

Посилання на джерела.....	3
1. Поняття моделі. Способи побудови моделей. Класифікація моделей.....	5
1.1. Загальна структура моделі.....	5
1.2. Класифікація моделей.....	6
1.3. Способи побудови моделей.....	6
2. Методи моделювання. Технології моделювання.....	7
3. Генератори випадкових чисел.....	8
3.1. Загальна схема.....	8
3.2. Генерування рівномірно розподілених в інтервалі $(0;1)$ випадкових величин на основі рекурсивних формул.....	9
3.3. Генерування випадкової величини методом оберненої функції.....	10
3.4. Табличний метод генерування випадкового числа r , що має закон розподілу $F(x)$..	11
4. Способи генерування рівномірно розподіленої в інтервалі $(0,1)$ випадкової величини.....	11
5. Формалізація процесів дискретно-подійних систем.....	11
6. Процес формалізації дискретно-подійної системи.....	12
6.1. Мережа масового обслуговування.....	12
6.2. Мережі Петрі.....	12
7. Формалізм мереж масового обслуговування.....	13
8. Алгоритм імітаційного моделювання дискретно-подійної системи.....	15
9. Алгоритм імітації на основі подійного представлення процесу функціонування та прирощення часу до найближчої події.....	16
10. Об'єктно-орієнтований підхід до розробки алгоритму імітаційного моделювання дискретно-подійної системи.....	16
10.1. Мережа масового обслуговування.....	16
10.2. Мережа Петрі.....	17
11. Універсальний алгоритм імітації мережі масового обслуговування.....	17
12. Верифікація алгоритму імітації.....	17
13. Оцінка точності імітації.....	18
14. Оцінка складності алгоритму імітації.....	18
15. Формалізм базових мереж Петрі.....	18
15.1. Формальне означення класичної мережі Петрі.....	19
15.2. Правило запуску переходу класичної мережі Петрі.....	19
15.3. Формальний опис запуску переходу.....	20
16. Формалізм стохастичних мереж Петрі.....	21
17. Універсальний алгоритм імітації стохастичної мережі Петрі.....	21

18. Алгоритм імітації стохастичної мережі Петрі з багатоканальними переходами....	26
19. Алгоритм імітації мережі Петрі з часовими затримками, з багатоканальними переходами, з конфліктними переходами.....	26
20. Математичний опис стохастичної мережі Петрі.....	31
21. Рівняння станів стохастичної мережі Петрі.....	31
21.1. Визначення моменту найближчої події.....	32
22. Матричні рівняння базової мережі Петрі.....	32
23. Матричні рівняння станів стохастичної мережі Петрі.....	33
24. Еквівалентні відношення між базовою, детермінованою та стохастичною мережами Петрі.....	35
25. Дослідження властивостей мереж Петрі. Інваріанти мережі Петрі.....	36
25.1. Збереження (консервативність).....	36
25.2. S – інваріант мережі Петрі.....	37
25.3. Циклічність.....	37
25.4. T – інваріант мережі Петрі.....	37
25.5. Досяжність.....	38
25.6. Активність.....	38
25.7. Дерево досяжності.....	38
26. Формалізм Петрі-об'єктної моделі.....	39
27. Математичний опис Петрі-об'єктної моделі.....	41
28. Алгоритм імітації Петрі-об'єктної моделі.....	46
29. Бібліотека класів Петрі-об'єктного моделювання.....	46
PetriNet.....	48
PetriP.....	48
PetriT.....	48
ArcOut та ArcIn.....	49
FunRand.....	49
NetLibrary.....	49
30. Програмне забезпечення Петрі-об'єктного моделювання.....	50
31. Моделювання багатопоточних програм стохастичною мережею Петрі.....	51
32. Експериментальні методи дослідження моделей систем.....	52
33. Еволюційні методи оптимізації імітаційних параметрів.....	53
34. Програмне забезпечення з імітаційного моделювання.....	54
35. Моделювання в Arena Professional Software.....	54
36. Побудова ієрархічних моделей в Arena.....	56
37. Елементи анімації програмного забезпечення Arena.....	57
38. Моделювання в CPNTools.....	59
39. Моделювання в GPSS.....	60
40. Загальні складові сучасного програмного забезпечення з імітаційного	

моделювання систем.....	60
40.1. GPSS (General Purpose Simulation System).....	60
40.2. PTRSIM (Petri Net Simulation System).....	61
40.3. Arena—сучасний пакет імітаційного моделювання.....	62
40.4. Порівняння GPSS, PTRSIM та Arena.....	63
41. Паралельні обчислення в імітаційному моделюванні.....	63
41.1. Прискорення експериментальних досліджень.....	63
41.2. Графічний інтерфейс.....	64
41.3. Прискорення виконання алгоритму імітації.....	64
42. Паралельний алгоритм імітації Петрі-об'єктної моделі.....	64
42.1. Варіант розпаралелювання Петрі-об'єктної моделі.....	64
42.2. Паралельний алгоритм імітації Петрі-об'єктної моделі.....	64
42.3. Взаємодія Петрі-об'єктів.....	65
42.4. Правила паралельного функціонування Петрі-об'єктів.....	65
42.5. Правила паралельного функціонування Петрі-об'єктів.....	66
42.6. Локальний час об'єкта просувається за такими правилами.....	66
42.7. Межа безпечного інтервалу.....	67
42.8. Об'єкт.....	67
42.9. Синхронізація виходу маркерів у спільну позицію.....	67
42.10. Умова завершення роботи потоків.....	67
42.11. Узагальнення.....	68
42.12. Взаємодія потоків паралельного алгоритму імітації Петрі-об'єктної моделі.....	68

Посилання на джерела

Моделювання Систем Посібник.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/blob/main/lections/МоделюванняСистемНавчПосібник_2011.pdf.

Лекція 5.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/blob/main/lections/Лекція 5. Універсальний алгоритм мережі масового обслуговування .pdf.

Лекція 10. Алгоритм імітації стохастичної мережі Петрі.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/blob/main/lections/Лекція 10. Алгоритм імітації стохастичної мережі Петрі.pdf.

Лекція 1. Поняття моделі. Методи моделювання 2022.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/blob/main/lections/Лекція 1. Поняття моделі. Методи моделювання 2022.pdf.

Лекція 11 Математична теорія стохастичної мережі Петрі.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/tree/main/lections/Лекція 11 Математична теорія стохастичної мережі Петрі.pdf.

Лекція 12 Формалізм Петрі-об'єктної моделі.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/tree/main/lections/Лекція 12 Формалізм Петрі-об'єктної моделі.pdf.

Лекція 14 Теоретичні основи Петрі-об'єктного моделювання.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/blob/main/lections/Лекція 14 Теоретичні основи Петрі-об'єктного моделювання.pdf.

Лекція 15 ПЗ Петрі-об'єктного моделювання.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/blob/main/lections/Лекція 15 ПЗ Петрі-об'єктного моделювання.pdf.

Лекція 16 Моделювання паралельних обчислень.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/blob/main/lections/Лекція 16 Моделювання паралельних обчислень.pdf.

Лекція 17 Паралельний алгоритм імітації.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/blob/main/lections/Лекція 17 Паралельний алгоритм імітації.pdf.

Лекція 19 GPSS.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/blob/main/lections/Лекція 19 GPSS.pdf.

Лекція 6. Формалізм мережі Петрі.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/blob/main/lections/Лекція 6. Формалізм мереж Петрі.pdf.

Лекція 9 Математична теорія базової мережі Петрі.

https://github.com/SideShowBoBGOT/system_modelling_seventh_semester/tree/main/lections/Лекція 9 Математична теорія базової мережі Петрі.pdf.

Моделирование телекоммуникационных систем в CPN Tools.

<http://daze.ho.ua/cpnmp-ru.pdf>.

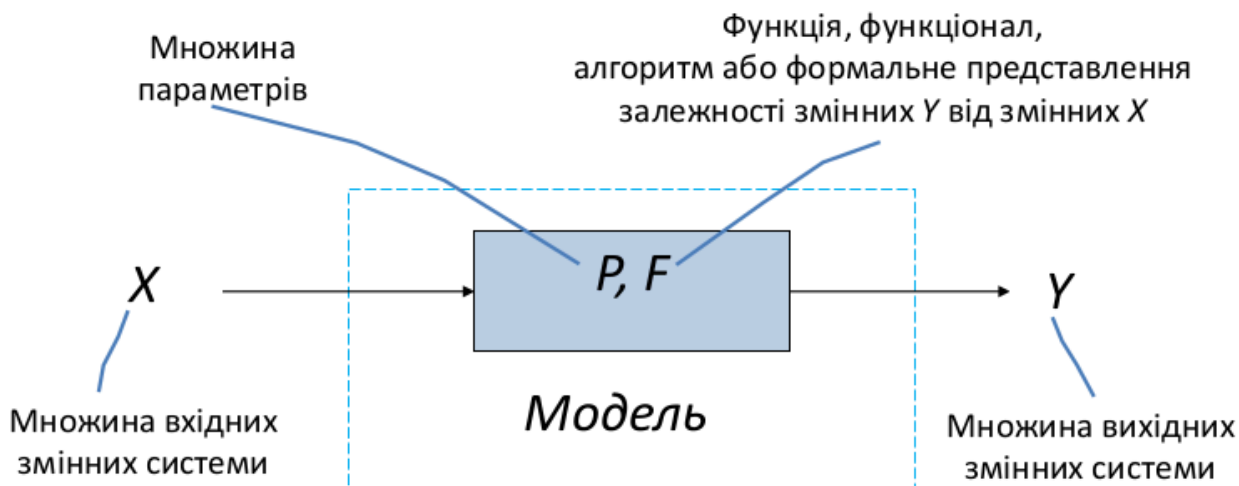
1. Поняття моделі. Способи побудови моделей. Класифікація моделей

(Лекція 1. Поняття моделі. Методи моделювання 2022 3)

Моделлю називається представлення об'єкта, системи чи поняття в деякій абстрактній формі, що є зручною для наукового дослідження.

1.1. Загальна структура моделі

(Лекція 1. Поняття моделі. Методи моделювання 2022 4)



- Модель завжди є спрощенням реального процесу/ системи.
- Від рівня деталізації опису процесів залежить складність моделі.
- Мистецтво дослідника, який будує модель, полягає в тому, щоб для найменш складної моделі отримати найбільш точні результати для задачі, яка поставлена

1.2. Класифікація моделей

(Лекція 1. Поняття моделі. Методи моделювання 2022 6)

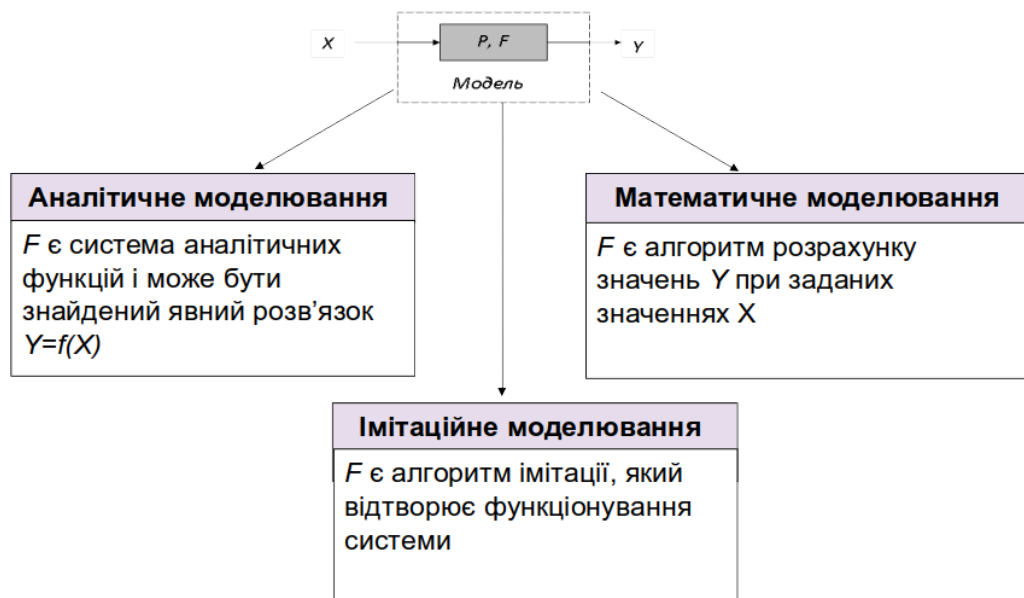
- 3 точки зору вихідної змінної моделі:
 - статичні та динамічні
 - неперервні та дискретні
 - детерміновані та стохастичні
- 3 точки зору способу побудови моделі:
 - фізичні та нефізичні
- 3 точки зору методу моделювання:
 - алгебраїчні, диференційні, аналітичні, імітаційні,...

