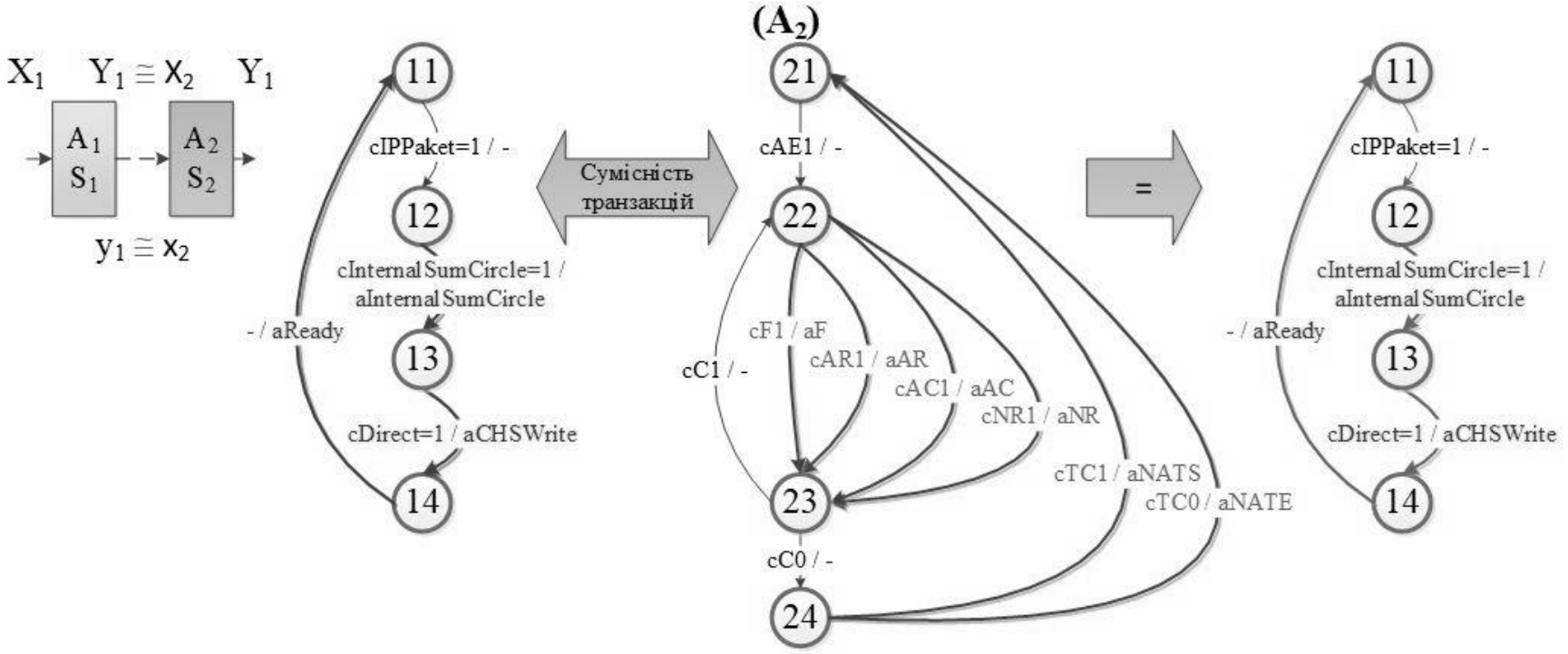
$$a_k{}^{Yk''}{}_m=(a_k \circ^{Y''} a_m)=(Y''\text{-}min(\downarrow_Y(a_k))\circ'' a_m)=(Y''\text{-}min(a_k{}^{Y''})\circ'' a_m)=(a_k{}^{Y''min}\circ'' a_m)$$