1.3. Способи побудови моделей

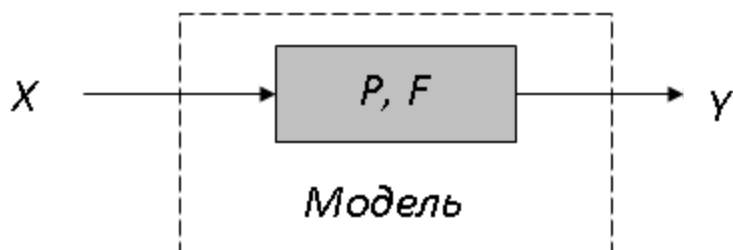
(Лекція 1. Поняття моделі. Методи моделювання 2022 5)

- Фізичний: в результаті ретельного вивчення системи встановлюються закони функціонування системи, які потім відтворюються за допомогою моделі => Фізичні моделі
- Нефізичний: без усякого фізичного обґрунтування припускається вид залежності F , невідомі параметри якої P потім відшуковуються за даними спостережень за змінними системи X, Y .

2. Методи моделювання. Технології моделювання.



Задача моделювання поділяються на:



Моделювання:	відомі $X, P, F \Rightarrow$ знайти Y
Управління:	відомі $Y, P, F \Rightarrow$ знайти X
Ідентифікації:	відомі X, Y , множина $F \Rightarrow$ знайти $f \in F, P$
Оптимізації:	відомі F , критерій $K \Rightarrow$ знайти P, X, Y
Прогнозування:	відомі $X_b, Y_b, T \Rightarrow$ знайти F, P, Y_{t+T}

Процес моделювання:



Для моделювання систем можна використати наступне програмне забезпечення:

Неперервні моделі: Matlab, Simulink

Дискретно-подійні моделі: VisSim, CPNTools, AnyLogic, ProModel, Simio, Arena Simulation Software

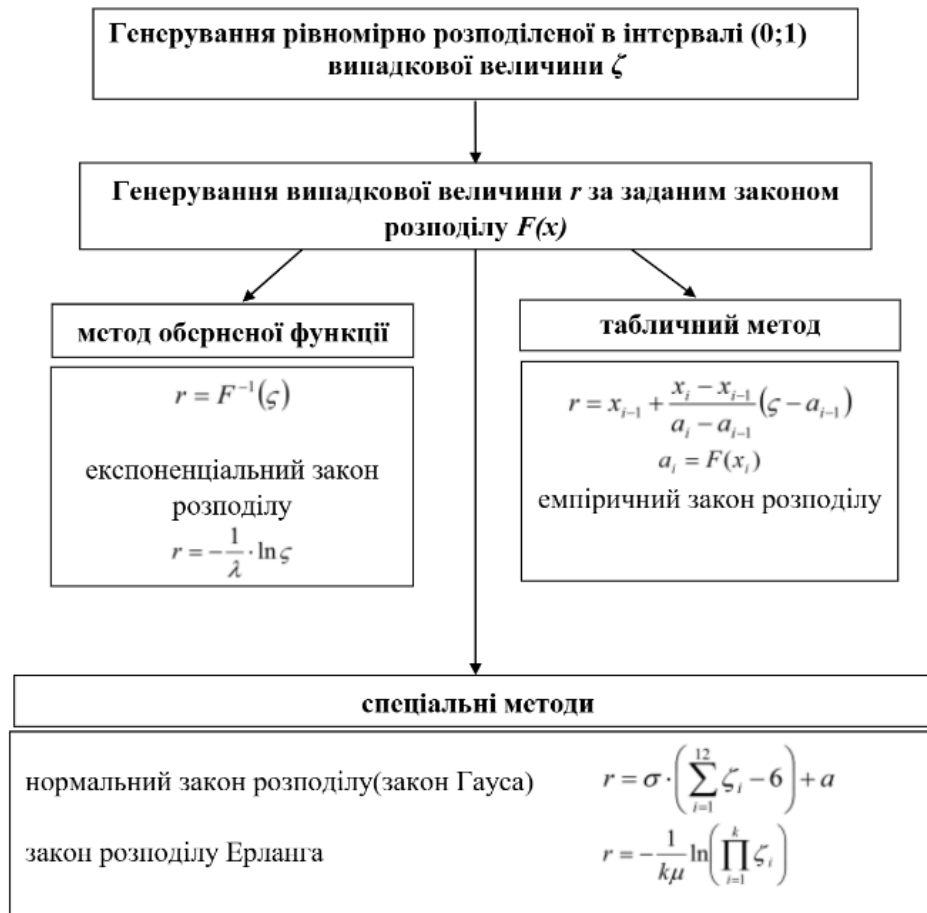
Концептуальну модель - Опис системи разом із указуванням цілі та задачі дослідження.

Розробка концептуальної моделі системи складає основну задачу системного аналізу об'єкта, явища чи поняття, що досліджується.

3. Генератори випадкових чисел

3.1. Загальна схема

(Лекція 1. Поняття моделі. Методи моделювання 2022 16)



3.2. Генерування рівномірно розподілених в інтервалі (0;1) випадкових величин на основі рекурсивних формул

(Лекція 1. Поняття моделі. Методи моделювання 2022 17)

$$z_{i+1} = (a \cdot z_i + b) \cdot (\text{mod}(c)), i = 0, 1, \dots$$

$$\xi_{i+1} = \frac{z_{i+1}}{c}$$

Тестування генераторів рівномірно розподілених в інтервалі (0,1) випадкових чисел:

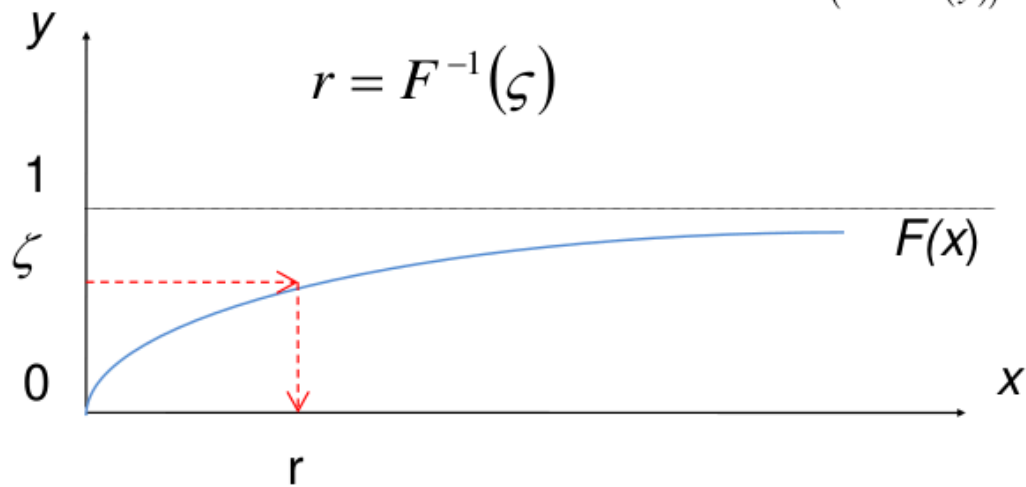
- перевірка на рівномірність,
- перевірка на випадковість,
- кореляції

3.3. Генерування випадкової величини методом оберненої функції

(Лекція 1. Поняття моделі. Методи моделювання 2022 18)

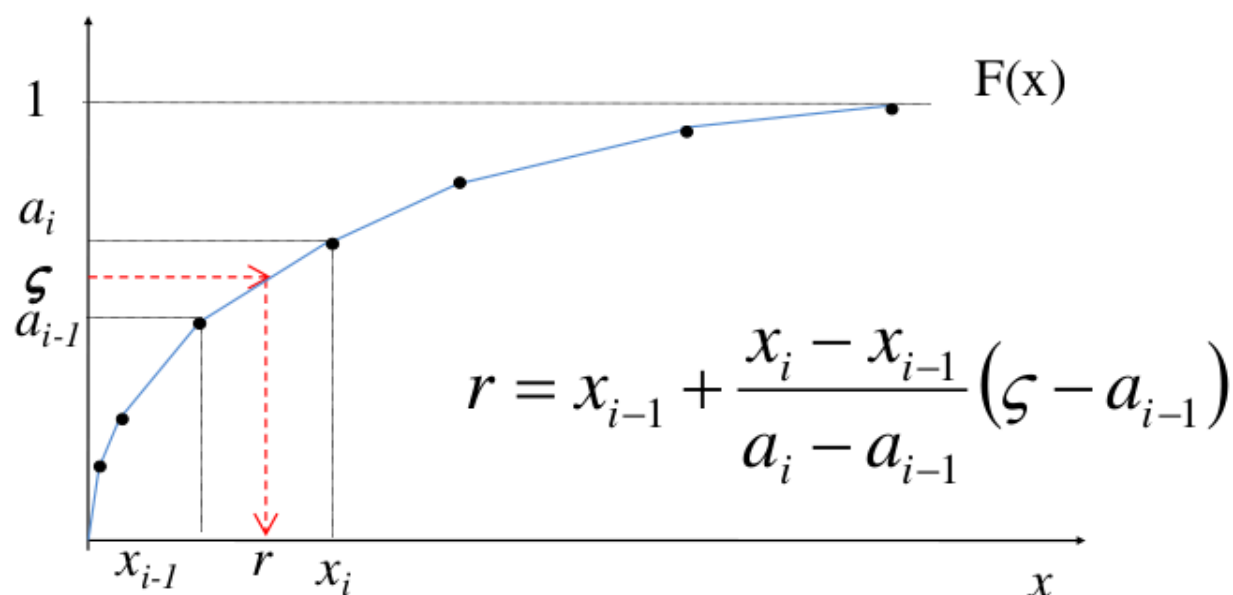
$$\zeta_i = F(r_1), \dots, \zeta_n = F(r_n)$$

$$\begin{aligned} G(y) &= P(\zeta \leq y) = P(F(r) \leq y) = P(F^{-1}(F(r)) \leq F^{-1}(y)) = \\ &= P(r \leq F^{-1}(y)) = F(F^{-1}(y)) = y \end{aligned}$$



Приклад: $\zeta = 1 - e^{-\lambda r} \Leftrightarrow r = -\frac{1}{\lambda} \cdot \ln(1 - \zeta).$

3.4. Табличний метод генерування випадкового числа r , що має закон розподілу $F(x)$



4. Способи генерування рівномірно розподіленої в інтервалі (0,1) випадкової величини.

5. Формалізація процесів дискретно-подійних систем

(Модельовання Систем Посібник {62, 66})

Формалізація процесів функціонування дискретних систем	Мережі масового обслуговування	Розімкнуті. Вимоги надходять ззовні мережі і після обробки залишають її.
		Замкнуті. Деяка кількість вимог весь час знаходиться в ній, переходячи з однієї СМО до іншої, але ніколи не залишаючи мережу
		З блокуванням маршруту. Управління блокуванням схематично позначається пунктирною стрілкою

		(управляючі зв'язки між елементами). Наприклад, канал обслуговування блокується, якщо наступний канал обслуговування зайнятий і зайняті всі місця в його черзі.
	Мережі Петрі	з часовими затримками
		з конфліктними переходами
		з багатоканальними переходами
		з інформаційними зв'язками

6. Процес формалізації дискретно-подійної системи

6.1. Мережа масового обслуговування

(Моделювання Систем Посібник 66)

Для того, щоб представити систему засобами мережі масового обслуговування потрібно:

- з'ясувати, що являється в системі об'єктом обслуговування;
- виділити елементи процесу обслуговування об'єктів і кожному елементу поставити у відповідність СМО;
- для кожної СМО визначити кількість пристроїв та наявність черги;
- з'єднати СМО у відповідності до процесу обслуговування;
- визначити маршрут проходження об'єкту обслуговування від однієї СМО до іншої;
- визначити умови надходження в кожную СМО (ймовірність вибору маршруту та інші);
- визначити наявність блокування маршруту та умови блокування;
- визначити числові значення параметрів кожної СМО;
- визначити числові значення параметрів зовнішнього потоку на обслуговування;
- визначити стан мережі масового обслуговування на початку моделювання.

6.2. Мережі Петрі

(Моделювання Систем Посібник 71)

Для того, щоб представити систему засобами мереж Петрі потрібно:

- виділити події, що виникають в системі, і поставити у відповідність кожній події перехід мережі Петрі;
- з'ясувати умови, при яких виникає кожна з подій, і поставити у відповідність кожній умові позицію мережі Петрі;

- визначити кількість фішок у позиції мережі Петрі, що символізує виконання умови;
- з'єднати позиції та переходи відповідно до логіки виникнення подій у системі : якщо умова передувє виконанню події, то з'єднати в мережі Петрі відповідну позицію з відповідним переходом; якщо умова являється наслідком виконання події, то з'єднати в мережі Петрі відповідний перехід з відповідною позицією;
- з'ясувати зміни, які відбуваються в системі при здійсненні кожної події, і поставити у відповідність змінам переміщення визначеної кількості фішок із позицій в переходи та з переходів у позиції;
- визначити числові значення часових затримок в переходах мережі Петрі;
- визначити стан мережі Петрі на початку моделювання.

7. Формалізм мереж масового обслуговування

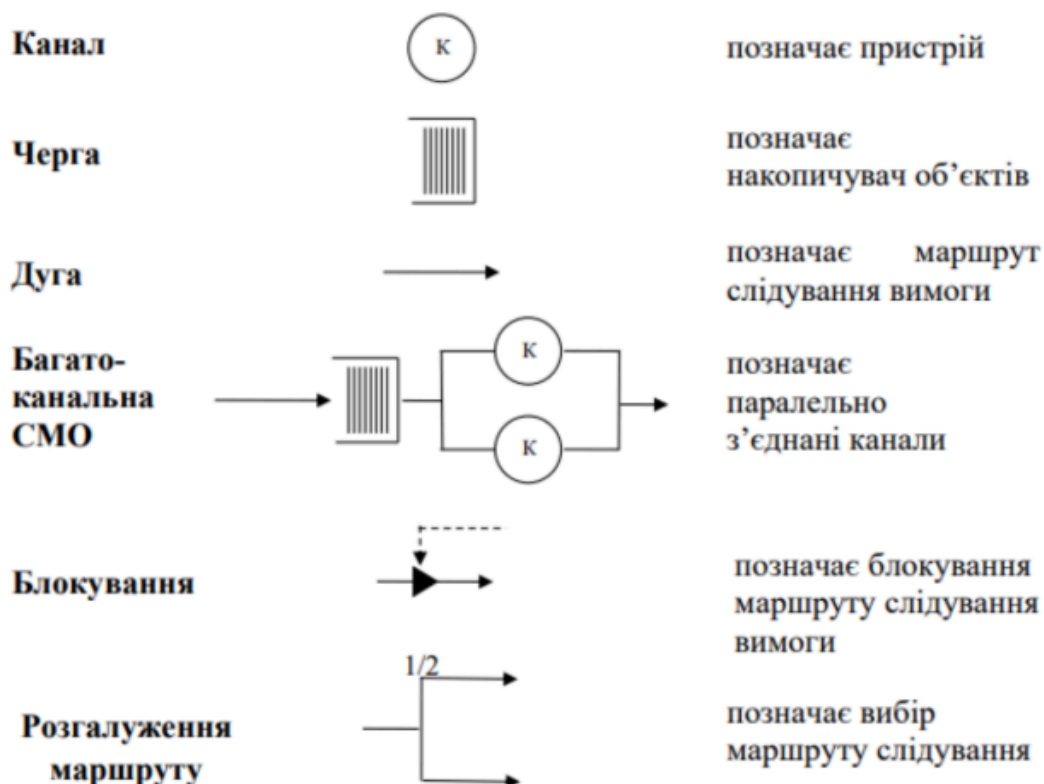
(Моделювання Систем Посібник 63)

Стохастична мережа масового обслуговування (МО) - це сукупність взаємозалежних систем масового обслуговування (СМО), у якій об'єкти, що обслуговуються, з визначеною ймовірністю переходять з однієї СМО в іншу.

Кожна система масового обслуговування - це одне або декілька обслуговуючих пристроїв з чергою. У **розімкнутій** мережі МО вимоги надходять ззовні мережі і після обробки залишають її. У **замкнутій** мережі МО деяка кількість вимог весь час знаходиться в ній, переходячи з однієї СМО до іншої, але ніколи не залишаючи мережу МО.

Мережі масового обслуговування функціонують за наступними простими правилами. Вимоги, що надходять на обслуговування, проходять одну за одною СМО згідно указанного маршруту. На вході кожної СМО вимога намагається знайти один вільний пристрій з декількох паралельно функціонуючих. Правило, за яким вимога обирає той чи інший пристрій за умови, що вони вільні, може бути указано окремо у відповідності до процесу функціонування реальної системи. Якщо всі пристрої однакові, то вимога переглядає пристрої по порядку і займає перший, що виявився вільним. Якщо в момент надходження вимоги до СМО всі пристрої виявились зайнятими, то вимога при наявності черги у СМО намагається в неї потрапити, або залишає мережу масового обслуговування і вважається не обслугованою.

Черга у СМО обов'язково одна. Кількість місць у черзі може бути обмежена або необмежена.



Мережа масового обслуговування може мати такі ускладнюючі елементи, як декілька типів вимог, що обслуговуються, розгалуження та блокування маршрутів, черги із складними правилами упорядкування вимог у них і т.і.

Розгалуження маршруту означає, що маршрут вимоги може бути визначений тільки з визначеною ймовірністю. Сума ймовірностей у місці розгалуження маршруту складає одиницю.

Блокування маршрутів допомагають ввести в мережу МО елементи, що управляють процесом обслуговування. Наприклад, вилучити поламаний пристрій з процесу обслуговування, заборонити подальше просування вимоги до виконання деякої умови тощо.

(Моделювання Систем Посібник 66)

Для того, щоб представити систему засобами мережі масового обслуговування потрібно:

- з'ясувати, що являється в системі об'єктом обслуговування;
- виділити елементи процесу обслуговування об'єктів і кожному елементу поставити у відповідність СМО;
- для кожної СМО визначити кількість пристроїв та наявність черги;
- з'єднати СМО у відповідності до процесу обслуговування;
- визначити маршрут проходження об'єкту обслуговування від однієї СМО до іншої;
- визначити умови надходження в кожну СМО (ймовірність вибору маршруту та інші);
- визначити наявність блокування маршруту та умови блокування;
- визначити числові значення параметрів кожної СМО;

- визначити числові значення параметрів зовнішнього потоку на обслуговування;
- визначити стан мережі масового обслуговування на початку моделювання.

8. Алгоритм імітаційного моделювання дискретно-подійної системи.

(Моделювання Систем Посібник 163)

Алгоритм, який відтворює функціонування системи, за допомогою комп'ютерної програми називається алгоритмом імітації. Побудова алгоритму імітації складається з побудови:

- *(Моделювання Систем Посібник 163)* алгоритму просування модельного часу;
 - за принципом Δt ;
 - за принципом найближчої події;
 - за принципом послідовного проведення об'єктів вздовж моделі.
- *(Моделювання Систем Посібник 165)* алгоритму просування стану моделі в залежності від часу;
 - орієнтований на події;
 - орієнтований на дії;
 - процесно-орієнтований.
- *(Моделювання Систем Посібник 173)* алгоритму збирання інформації про поведінку моделі у процесі імітації;

Алгоритм імітації дискретно-подійної системи відтворює упорядковану в часі послідовність подій. Функціонування системи повністю визначається послідовністю подій, які відбуваються в часі. Алгоритм імітації вирішує наступні питання: просування моделі в часі, просування стану моделі в часі та розробка алгоритму для збору інформації про функціонування системи.

Узагальнена схема алгоритму імітації

BEGIN

введення_початкових_даних()

WHILE поточний_час < час_моделювання DO

просунути_поточний_час()

виконати_змінювання_стану_моделі()

END WHILE

виведення_результатів_моделювання()

END

9. Алгоритм імітації на основі подійного представлення процесу функціонування та прирощення часу до найближчої події.

(Моделювання Систем Посібник 164)

За принципом найближчої події модельний час просувається від моменту виникнення однієї події до моменту виникнення іншої, і після кожного просування часу реалізуються зміни стану моделі, відповідні до події, що виникла.

10. Об'єктно-орієнтований підхід до розробки алгоритму імітаційного моделювання дискретно-подійної системи.

10.1. Мережа масового обслуговування

(Моделювання Систем Посібник 178)

Об'єктно-орієнтований підхід до побудови імітаційних моделей мереж масового обслуговування дозволяє складати програми, що можуть легко стати універсальними програми імітації мереж масового обслуговування. Час, що витрачається на побудову одного об'єкта надолужується під час створення однотипних екземплярів об'єкта.

Розглянемо побудову алгоритму імітації мережі масового обслуговування на основі об'єктно-орієнтованого підходу. Виберемо об'єкти, з яких складається мережа масового обслуговування:

- об'єкт «вхідний потік»,
- об'єкт «СМО»,
- об'єкт «маршрут»
- об'єкт «маршрут входу»
- об'єкт «маршрут виходу»

Об'єкт «вхідний потік» призначений для створення вхідного потоку вимог із заданим середнім значенням інтервалу надходження вимог у мережу МО. Об'єкт «СМО» призначений для створення системи масового обслуговування, що характеризується заданою кількістю пристроїв, обмеженням на довжину черги, тривалістю обслуговування вимоги в пристрої. Дані про значення цих змінних присвоюються екземпляру об'єкта в момент його створення за допомогою методу Create.

Об'єкт «маршрут» призначений для створення зв'язків між СМО. Об'єкти «маршрут входу» та «маршрут виходу» призначені для створення зв'язків між зовнішнім середовищем та мережею масового обслуговування.

Кожний об'єкт має поле «вхід» та «вихід». Передача вимоги уздовж маршруту від однієї СМО до іншої означає, що вимога, яка знаходиться на виході об'єкта, що означений на початку маршруту, передається на вхід об'єкта, що означений на кінці маршруту. Наприклад, передача зі СМО1 до СМО2 означає зникнення вимоги на виході СМО1 і поява вимоги на вході СМО2.

Об'єкти мають методи, що виконують властиві об'єктам дії. Так, об'єкт «СМО» містить методи «зайняти СМО» та «звільнити СМО», а також методи «повідомити про середню кількість зайнятих пристроїв у СМО», «повідомити про стан черги СМО» та інші. Об'єкт «маршрут» має метод «передати вимогу», а також метод «повідомити про кількість не обслуговуваних вимог».

Після того як об'єкт «СМО» полагоджений, до головної програми, яка здійснює імітаційне моделювання, підключається модуль, що містить опис об'єкта «СМО», і різні СМО створюються як екземпляри об'єкту «СМО».

10.2. Мережа Петрі

(Моделювання Систем Посібник 274)

11. Універсальний алгоритм імітації мережі масового обслуговування

(Лекція 5 2)

- Використовуємо способи:
 - просування часу за принципом до найближчої події,
 - орієнтований на події спосіб просування моделі в часі.
- Об'єктно-орієнтований підхід
- Розгалуження маршруту: за заданою ймовірністю або за пріоритетом
- Багатоканальність обслуговування
- Можливість вибору правила вибору замовлення з черги: FIFO або LIFO
- Блокування маршрутів за умовою, що враховує стан мережі або окремих її елементів
- Емпіричний закон розподілу

12. Верифікація алгоритму імітації

(Моделювання Систем Посібник 193)

Верифікація здійснюється спостереженням змінювання вихідних змінних моделі при змінюванні вхідних змінних моделі і порівнянням їх із очікуваним змінюванням. Наприклад,

при зменшенні інтервалу часу, з яким вимоги надходять у мережу масового обслуговування, очікується зменшення кількості вимог у її чергах та зменшення завантаження пристроїв. При збільшенні тривалості обслуговування вимог у пристроях деякої системи масового обслуговування очікується збільшення середньої кількості вимог у черзі цієї СМО та збільшення середнього завантаження пристроїв цієї СМО.