Паралельна

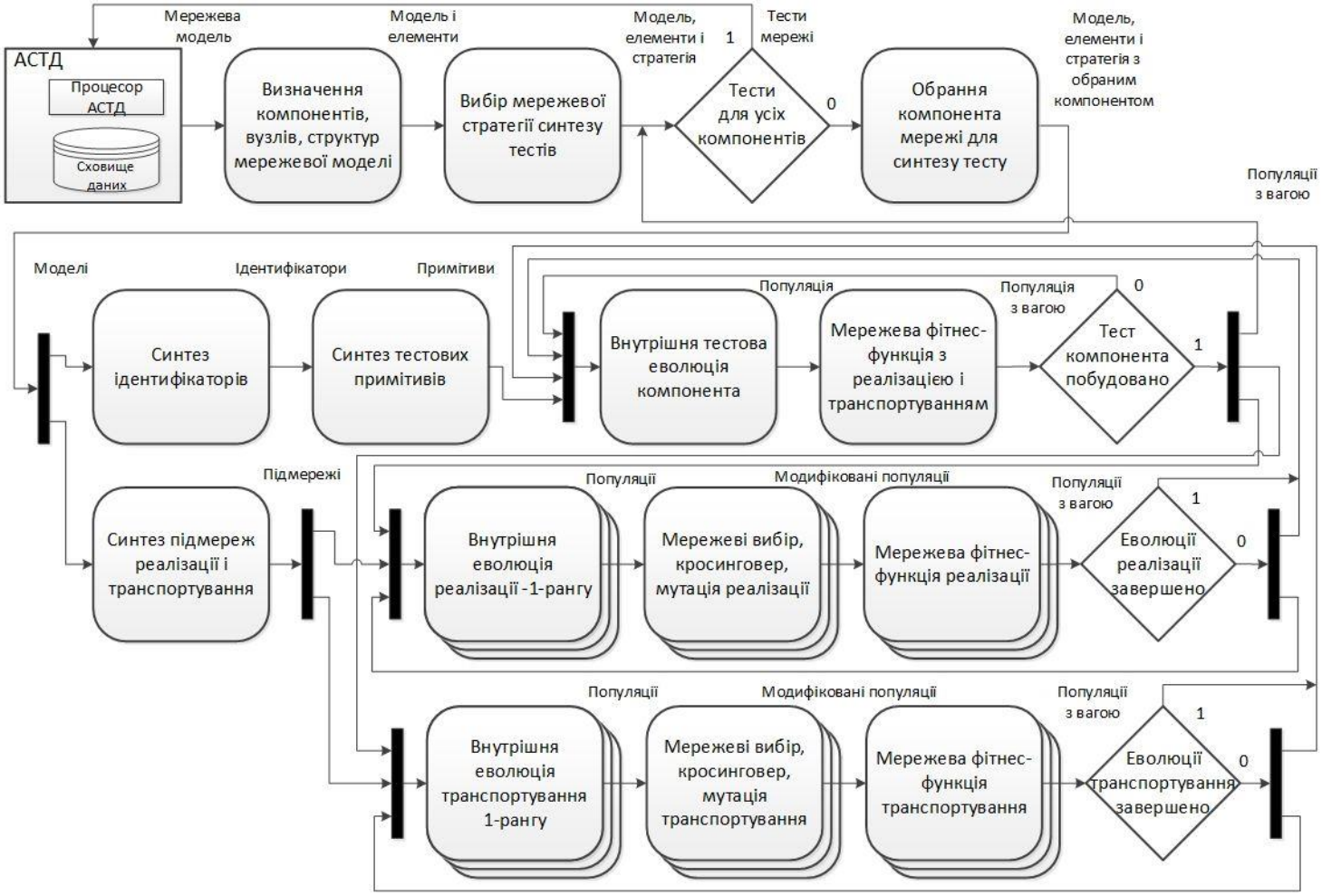
$$a_{km}=(a_k \times'' a_m)=\begin{cases}(a_k \times a_m), \text{ если } (np_1Y_{km}''=np_1Y_k''\times np_1Y_m'')\&(np_1X_{km}''=\\ \quad =np_1X_k''\times np_1X_m'')\ \&\ \forall x_{km}''\in X_{km}''(x_{km}''=(x_{km},t^x{}_{km},pb^x{}_{km})\ \&\\ \quad \&\ (x_{km}=(x_k,x_m)\ \&\ t^x{}_{km}=t^x{}_k\cap t^x{}_m\neq\varnothing\ \&\ pb^x{}_{km}=pb^x{}_k\cdot pb^x{}_m\neq 0\ \&\\ \quad \&\ \forall y_{km}''\in Y_{km}''(y_{km}''=(y_{km},t^y{}_{km},pb^y{}_{km})\ \&\ (y_{km}=(y_k,y_m)\ \&\\ \quad \&\ (y_{km}=(y_k,y_m)\ \&\ t^y{}_{km}=t^y{}_k\cap t^y{}_m\neq\varnothing\ \&\ pb^y{}_{km}=pb^y{}_k\cdot pb^y{}_m\neq 0)),\\ \text{иначе не определена}\end{cases}$$