(Модельовання Систем Посібник 15)

Наприклад, змінюють значення вхідних змінних і спостерігають як модель реагує на таке змінювання. Якщо реакція моделі відповідає логіці її функціонування, то модель вважається правильною. Завершується створення моделі перевіркою адекватності моделі, що полягає у порівнянні значень вихідних змінних об'єкта, що моделюється, і моделі при однакових значеннях вхідних змінних. Очевидно, що таку перевірку можна здійснити тільки, якщо відомі деякі значення вхідних і вихідних змінних досліджуваного об'єкта.

13. Оцінка точності імітації

(Лекція 5 3)

- Виконується порівнянням з результатами аналітичного розрахунку.
- Аналітичний розрахунок можливий за таких умов:
 - усі черги необмеженої довжини
 - усі часові затримки задані випадковими величинами з експоненціальним законом розподілу
 - вибір маршруту виключно за заданими ймовірностями
 - блокування маршрутів відсутні

14. Оцінка складності алгоритму імітації.

(Лекція 5 7)

- Експериментальна оцінка складності виконується побудовою залежності часу виконання алгоритму в залежності від складності моделі.
- Теоретична оцінка складності виконується підрахунком кількості елементарних операцій в алгоритмі в залежності від складності моделі.

$$O(v \cdot timeMod \cdot k),$$

де O – інтенсивність подій;

$v \cdot timeMod$ – час моделювання;

k – середня, або максимальна, кількість елементарних операцій для обробки однієї події.

15. Формалізм базових мереж Петрі

(Модельовання Систем Посібник 67)

Мережі Петрі є засобом формального опису процесів функціонування дискретних систем. У дискретній системі зміни її стану трапляються в особливі моменти часу, коли виникають умови для здійснення події. Здійснення події означає зміну стану системи, а

значить, виникнення або не виникнення умов для інших подій. Процес функціонування дискретної системи – це упорядкована в часі послідовність подій.

(Лекція 6. Формалізм мережі Петрі 3)

Переваги формалізації мережею Петрі:

- Гнучкість формалізації подій
- Універсальність алгоритму імітації
- Візуалізація процесу функціонування
- Пристосованість до представлення паралельних процесів

Недоліки формалізації мережі Петрі:

- Велика кількість елементів

15.1. Формальне означення класичної мережі Петрі

(Лекція 6. Формалізм мережі Петрі 7)

$N = (P, T, A, W)$

$P = \{P\}$ - множина позицій;

$T = \{T\}$ - множина переходів;

$P \cap T = \emptyset$

$A \subseteq (P \times T \cup T \times P)$ - множина дуг;

$W: A \rightarrow \mathbb{N}$ - множина натуральних чисел, що задають кратності дуг (кількість зв'язків);



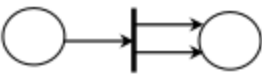


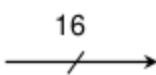
15.2. Правило запуску переходу класичної мережі Петрі

(Лекція 6. Формалізм мережі Петрі 8)

- Якщо в усіх вхідних позиціях переходу є маркери у кількості, рівній кратності дуги, то умова запуску переходу виконана
- Якщо умова запуску переходу виконана, то з усіх вхідних позицій переходу маркери видаляються у кількості, рівній кратності дуги, а в усі вихідні позиції переходу маркери додаються у кількості, рівній кратності дуги. Таким чином, в класичній мережі Петрі запуск переходу здійснюється миттєво.

Елементи мережі Петрі

ЕЛЕМЕНТИ МЕРЕЖІ ПЕТРІ

Перехід		позначає подію
Позиція		позначає умову
Дуга		позначає зв'язки між подіями та умовами
Маркер(один) (один)		позначає виконання (або не виконання) умови
Багато фішок		позначає багатократне виконання умови
Багато дуг		позначає велику кількість зв'язків

+ інформаційні дуги – це особливий зв'язок, при запуску такого переходу маркери з позиції з якої виходить дуга – не списуються.

15.3. Формальний опис запуску переходу

(Лекція 6. Формалізм мережі Петрі 9)

Множина вхідних позицій переходу:

$$\bullet T = \{P \in P \mid (P, T) \in A\}$$

Множина вихідних позицій переходу:

$$T^\circ = \{P \in P \mid (T, P) \in A\}$$

Умова запуску переходу:

$$\forall P \in \bullet T \quad M_P \geq W_{P,T} \rightarrow IT = 1$$

$$\exists P \in \bullet T \quad M_P < W_{P,T} \rightarrow IT = 0$$

Запуск переходу

$$IT = 1 \rightarrow \{\{\forall P \in \bullet T \quad M'_P = M_P - W_{P,T}\}, \forall P \in T^\circ \quad M''_P = M'_P + W_{P,T}\}$$

16. Формалізм стохастичних мереж Петрі

(Лекція 6. Формалізм мережі Петрі 35, Лекція 11 Математична теорія стохастичної мережі Петрі 2)

$N = (P, T, A, W, K, R)$

$P = \{P\}$ – множина позицій;

$T = \{T\}$ – множина переходів;

$P \cap T = \emptyset$

$A \subseteq P \times T \cup T \times P$ – множина дуг;

$W: A \rightarrow \mathbb{N}$ – множина натуральних чисел, що задають кратності дуг (кількість зв'язків);

$K = \{(c_T, b_T) | T \in T, c_T \in \mathbb{N}, b_T \in [0; 1]\}$ – множина пар значень, що задають пріоритет та ймовірність запуску переходів;

$R: T \rightarrow R_+$ – множина невід'ємних чисел, що характеризують часові затримки;

- $T, T \bullet$ – множина вхідних та множина вихідних позицій перехода T ;
- $P, P \bullet$ – множина вхідних і множина вихідних переходів позиції P ;

Стохастична мережа Петрі:

- містить часові затримки переходів, що можуть бути визначені випадковою величиною (з заданим законом розподілом)
- містить розв'язання конфлікту переходів з зазадною ймовірністю

Стохастична мережа Петрі є узагальненням мереж Петрі:

- При нульових часових затримках та рівноймовірнісному способі розв'язання конфліктів вона еквівалентна класичній мережі Петрі;
- При детермінованих затримках та рівноймовірнісному способі розв'язання конфліктів вона еквівалентна детермінованій мережі Петрі з часовими затримками.

17. Універсальний алгоритм імітації стохастичної мережі Петрі

На мій погляд, завдання 18 та 19 однакові абсолютно: у стохастичній мережах Петрі затримка в переходах, пріоритет та ймовірність обов'язкові відповідно до (Лекція 6. Формалізм мережі Петрі 35), тому у нас є у будь-якому разі усі інструменти, щоб вирішувати конфлікт переходів.

Для завдання 18 і 19 відповідь однакова. Поглянемо чим відрізняється 17 і 19. У 17 перехід може трекати лише один час виходу з себе, але не цілий список. Тому у 19 завданні відрізняються методи ВИКОНАТИ_ВХІД, ВИКОНАТИ_ВИХІД. Також у 17 перехід має поле активний, бо в нас лише один час виходу трекається, натомість у 19 ми маємо поле буфер - це кількість часів виходу, що зберігаються у списку.

Цей код написав відповідно до класу PetriSim Стеценко. Спочатку переписав його з допомогою гпт 4о з ООП у процедурний стиль, а потім переклав українською.

(Лекція 10. Алгоритм імітації стохастичної мережі Петрі 10, код з PetriObjModel)

АЛГОРИТМ: МОДЕЛЮВАТИ_ПЕТРІ_МЕРЕЖУ(мережа, часМоделювання, відстежуватиСтатистику)

ВХІД:

мережа: об'єкт PetriNet, що містить Місця та Переходи

часМоделювання: загальний час моделювання

відстежуватиСтатистику: функція для збору статистики

ЗМІННІ:

списокМісць: масив Місць (PetriPlaces) з мережі

списокПереходів: масив Переходів (PetriTransitions) з мережі

поточнийЧас: поточний час моделювання

мінімальнийЧас: час найближчої події

найближчаПодія: перехід із найближчим часом події

ОСНОВНИЙ АЛГОРИТМ:

поточнийЧас = 0

мінімальнийЧас = MAX_VALUE

найближчаПодія = null

мінімальнийЧас = ОБРОБИТИ_ВХІД(списокМісць, списокПереходів, поточнийЧас, найближчаПодія)

найближчаПодія = ЗНАЙТИ_МІН_ПОДІЮ(списокПереходів)

поки (поточнийЧас < часМоделювання) виконуємо

відстежуватиСтатистику(поточнийЧас)

поточнийЧас = мінімальнийЧас

// ФАЗА ВИХОДУ

якщо (поточнийЧас ≤ часМоделювання) тоді

найближчаПодія.ВИКОНАТИ_ВИХІД(списокМісць, поточнийЧас)

ОБРОБИТИ_БУФЕРИЗОВАНІ_ПОДІЇ(найближчаПодія, списокМісць,

поточнийЧас)

для кожного переходу із списокПереходів

якщо (перехід.активний

І перехід.часВиходу == поточнийЧас) тоді

перехід.ВИКОНАТИ_ВИХІД(списокМісць, поточнийЧас)

```

                                ОБРОБИТИ_БУФЕРИЗОВАНІ_ПОДІЇ(перехід,
списокМісць, поточнийЧас)
                                кінець якщо
                                кінець для
                                кінець якщо

                                // ФАЗА ВХОДУ
                                мінімальнийЧас = ОБРОБИТИ_ВХІД(списокМісць, списокПереходів,
поточнийЧас, найближчаПодія)
                                найближчаПодія = ЗНАЙТИ_МІН_ПОДІЮ(списокПереходів)
                                кінець поки

```

```

ПІДПРОГРАМА: ОБРОБИТИ_ВХІД(списокМісць, списокПереходів, поточнийЧас,
найближчаПодія)
    активніПереходи = ЗНАЙТИ_АКТИВНІ_ПЕРЕХОДИ(списокМісць, списокПереходів)
    мінімальнийЧас = MAX_VALUE

    якщо (активніПереходи порожній І ВСІ_БУФЕРИ_ПОРОЖНІ(списокПереходів)) тоді
        повернути мінімальнийЧас
    кінець якщо

    поки (активніПереходи не порожній) виконуємо
        обранийПерехід = ВИРІШИТИ_КОНФЛІКТ(активніПереходи)
        обранийПерехід.ВИКОНАТИ_ВХІД(списокМісць, поточнийЧас)
        активніПереходи = ЗНАЙТИ_АКТИВНІ_ПЕРЕХОДИ(списокМісць,
списокПереходів)
    кінець поки

    для кожного переходу із списокПереходів
        часМожливоїПодії = перехід.часВиходу
        якщо (часМожливоїПодії < мінімальнийЧас) тоді
            мінімальнийЧас = часМожливоїПодії
        кінець якщо
    кінець для

    повернути мінімальнийЧас

```

```

ПІДПРОГРАМА: ЗНАЙТИ_АКТИВНІ_ПЕРЕХОДИ(списокМісць, списокПереходів)
    активніПереходи = порожній список

    для кожного переходу із списокПереходів

```

якщо (перехід.УМОВА_ВИКОНУЄТЬСЯ(списокМісць) І перехід.імовірність \neq 0) тоді

 додати перехід до активніПереходи
 кінець якщо
кінець для

якщо (активніПереходи.розмір > 1) тоді
 відсортувати активніПереходи за пріоритетом за спаданням
кінець якщо

повернути активніПереходи

ПІДПРОГРАМА: ВИРІШИТИ_КОНФЛІКТ(переходи)

якщо (переходи.розмір \leq 1) тоді
 повернути переходи[0]
кінець якщо

обранийПерехід = переходи[0]
к-стьОднаковогоПріоритету = ПОРАХУВАТИ(переходи, де перехід.пріоритет ==
обранийПерехід.пріоритет)

якщо (к-стьОднаковогоПріоритету == 1) тоді
 повернути обранийПерехід
кінець якщо

випадковеЧисло = ЗГЕНЕРУВАТИ_ВИПАДКОВЕ(0, 1)
сумаімовірностей = 0

для кожного переходу із переходи, де перехід.пріоритет ==
обранийПерехід.пріоритет

якщо (перехід.імовірність == 1.0) тоді
 імовірність = 1.0 / к-стьОднаковогоПріоритету
інакше
 імовірність = перехід.імовірність
кінець якщо

сумаімовірностей = сумаімовірностей + імовірність

якщо (випадковеЧисло < сумаімовірностей) тоді
 повернути перехід
кінець якщо

кінець для

повернути обранийПерехід

ПІДПРОГРАМА: ОБРОБИТИ_БУФЕРИЗОВАНІ_ПОДІЇ(перехід, списокМісць, поточнийЧас)

поки (перехід.активний > 0) виконуємо

перехід.ВИДАЛИТИ_МІН_ПОДІЮ()

якщо (перехід.часВиходу == поточнийЧас) тоді

перехід.ВИКОНАТИ_ВИХІД(списокМісць, поточнийЧас)

інакше

перервати

кінець якщо

кінець поки

ПІДПРОГРАМА: ЗНАЙТИ_МІН_ПОДІЮ(списокПереходів)

мінПерехід = null

час = MAX_VALUE

для кожного переходу із списокПереходів

часПодії = перехід.часВиходу

якщо (часПодії < час) тоді

мінПерехід = перехід

час = часПодії

кінець якщо

кінець для

повернути мінПерехід

ПІДПРОГРАМА: ВИКОНАТИ_ВХІД(перехід, списокПозицій, поточнийЧас)

якщо (УМОВА_ВИКОНУЄТЬСЯ(перехід, списокПозицій)) тоді

для кожної позиції, яка визначена як вхідна для цього переходу

зменшити кількість маркерів у позиції на значення, що відповідає

логіці входу

кінець для

перехід.часВиходу = поточнийЧас + ЗГЕНЕРУВАТИ_ЧАС_ОБРОБКИ()

перехід.активний = true

кінець якщо

ПІДПРОГРАМА: ВИКОНАТИ_ВИХІД(перехід, списокПозицій, поточнийЧас)

якщо (перехід.активний) тоді

для кожної позиції
 збільшуємо кількість маркерів позиції на таке значення, що відповідає
 умові виходу
 кінець для
 перехід.часВиходу = MAX_VALUE
 перехід.активний = false
 кінець якщо

18. Алгоритм імітації стохастичної мережі Петрі з багатоканальними переходами.

19. Алгоритм імітації мережі Петрі з часовими затримками, з багатоканальними переходами, з конфліктними переходами

АЛГОРИТМ: МОДЕЛЮВАТИ_ПЕТРІ_МЕРЕЖУ(мережа, часМоделювання,
 відстежуватиСтатистику)

ВХІД:

мережа: об'єкт PetriNet, що містить Місця та Переходи
 часМоделювання: загальний час моделювання
 відстежуватиСтатистику: функція для збору статистики

ЗМІННІ:

списокМісць: масив Місць (PetriPlaces) з мережі
 списокПереходів: масив Переходів (PetriTransitions) з мережі
 поточнийЧас: поточний час моделювання
 мінімальнийЧас: час найближчої події
 найближчаПодія: перехід із найближчим часом події

ОСНОВНИЙ АЛГОРИТМ:

поточнийЧас = 0
 мінімальнийЧас = MAX_VALUE
 найближчаПодія = null

мінімальнийЧас = ОБРОБИТИ_ВХІД(списокМісць, списокПереходів, поточнийЧас,
 найближчаПодія)

найближчаПодія = ЗНАЙТИ_МІН_ПОДІЮ(списокПереходів)

поки (поточнийЧас < часМоделювання) виконуємо

відстежуватиСтатистику(поточнийЧас)
 поточнийЧас = мінімальнийЧас

// ФАЗА ВИХОДУ

якщо (поточнийЧас \leq часМоделювання) тоді
 найближчаПодія.ВИКОНАТИ_ВИХІД(списокМісць, поточнийЧас)
 ОБРОБИТИ_БУФЕРИЗОВАНІ_ПОДІЇ(найближчаПодія, списокМісць,
 поточнийЧас)

 для кожного переходу із списокПереходів
 якщо (перехід.буфер > 0

 |

 ОНОВИТИ_ТА_ОТРИМАТИ_МІНІМАЛЬНИЙ_ЧАС(перехід) == поточнийЧас) тоді
 перехід.ВИКОНАТИ_ВИХІД(списокМісць, поточнийЧас)
 ОБРОБИТИ_БУФЕРИЗОВАНІ_ПОДІЇ(перехід,
 списокМісць, поточнийЧас)

 кінець якщо

 кінець для

кінець якщо

// ФАЗА ВХОДУ

 мінімальнийЧас = ОБРОБИТИ_ВХІД(списокМісць, списокПереходів,
 поточнийЧас, найближчаПодія)
 найближчаПодія = ЗНАЙТИ_МІН_ПОДІЮ(списокПереходів)
 кінець поки

ПІДПРОГРАМА: ОБРОБИТИ_ВХІД(списокМісць, списокПереходів, поточнийЧас,
 найближчаПодія)

 активніПереходи = ЗНАЙТИ_АКТИВНІ_ПЕРЕХОДИ(списокМісць, списокПереходів)
 мінімальнийЧас = MAX_VALUE

 якщо (активніПереходи порожній І ВСІ_БУФЕРИ_ПОРОЖНІ(списокПереходів)) тоді
 повернути мінімальнийЧас
 кінець якщо

 поки (активніПереходи не порожній) виконуємо
 обранийПерехід = ВИРІШИТИ_КОНФЛІКТ(активніПереходи)
 обранийПерехід.ВИКОНАТИ_ВХІД(списокМісць, поточнийЧас)
 активніПереходи = ЗНАЙТИ_АКТИВНІ_ПЕРЕХОДИ(списокМісць,
 списокПереходів)
 кінець поки

```

    для кожного переходу із списокПереходів
        часМожливоїПодії
ОНОВИТИ_ТА_ОТРИМАТИ_МІНІМАЛЬНИЙ_ЧАС(перехід)
        якщо (часМожливоїПодії < мінімальнийЧас) тоді
            мінімальнийЧас = часМожливоїПодії
        кінець якщо
    кінець для

    повернути мінімальнийЧас

```

ПІДПРОГРАМА: ЗНАЙТИ_АКТИВНІ_ПЕРЕХОДИ(списокМісць, списокПереходів)
 активніПереходи = порожній список

```

    для кожного переходу із списокПереходів
        якщо (перехід.УМОВА_ВИКОНУЄТЬСЯ(списокМісць) І перехід.імовірність ≠
0) тоді
            додати перехід до активніПереходи
        кінець якщо
    кінець для

    якщо (активніПереходи.розмір > 1) тоді
        відсортувати активніПереходи за пріоритетом за спаданням
    кінець якщо

    повернути активніПереходи

```

ПІДПРОГРАМА: ВИРІШИТИ_КОНФЛІКТ(переходи)

```

    якщо (переходи.розмір ≤ 1) тоді
        повернути переходи[0]
    кінець якщо

    обранийПерехід = переходи[0]
    к-стьОднаковогоПріоритету = ПОРАХУВАТИ(переходи, де перехід.пріоритет ==
обранийПерехід.пріоритет)

    якщо (к-стьОднаковогоПріоритету == 1) тоді
        повернути обранийПерехід
    кінець якщо

    випадковеЧисло = ЗГЕНЕРУВАТИ_ВИПАДКОВЕ(0, 1)
    сумаІмовірностей = 0

```

```

    для кожного переходу із переходи, де перехід.пріоритет ==
    обранийПерехід.пріоритет
        якщо (перехід.імовірність == 1.0) тоді
            імовірність = 1.0 / к-стьОднаковогоПріоритету
        інакше
            імовірність = перехід.імовірність
        кінець якщо

    сумалімовірностей = сумалімовірностей + імовірність

    якщо (випадковеЧисло < сумалімовірностей) тоді
        повернути перехід
    кінець якщо
кінець для

повернути обранийПерехід

```

ПІДПРОГРАМА: ОБРОБИТИ_БУФЕРИЗОВАНІ_ПОДІЇ(перехід, списокМісць, поточнийЧас)

```

    поки (перехід.буфер > 0) виконуємо
        перехід.ВИДАЛИТИ_МІН_ПОДІЮ()

        якщо (ОНОВИТИ_ТА_ОТРИМАТИ_МІНІМАЛЬНИЙ_ЧАС(перехід) ==
    поточнийЧас) тоді
            перехід.ВИКОНАТИ_ВИХІД(списокМісць, поточнийЧас)
        інакше
            перервати
        кінець якщо
    кінець поки

```

ПІДПРОГРАМА: ЗНАЙТИ_МІН_ПОДІЮ(списокПереходів)

```

    мінПерехід = null
    час = MAX_VALUE

    для кожного переходу із списокПереходів
        часПодії = ОНОВИТИ_ТА_ОТРИМАТИ_МІНІМАЛЬНИЙ_ЧАС(перехід)
        якщо (часПодії < час) тоді
            мінПерехід = перехід
            час = часПодії
        кінець якщо
    кінець для

```

повернути мінПерехід

ПІДПРОГРАМА: ВИКОНАТИ_ВХІД(перехід, списокПозицій, поточнийЧас)

якщо (УМОВА_ВИКОНУЄТЬСЯ(перехід, списокПозицій)) тоді

// Зменшуємо кількість маркерів у вхідних позиціях (згідно з логікою умови входу)

для кожної позиції, яка визначена як вхідна для цього переходу

зменшити кількість маркерів у позиції на значення, що відповідає

логіці входу

кінець для

// Додаємо час завершення обробки в список часів виходу

перехід.списокЧасівВиходу.додати(поточнийЧас

+

ЗГЕНЕРУВАТИ_ЧАС_ОБРОБКИ())

// Збільшуємо буфер переходу

перехід.буфер++

// Оновлюємо мінімальний час події

ОНОВИТИ_ТА_ОТРИМАТИ_МІНІМАЛЬНИЙ_ЧАС(перехід)

кінець якщо

ПІДПРОГРАМА: ВИКОНАТИ_ВИХІД(перехід, списокПозицій, поточнийЧас)

якщо (перехід.буфер > 0) тоді

для кожної позиції

збільшуємо кількість маркерів позиції на таке значення, що відповідає

умові виходу

кінець для

якщо (перехід.num == 0)

і (перехід.списокЧасівВиходу.розмір == 1) тоді

перехід.списокЧасівВиходу[0] = MAX_VALUE

інакше

перехід.списокЧасівВиходу.видалити(перехід.num)

кінець якщо

перехід.буфер = перехід.буфер - 1

кінець якщо

ПІДПРОГРАМА: ОНОВИТИ_ТА_ОТРИМАТИ_МІНІМАЛЬНИЙ_ЧАС(перехід)

перехід.мінімальнийЧас = MAX_VALUE

якщо перехід.списокЧасівВиходу не пустий:

для кожного i у проміжку(0, перехід.списокЧасівВиходу.РОЗМІР()):

якщо $\text{перехід.списокЧасівВиходу}[i] < \text{перехід.мінімальнийЧас}$;
 $\text{перехід.мінімальнийЧас} = \text{перехід.списокЧасівВиходу}[i]$;
 $\text{перехід.індексМінімальногоЧасу} = i$;
 кінець для
 кінець якщо

20. Математичний опис стохастичної мережі Петрі.