$$a_{km}{}^{Ym''min}=(a_k \times^{Y''} a_m)=(Y''\text{-}min(\downarrow_Y(a_k\times'' a_m)))=(Y''\text{-}min(\downarrow_Y(a_{km})))=(Y''\text{-}min(a_{km}{}^{Y''}))$$

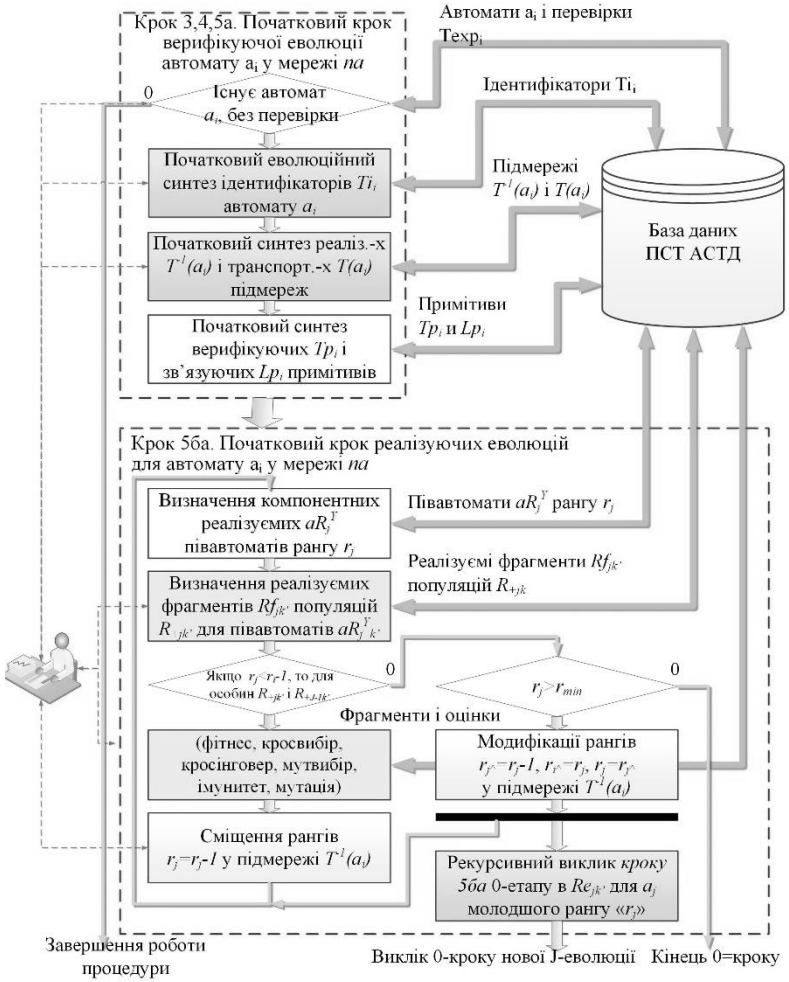
ЗОВНІШНЯ МЕРЕЖЕВА МУТАЦІЯ ДЛЯ ВЕРИФІКУЮЧОГО ТА ТРАНСПОРТУЮЧОГО ФРАГМЕНТІВ ПОВЕДІНКИ АВТОМАТІВ CHECKSUM(A_1) І NAT