(Лекція 6. Формалізм мережі Петрі 35, Лекція 11 Математична теорія стохастичної мережі Петрі 2)

$N = (P, T, A, W, K, R)$

$P = \{P\}$ – множина позицій;

$T = \{T\}$ – множина переходів;

$P \cap T = \emptyset$

$A \subseteq P \times T \cup T \times P$ – множина дуг;

$W: A \rightarrow \mathbb{N}$ – множина натуральних чисел, що задають кратності дуг (кількість зв'язків);

$K = \{(c_T, b_T) | T \in T, c_T \in \mathbb{N}, b_T \in [0; 1]\}$ – множина пар значень, що задають пріоритет та ймовірність запуску переходів;

$R: T \rightarrow R_+$ – множина невід'ємних чисел, що характеризують часові затримки;

- $T, T \bullet$ – множина вхідних та множина вихідних позицій переходу T ;
- $P, P \bullet$ – множина вхідних і множина вихідних переходів позиції P ;

21. Рівняння станів стохастичної мережі Петрі

(Лекція 11 Математична теорія стохастичної мережі Петрі 3)

$S(t) = (M(t), E(t))$ – стан стохастичної мережі Петрі в момент часу t ;

$M(t) = \{M_P(t) | M_P(t) \in Z_+, P \in \mathcal{P}\}$ – стан позицій;

$E(t) = \{E_T(t) | T \in \mathcal{T}\}$ – стан переходів;

$E_T(t) = \{[E(t)] | [E(t)] \in \mathcal{R}, q \in \mathcal{N}\}$ – стан переходу T , q – номер запланованої події

виходу маркерів з переходу в момент часу t ;

$E_T(t) = \{\infty\}$; якщо найближчим часом не очікується вихід маркерів з переходу;

$S(t) \in \{S(t) | M_P(t) \geq 0 \forall P \in \mathcal{P}\} \wedge ([E_T(t)]_q \geq 0 \forall T \in \mathcal{T}, \forall q = 1, \dots, |E_T(t)|)\}$

21.1. Визначення моменту найближчої події

$t_T(t) = \min [E_T(t)]_q$ – момент запланованої найближчої події для переходу T на поточний момент часу;

$t^\bullet = \min_T \tau_T(t), t^\bullet \geq t$ – момент запланованої найближчої події для мережі Петрі на поточний момент часу;

$t_n = \min_T (\min_q [E_T(t_{n-1})]_q), t_n \geq t_{n-1}$ – визначення моменту найближчої події для мережі Петрі на поточний момент часу;

22. Матричні рівняння базової мережі Петрі

(Лекція 9 Математична теорія базової мережі Петрі 5)

Матричний опис базової мережі Петрі

Вектор запуску переходу $v = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ Запуск переходу T_3 Умова запуску переходу $\forall i \ M_i \geq (a^- \cdot v)_i$

Результат запуску переходу $M' = M - a^- \cdot v + a^+ \cdot v$

$$M' = M + (a^+ - a^-) \cdot v$$

Матриця змінювань $a = a^+ - a^-$

Результат запуску послідовності $T_1 - T_2 - T_1$ $M' = M + a \cdot v^{(1)}$

$$M'' = M' + a \cdot v^{(2)}$$

$$M''' = M'' + a \cdot v^{(3)}$$

$$M''' = M + a \cdot (v^{(1)} + v^{(2)} + v^{(3)})$$

$$M''' = M + a \cdot v$$

Вектор запуску переходів

$$v = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Кількість запусків переходу T_1

Матричне рівняння станів базової мережі Петрі

Результуюче маркірування
(після запуску переходів)

Початкове маркірування

$$M' = M + a \cdot v$$

Матриця змінювань

Вектор запуску переходів

Змінювання маркірування

$$\Delta M = a \cdot v$$

23. Матричні рівняння станів стохастичної мережі Петрі

(Лекція 11 Математична теорія стохастичної мережі Петрі 13)

Матричний опис стану стохастичної мережі Петрі

$$a_{p,T}^+ = \begin{cases} W_{T,P}, T \in \bullet P \\ 0, T \notin \bullet P \end{cases} \quad \text{- матриця виходів} \quad \mathbf{a}^+ = \|a_{T,P}^+\| \quad \mathbf{a} = \mathbf{a}^+ - \mathbf{a}^-$$

$$a_{p,T}^- = \begin{cases} W_{P,T}, T \in P^\bullet \setminus P^\circ \\ 0, T \notin P^\bullet \setminus P^\circ \end{cases} \quad \text{- матриця входів} \quad \mathbf{a}^- = \|a_{T,P}^-\|$$

$$\mathbf{v}(t) = \|v_T(t)\| \quad \text{- вектор кількості активних каналів переходів} \quad v_T(t) = \begin{cases} \lfloor E_T(t) \rfloor, \tau_T < \infty, \\ 0, \tau_T = \infty, \end{cases}$$

$$\boldsymbol{\gamma}(t) = \|\gamma_T(t)\| \quad \text{- вектор кількості входів в переходи за період часу } [t_0, t] \quad \sum_{j=1}^n Z(T, t_j) \cdot u_T(t_j) = \gamma_T(t_n)$$

$$\boldsymbol{\eta}(t) = \|\eta_T(t)\| \quad \text{- вектор кількості виходів з переходів за період часу } [t_0, t] \quad \sum_{j=1}^n Y(T, t_j) \cdot |s_T(t_{j-1})| = \eta_T(t_n)$$

$$\mathbf{M}(t) = M_P(t) \quad \text{- вектор маркірування}$$

$$\boldsymbol{\mu}(t) = \mathbf{M}(t) + \mathbf{a}^+ \cdot \mathbf{v}(t) \quad \text{- вектор розширеного маркірування}$$

!!!! враховує не тільки маркери, які в позиціях, але й маркери, які очікують виходу з переходів

(Лекція 11 Математична теорія стохастичної мережі Петрі 14)

Матричні рівняння станів стохастичної мережі Петрі з часовими затримками, з багатоканальними та конфліктними переходами, з інформаційними зв'язками

$$\mathbf{M}(t_n) + \mathbf{a}^+ \cdot \mathbf{v}(t_n) - (\mathbf{M}(t_0) + \mathbf{a}^+ \cdot \mathbf{v}(t_0)) = \mathbf{a}^+ \cdot \boldsymbol{\gamma}(t_n) - \mathbf{a}^- \cdot \boldsymbol{\gamma}(t_n)$$

$$\boldsymbol{\eta}(t_n) = -\mathbf{v}(t_n) + \mathbf{v}(t_0) + \boldsymbol{\gamma}(t_n)$$

В термінах вектора розширеного маркірування:

$$\boldsymbol{\mu}(t_n) = \boldsymbol{\mu}(t_0) + \mathbf{a} \cdot \boldsymbol{\gamma}(t_n)$$

$$\boldsymbol{\eta}(t_n) = -\mathbf{v}(t_n) + \mathbf{v}(t_0) + \boldsymbol{\gamma}(t_n)$$

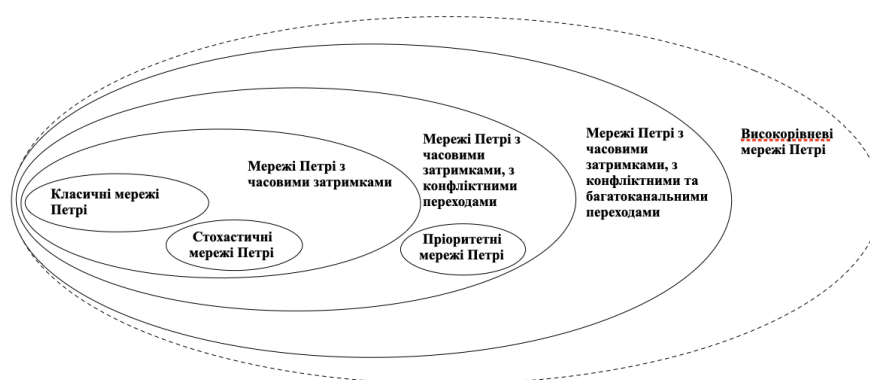
Аналогічне базовому, але сформульованого для розширеного маркірування та вектора кількості входів в перехід

Додаткове рівняння формулює залежність між кількістю запусків, кількістю входів в переходи та кількістю виходів з переходів

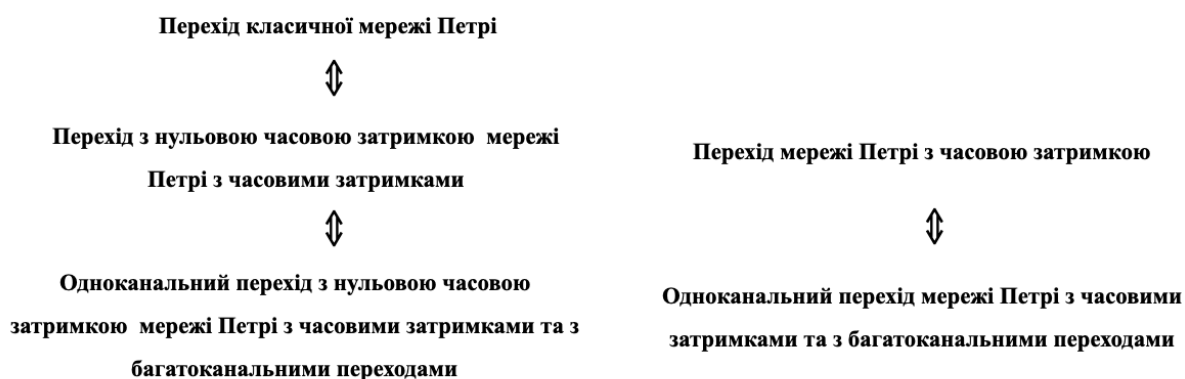
24. Еквівалентні відношення між базовою, детермінованою та стохастичною мережами Петрі

(Лекція 12 Формалізм Петрі-об'єктної моделі 2)

Взаємне співвідношення класів мереж Петрі



Еквівалентності переходів різних класів мереж Петрі



25. Дослідження властивостей мереж Петрі. Інваріанти мережі Петрі

(Модельовання Систем Посібник 136, Лекція 9 Математична теорія базової мережі Петрі 27)

Аналітичне дослідження властивостей мережі Петрі	Властивість	Спосіб дослідження	
		Матричний підхід	Дерево досяжності
	k-обмеженість	Не досліджується	Необхідна і достатня
	зберігання	Необхідна і достатня умова	Необхідна і достатня
	досяжність	Тільки необхідна умова	Послідовність переходів залишається невідомою
	активність	Не досліджується	Не досліджується

(Лекція 9 Математична теорія базової мережі Петрі 8)

Властивість	Визначення
k-обмеженість	Якщо кількість маркерів в будь-якій позиції мережі Петрі не перевищує k маркерів, то мережа являється k – обмеженою.
досяжність	Досяжністю мережі Петрі називається множина досяжних маркірувань.
збереження	Якщо в мережі Петрі неможливе виникнення і знищення ресурсів, то мережа володіє властивістю збереження.
активність	Якщо з будь-якого досяжного початкового стану можливий перехід в будь-який інший досяжний стан, то мережа Петрі володіє властивістю активності.

25.1. Збереження (консервативність)

(Лекція 9 Математична теорія базової мережі Петрі 9)

Збереження (консервативність) – Якщо існує вектор w , компоненти якого цілі додатні числа, такий що $w^T \cdot M = w^T \cdot M^{\bullet}$ для будь-якого досяжного з початкового маркування, то мережа Петрі володіє властивістю збереження.

$$w^T \cdot M = w^T \cdot M^{\bullet}, \forall M^{\bullet} (\text{досяжного})$$

$$w^T \cdot M = w^T \cdot (M + a \cdot v), \forall v$$

$$0 = w^T \cdot a \cdot v, \forall v$$

$$0 = w^T \cdot a$$

$$0^T = (w^T \cdot a)^T$$

$$0^T = a^T \cdot w$$

Твердження. Мережа Петрі володіє властивістю зберігання тоді і тільки тоді, коли існує вектор w , компоненти якого цілі додатні числа, такий, що $a^T \cdot w = 0, w_i \in Z_+$.

25.2. S – інваріант мережі Петрі

(Лекція 9 Математична теорія базової мережі Петрі 10)

Розв'язки рівняння $a^T \cdot w = 0$, де w – невідомий вектор розміру $|P| \times 1$, називають S-інваріантом мережі Петрі.

S-інваріант, або інваріант стану, дозволяє досліджувати консервативність системи. Консервативність означає, що існує зважена сума маркувань позицій мережі Петрі, яка для будь-якого досяжного маркування залишається незмінною. Рівняння, які формулюються і розв'язуються в термінах цілих чисел, називають діофантовими.

25.3. Циклічність

(Лекція 9 Математична теорія базової мережі Петрі 11)

Якщо існує послідовність запусків переходів, така що мережа повертається в початкове маркування, то функціонування мережі Петрі є циклічним.

$$M^{\bullet} = M$$

$$M + a \cdot v = M, \exists v$$

$$0 = a \cdot v, \exists v$$

Функціонування мережі Петрі є циклічним тоді і тільки тоді, коли існує вектор v , компоненти якого цілі невід'ємні числа, такий, що $a \cdot v = 0, v_i \in Z_+$.

25.4. T – інваріант мережі Петрі

(Лекція 9 Математична теорія базової мережі Петрі 12)

Розв'язки рівняння $a \cdot v = 0$, де v – невідомий вектор розміру $|P| \times 1$, називають T-інваріантом мережі Петрі.

T-інваріант, або інваріант функціонування, означає досяжність початкового маркування. Цей інваріант є важливим для дослідження циклічності процесів функціонування.

Циклічність означає існування такої послідовності запусків переходів, що мережа Петрі повертається в початкове маркування. Наявність T-інваріантів гарантує циклічність функціонування системи.

25.5. Досяжність

(Лекція 9 Математична теорія базової мережі Петрі 13)

Існування невід'ємного цілого вектора запуску переходів, що задовольняє рівнянню $M' = M + a \cdot v$, є тільки необхідною, але не достатньою умовою.

25.6. Активність

(Лекція 9 Математична теорія базової мережі Петрі 14)

Рівень активності	Перехід T має рівень активності A , якщо
0	він ніколи не може бути запущений
1	існує маркірування (досягне з початкового), яке дозволяє запуск цього переходу T
2	для довільного цілого числа n існує послідовність запусків переходів, в якій перехід T присутній принаймні n раз
3	існує нескінченна послідовність запусків переходів, в якій перехід T присутній необмежено багато разів
4	якщо для довільного маркірування M , що є досяжним з початкового маркірування, існує послідовність запусків переходів, яка призводить до маркірування, що дозволяє запуск переходу T

25.7. Дерево досяжності

(Лекція 9 Математична теорія базової мережі Петрі 21)

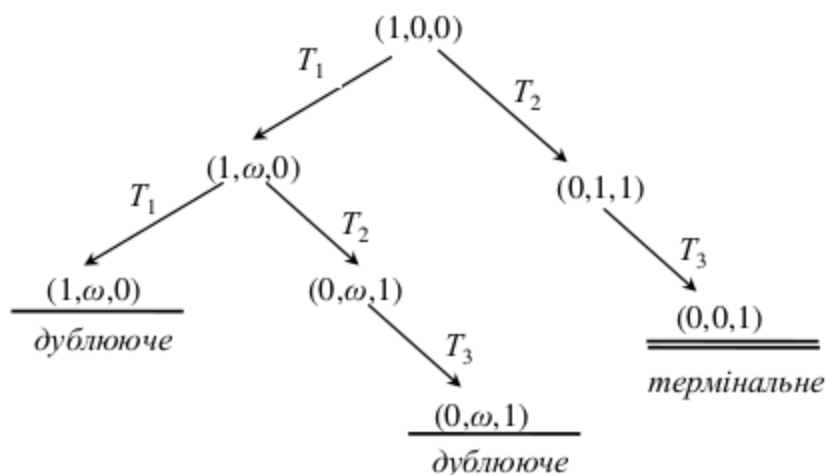
Дерево досяжності представляє множину досяжних маркувань мережі Петрі. Дерево досяжності розпочинається з початкового маркування, а закінчується термінальним або дублюючим маркуванням.

Термінальним маркуванням називається маркування, в якому жоден з переходів мережі Петрі не запускається.

Дублюючим маркіруванням називається маркірування, що раніше зустрічалося в дереві досяжності.

Символ ω в позиції M_j маркування M з'являється тоді, коли на шляху до маркування M спостерігається маркування M' , в якому всі значення, крім j -ого, не перевищують значення маркування M , а j -е значення є меншим. Одного разу з'явившись, символ ω уже

не змінюється і не зникає в дереві досяжності: додавання або віднімання від нескінченності є нескінченність.



26. Формалізм Петрі-об'єктної моделі

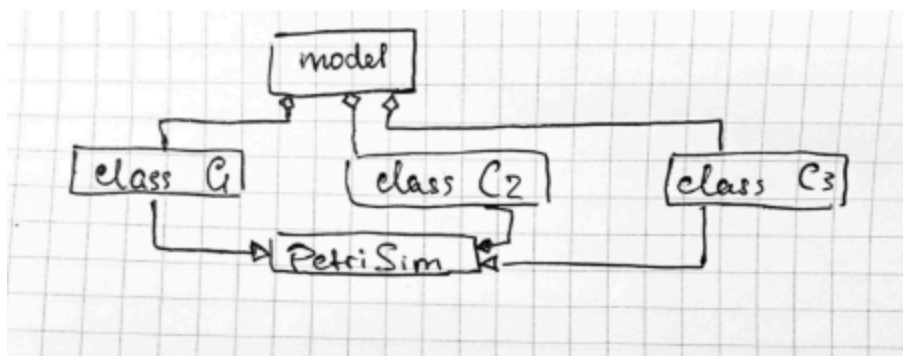
Петрі-об'єктом (PetriObj) називатимемо об'єкт, що є нащадком inherit об'єкта Петрі-імітатор(PetriSim)

Петрі-об'єкти:

- володіють всіма властивостями звичайного об'єкта (як елемента ООП)
- імітують функціонування об'єкта на основі мережі Петрі, опис якої міститься в полі PetriNet
- є конструктивними елементами, з яких складається мережу Петрі складної системи

PetriSim – такий клас, що відтворює імітацію деякого реального об'єкту та враховує його важливі функціональні залежності.

Петрі-об'єктною моделлю називатимемо модель, що отримана в результаті агрегування Петрі-об'єктів.



де об'єкту O_N

$$N = (P_N, T_N, A_N, W_N, K_N, J_N, R_N)$$

$P_N = \{P\}$ мн. позицій

$T_N = \{T\}$ мн. переходів

$$P_N \cap T_N = \emptyset$$

$A_N \subseteq (P_N \times T_N \cup T_N \times P_N)$ - мн. дуг

$J_N \subseteq (P_N \times T_N)$ - мн. ініц. дуг

$W: A \cup J \rightarrow N$ - мн. натур. чисел, що задає кратності дуг

$K = \{(c_t, v_t) \mid T \in T, c_t \in N, v_t \in [0; +\infty)\}$ - мн. пар значень, що задає пріоритет та веровітність переходів

$R: T \rightarrow \mathbb{R}_+$ - мн. невід'ємних значень, що характеризує часові задержки

Мережа Петрі об'єкта створюється за допомогою статичної функції класу NetLibrary і

потім передається конструктору Петрі-об'єкта в якості аргумента. Конструктор у свою

чергу ставить дану мережу у потрібне поле.

Зв'язки Петрі-об'єктів між собою можуть здійснюватися:

- за допомогою спільних позицій (загальна позиція є позицією мережі Петрі декількох різних Петрі-об'єктів)

- за допомогою ініціалізації подій (з переходу мережі Петрі об'єкта O_N при

кожному виході маркерів з переходу передаються маркери в позицію мережі Петрі об'єкта OJ в заданій кількості wT , P в момент часу, відповідний моменту виходу маркерів з переходу).

27. Математичний опис Петрі-об'єктної моделі

(Лекція 14 Теоретичні основи Петрі-об'єктного моделювання 2)

Твердження 1

Петрі-об'єктна модель описується стохастичною мережею Петрі, що є об'єднанням мереж Петрі-об'єктів, з яких вона складається:

$$ModelNet = \bigcup_{\tilde{N}} \tilde{N}$$

де $\tilde{N} = (\tilde{P}_N, T_N, \tilde{A}_N, \tilde{W}_N, K_N, I_N, R_N)$

$$\tilde{P}_N = U_N(\cdot T_N \cup T_N \cdot)$$

$$\bigcup_{\tilde{N}} \tilde{N}: P = \bigcup_N \tilde{P}_N \quad T = \bigcup_N T_N \quad \tilde{A} = \bigcup_N \tilde{A}_N \quad \tilde{W} = \bigcup_N \tilde{W}_N \quad K = \bigcup_N K_N \quad I = \bigcup_N I_N \quad R = \bigcup_N R_N$$

$$T_N \cdot = \bigcup_{T \in T_N} T \cdot = \{P \in P \mid \exists T \in T: \exists (T, P) \in \tilde{A}_N\}$$

$$\tilde{A}_N = A_N \cup U_N \quad \tilde{W}_N = W_N \cup w_N$$

$$U_N = \{(T, P) \mid T \in T_k, P \in P_l, w_{k,l} > 0\}$$

- множина дуг Петрі- об'єкта, що з'єднує його з іншими об'єктами через ініціалізацію подій

Наслідок. Петрі-об'єктная модель є обчислюваною.

(Лекція 14 Теоретичні основи Петрі-об'єктного моделювання 3)

Твердження 2

Перетворення D^+ мережі Петрі-об'єктної моделі $\bigcup_{\tilde{N}} \tilde{N}$
еквівалентно перетворенню D^+ мереж Петрі-об'єктів

$$\tilde{N} = (\mathbf{T}_N^\bullet, \mathbf{T}_N, \tilde{\mathbf{A}}_N, \tilde{\mathbf{W}}_N, \mathbf{K}_N, \mathbf{I}_N, \mathbf{R}_N)$$

Наслідок. Стан Петрі-об'єктної моделі, який є результатом виходу маркерів з переходів мережі Петрі-об'єктної моделі, описується станом її Петрі-об'єктів:

$$\mathbf{S}^+(t_n) = D^+(\mathbf{S}(t_{n-1})) = \begin{pmatrix} D^+(\tilde{\mathbf{S}}_1(t_{n-1})) \\ \dots \\ D^+(\tilde{\mathbf{S}}_N(t_{n-1})) \\ \dots \\ D^+(\tilde{\mathbf{S}}_L(t_{n-1})) \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{S}}_1^+(t_n) \\ \dots \\ \tilde{\mathbf{S}}_N^+(t_n) \\ \dots \\ \tilde{\mathbf{S}}_L^+(t_n) \end{pmatrix}$$

(Лекція 14 Теоретичні основи Петрі-об'єктного моделювання 4)

Твердження 3

Перетворення D^- мережі Петрі-об'єктної моделі $\bigcup_{\tilde{N}} \tilde{N}$

еквівалентно перетворенню D^- мереж Петрі-об'єктів

$$\tilde{N} = (\mathbf{T}_N^*, \mathbf{T}_N, \tilde{\mathbf{A}}_N, \tilde{\mathbf{W}}_N, \mathbf{K}_N, \mathbf{I}_N, \mathbf{R}_N),$$

для яких у випадку існування спільних позицій Петрі-об'єктів вирішений конфлікт

Наслідок. Стан Петрі-об'єктної моделі, який є результатом входу маркерів в переходи мережі Петрі-об'єктної моделі, описується станом її Петрі-об'єктів.

$$\mathbf{S}(t_n) = D^-(\mathbf{S}^+(t_n)) = \begin{pmatrix} D^-(\tilde{\mathbf{S}}_1^+(t_n)) \\ \dots \\ D^-(\tilde{\mathbf{S}}_N^+(t_n)) \\ \dots \\ D^-(\tilde{\mathbf{S}}_L^+(t_n)) \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{S}}_1(t_n) \\ \dots \\ \tilde{\mathbf{S}}_N(t_n) \\ \dots \\ \tilde{\mathbf{S}}_L(t_n) \end{pmatrix}$$

(Лекція 14 Теоретичні основи Петрі-об'єктного моделювання 5)

Рівняння станів Петрі-об'єктної моделі

Наслідок. Стан Петрі-об'єктної моделі в кожний момент часу описується станом її Петрі-об'єктів:

$$S(t_n) = (D^-)^m (D^+ (S(t_{n-1}))) = (D^-)^m \begin{pmatrix} D^+ (\tilde{S}_1(t_{n-1})) \\ \dots \\ D^+ (\tilde{S}_N(t_{n-1})) \\ \dots \\ D^+ (\tilde{S}_L(t_{n-1})) \end{pmatrix} = \begin{pmatrix} (D^-)^m (D^+ (\tilde{S}_1(t_{n-1}))) \\ \dots \\ (D^-)^m (D^+ (\tilde{S}_N(t_{n-1}))) \\ \dots \\ (D^-)^m (D^+ (\tilde{S}_L(t_{n-1}))) \end{pmatrix}$$