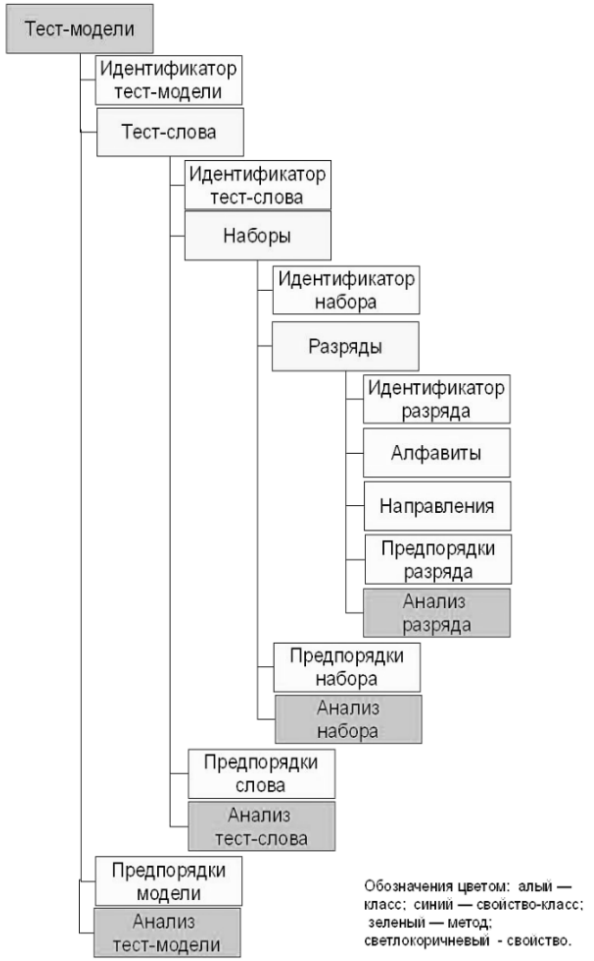


МЕРЕЖЕВА ПРОЦЕДУРА ВЕРИФІКАЦІЇ КОМУТАЦІЙ



ПОЧАТКОВИЙ ФРАГМЕНТ СТАДІЙ ВЕРИФІКУЮЧОЇ ТА РЕАЛІЗУЮЧОЇ ЕВОЛЮЦІЙ МЕРЕЖЕВОЇ ПРОЦЕДУРИ ТА ІЄРАРХІЯ МОДЕЛЕЙ ВЕРИФІКАЦІЇ ТА ТЕСТУВАННЯ





ОБ'ЄКТНО-РЕЛЯЦІЙНА МОДЕЛЬ БАЗИ ДАНИХ КОМПЛЕКСА ВЕРИФІКАЦІЇ

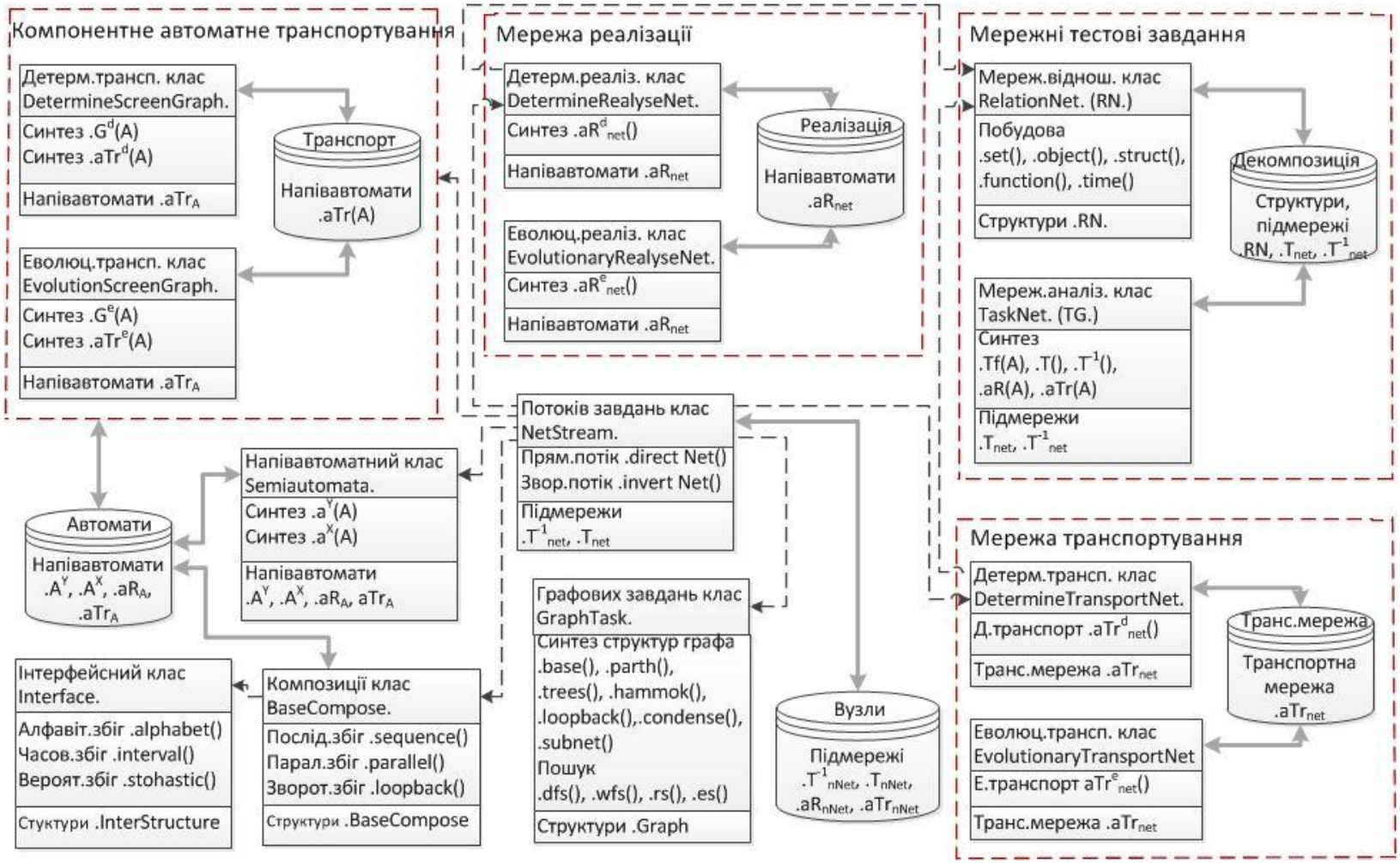
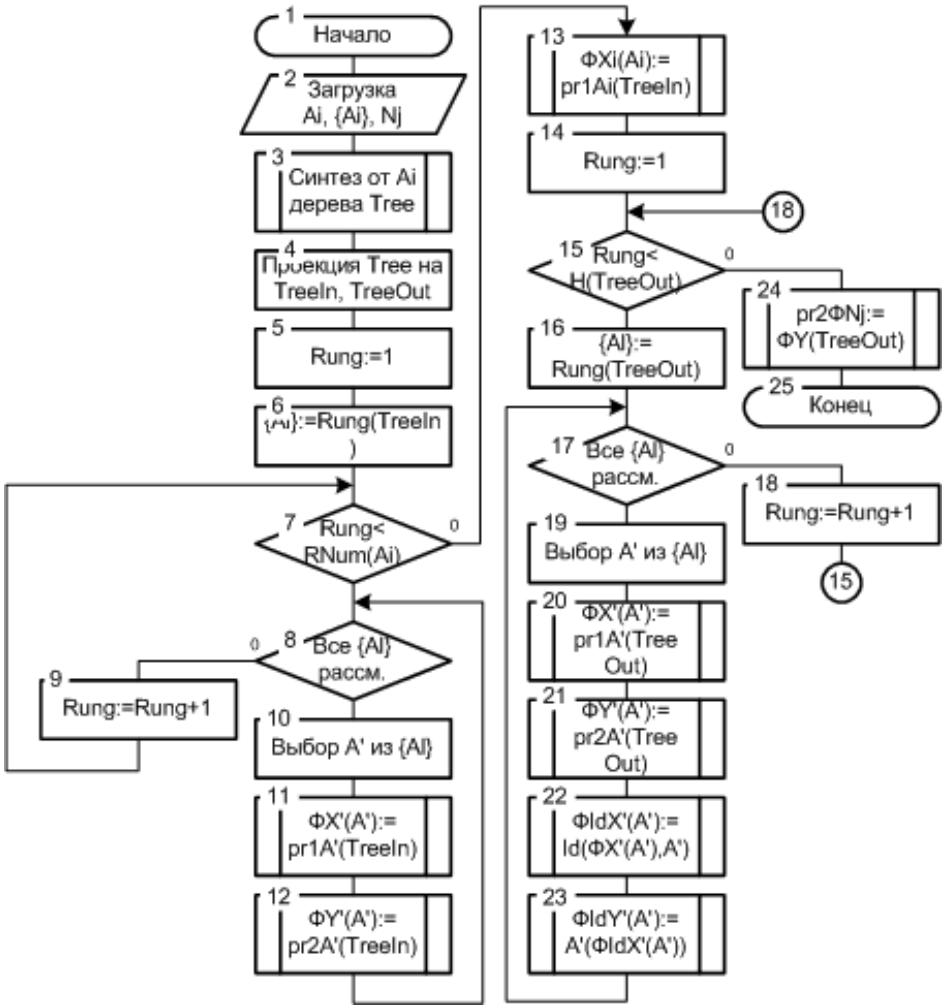
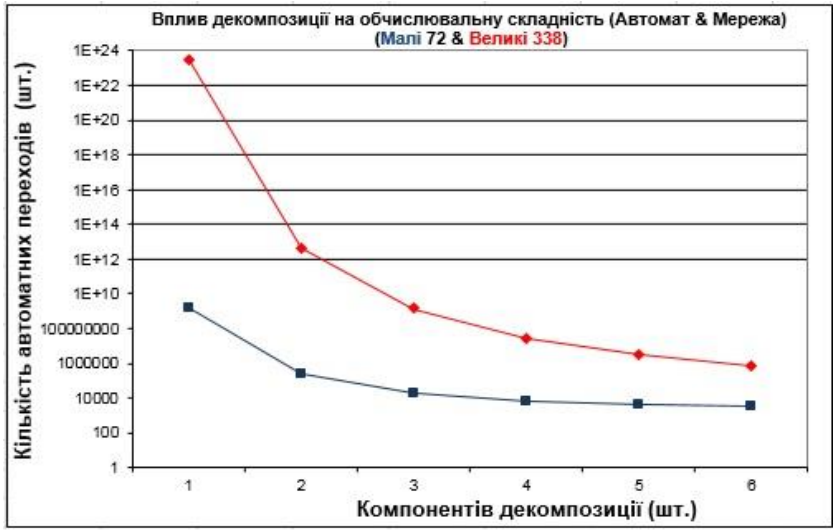


СХЕМА АЛГОРИТМУ РЕАЛІЗОВАНОЇ Й ТРАНСЛЬОВАНОЇ ПОВЕДІНКИ КОМУТАЦІЇ



ЗАЛЕЖНІСТЬ ОБЧИСЛЮВАЛЬНОЇ СКЛАНОСТІ ЕВОЛЮЦІЙННОЇ ВЕРИФІКАЦІЇ ВІД СТУПЕНЯ ДЕКОМПОЗИЦІЇ ТА ЕКСПЕРИМЕНТАЛЬНІ ЗНАЧЕННЯ СКЛАДНОСТІ (ЧАСУ) ВЕРИФІКАЦІЇ ДЛЯ АГЕНТНИХ КОМУТАЦІЙ



Механізм	Ступінь декомпозиції	Вхідна складність	Складність
Send	5	26	2495368
Receive	4	21	2013969
Cript	3	17	14701736
Roaming	2	8	11612
Client	5	27	20751408