**Рівняння станів
Петрі-об'єктної
моделі**

$$t_n = \min_N \tau_N, t_n \geq t_{n-1}$$

$$S(t_n) = \begin{pmatrix} (D^-)^m (D^+ (\tilde{S}_1(t_{n-1}))) \\ \dots \\ (D^-)^m (D^+ (\tilde{S}_N(t_{n-1}))) \\ \dots \\ (D^-)^m (D^+ (\tilde{S}_L(t_{n-1}))) \end{pmatrix}$$

$$S(t_1) = \begin{pmatrix} (D^-)^m (S_1(t_0)) \\ \dots \\ (D^-)^m (S_N(t_0)) \\ \dots \\ (D^-)^m (S_L(t_0)) \end{pmatrix}$$

$$\forall \tilde{S}_N(t_n): \bigvee_{T \in T_N} Z(T, t_n) = 0$$

(Лекція 14 Теоретичні основи Петрі-об'єктного моделювання 7)

Матричні рівняння станів Петрі-об'єктної моделі

Матричні рівняння Петрі-об'єктів $\tilde{N} = (\mathbf{T}_N^*, \mathbf{T}_N, \tilde{\mathbf{A}}_N, \tilde{\mathbf{W}}_N, \mathbf{K}_N, \mathbf{I}_N, \mathbf{R}_N)$

$$\boldsymbol{\mu}_N(t_n) = \boldsymbol{\mu}_N(t_0) + \mathbf{a}_N \cdot \boldsymbol{\gamma}_N(t_n)$$

$$\boldsymbol{\eta}_N(t_n) = -\mathbf{v}_N(t_n) + \mathbf{v}_N(t_0) + \boldsymbol{\gamma}_N(t_n)$$

$\boldsymbol{\mu}_N(t) = \tilde{\mathbf{M}}(t) + \mathbf{a}_N^+ \cdot \mathbf{v}_N(t)$ - вектор розширеного маркірування Петрі-об'єкта

$\mathbf{a}_N = \mathbf{a}_N^+ - \mathbf{a}_N^-$ - матриця інцидентності

$\mathbf{v}_N(t)$ - вектор кількості активних каналів переходів Петрі-об'єкта

$\boldsymbol{\gamma}_N(t_n)$ - вектор кількості входів в переход Петрі-об'єкта за період часу $[t_0, t]$

$\boldsymbol{\eta}_N(t_n)$ - вектор кількості виходів з переходів Петрі-об'єкта за період часу $[t_0, t]$

(Лекція 14 Теоретичні основи Петрі-об'єктного моделювання 8)

Матричні рівняння станів Петрі-об'єктної моделі

$$\boldsymbol{\mu}(t) = \|\mu_P(t)\| \quad \forall P \in \mathbf{P}_N \quad [\boldsymbol{\mu}(t)]_P = [\boldsymbol{\mu}_N(t)]_P$$

$$\mathbf{v}(t) = \|v_T(t)\| \quad \forall T \in \mathbf{T}_N \quad [\mathbf{v}(t)]_T = [\mathbf{v}_N(t)]_T \quad \mathbf{a}^- = \|a_{P,T}^-\| \quad \forall P \in \mathbf{P}_N \quad \forall T \in \mathbf{T}_N \quad [\mathbf{a}^-]_{P,T} = [\mathbf{a}_N^-]_{P,T}$$

$$\boldsymbol{\gamma}(t) = \|\gamma_T(t)\|: \quad \forall T \in \mathbf{T}_N \quad [\boldsymbol{\gamma}(t)]_T = [\boldsymbol{\gamma}_N(t)]_T \quad \mathbf{a}^+ = \|a_{P,T}^+\| \quad \forall P \in \mathbf{P}_N \quad \forall T \in \mathbf{T}_N \quad [\mathbf{a}^+]_{P,T} = [\mathbf{a}_N^+]_{P,T}$$

$$\boldsymbol{\eta}(t) = \|\eta_T(t)\|: \quad \forall T \in \mathbf{T}_N \quad [\boldsymbol{\eta}(t)]_T = [\boldsymbol{\eta}_N(t)]_T$$

Фундаментальні рівняння Петрі-об'єктної моделі:

$$\boldsymbol{\mu}(t_n) = \boldsymbol{\mu}(t_0) + \mathbf{a} \cdot \boldsymbol{\gamma}(t_n)$$

$$\boldsymbol{\eta}(t_n) = -\mathbf{v}(t_n) + \mathbf{v}(t_0) + \boldsymbol{\gamma}(t_n)$$

28. Алгоритм імітації Петрі-об'єктної моделі

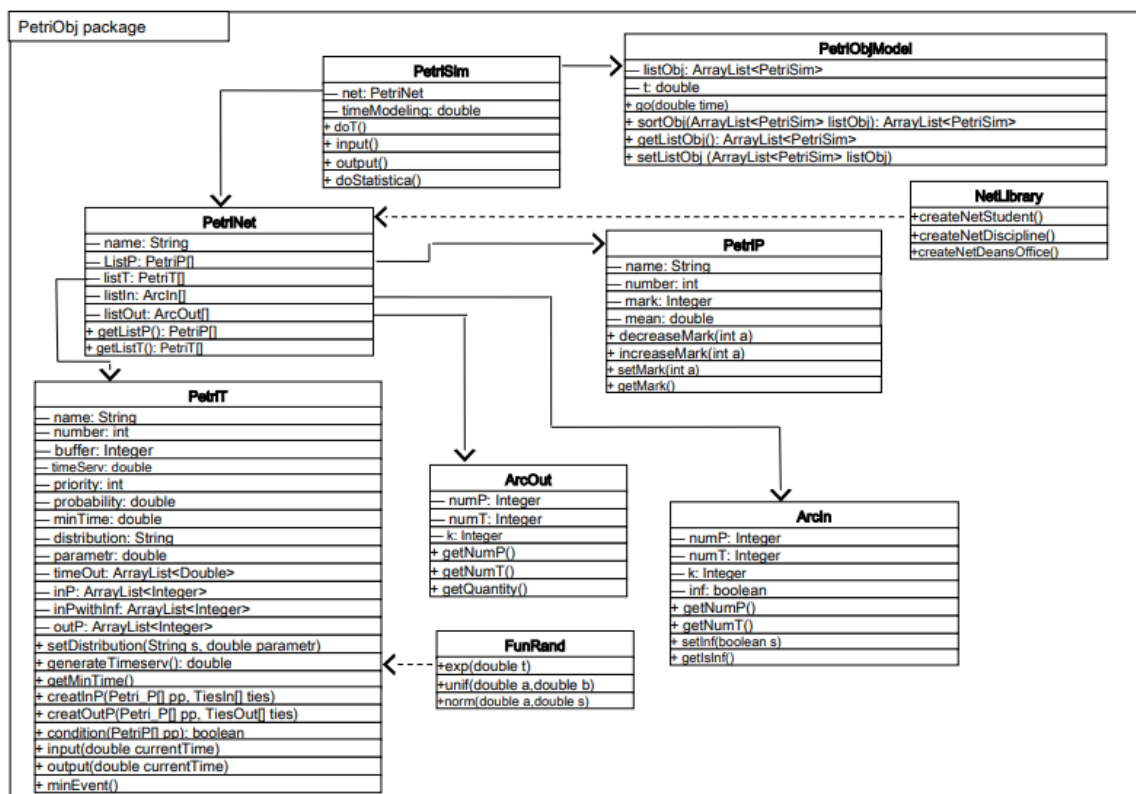
(Лекція 14 Теоретичні основи Петрі-об'єктного моделювання 6)

- Формувати список Петрі-об'єктів;
- Виконати перетворення $(D^-)^m$ (метод input());
- Доки не досягнутий момент завершення моделювання
 - Просунути час в момент найближчої події;
 - визначити список конфліктних об'єктів та вибрати об'єкт із списку конфліктних об'єктів;
 - для вибраного об'єкта виконати перетворення $(D^-)^m \circ D^+$ (методи output(), input(), doT());
 - для всіх інших об'єктів виконати перетворення $(D^-)^m$ (метод input());
- Вивести результати моделювання.

29. Бібліотека класів Петрі-об'єктного моделювання

(Лекція 15 ПЗ Петрі-об'єктного моделювання)

Бібліотека класів Петрі-об'єктного моделювання



© Стеценко Інна Вячеславівна КПІ ім. Ігоря Сікорського

PetriObjModel

Призначення: Основний клас для управління моделлю об'єктів Петрі.

Атрибути:

- **listObj**: Масив об'єктів PetriSim.
- **t**: Поточний час моделювання.

Методи:

- **go(double time)**: Імітація моделі до зазначеного часу.
- **sortObj()**: Сортування об'єктів моделі.
- **getListObj(), setListObj()**: Отримання/встановлення списку об'єктів.

PetriSim

Призначення: Клас для симуляції поведінки мережі Петрі.

Атрибути:

- `net`: Об'єкт PetriNet.
- `timeModeling`: Час моделювання.

Методи:

- `doT()`: Виконання переходів мережі.
- `input()`, `output()`: Обробка вхідних і вихідних даних.
- `doStatistics()`: Виконання статистичного аналізу.

PetriNet

- **Призначення:** Визначає структуру мережі Петрі.
- **Атрибути:**
 - `name`: Назва мережі.
 - `listP`: Список позицій.
 - `listT`: Список переходів.
 - `listArc`: Список дуг.
- **Методи:**
 - `getListP()`, `getListT()`: Отримання списку позицій і переходів.

PetriP

Призначення: Визначає позицію в мережі Петрі.

Атрибути:

- `name`: Назва позиції.
- `number`: Ідентифікатор.
- `mark`: Поточна мітка (кількість токенів).
- `mean`: Середнє значення токенів.

Методи:

- `increaseMark(int a)`, `decreaseMark(int a)`: Зміна кількості токенів.
- `setMark()`, `getMark()`: Встановлення/отримання токенів.

PetriT

Призначення: Визначає перехід у мережі Петрі.

Атрибути:

- `name`: Назва переходу.
- `buffer`: Буфер для токенів.
- `priority`: Пріоритет переходу.

- `minTime, timeOut`: Часові параметри.
- `inP, inWithInf`: Списки вхідних позицій.
- `outP`: Список вихідних позицій.
- **Методи:**
 - `setDistribution(String, double)`: Встановлення розподілу ймовірності.
 - `generateTimeServ()`: Генерація часу обслуговування.
 - `condition()`: Перевірка умови виконання переходу.

ArcOut та ArcIn

Призначення: Визначають дуги між позиціями та переходами.

Атрибути:

- `numP`: Ідентифікатор позиції.
- `numT`: Ідентифікатор переходу.
- `k`: Кількість токенів (для ArcIn).
- `inf`: Чи може дуга приймати нескінченні токени.

Методи:

- `getNumP()`, `getNumT()`: Отримання ідентифікаторів.
- `getQuantity()`: Отримання кількості токенів.

FunRand

Призначення: Генерація випадкових чисел для імітації різних розподілів.

Методи:

- `fexp(double t)`: Експоненційний розподіл.
- `funif(double a, double b)`: Рівномірний розподіл.
- `fnorm(double a, double s)`: Нормальний розподіл.

NetLibrary

Призначення: Шаблони для створення типових мереж.

Методи можна створювати самостійно та зберігати за допомогою зручного інтерфейсу для перевикористання

30. Програмне забезпечення Петрі-об'єктного моделювання

(Лекція 15 ПЗ Петрі-об'єктного моделювання 11)

```
public static PetriNet CreateNetSMOwithoutQueue(int numChannel, double timeMean,
String name) throws ExceptionInvalidTimeDelay, ExceptionInvalidNetStructure{
    ArrayList<PetriP> d_P = new ArrayList<>();
    ArrayList<PetriT> d_T = new ArrayList<>();
    ArrayList<ArcIn> d_In = new ArrayList<>();
    ArrayList<ArcOut> d_Out = new ArrayList<>();
    d_P.add(new PetriP("P1",0));
    d_P.add(new PetriP("P2",numChannel));
    d_P.add(new PetriP("P3",0));
    d_T.add(new PetriT("T1",timeMean,Double.MAX_VALUE));
    d_T.get(0).setDistribution("exp", d_T.get(0).getTimeServ());
    d_T.get(0).setParamDeviation(0.0);
    d_In.add(new ArcIn(d_P.get(0),d_T.get(0),1));
    d_In.add(new ArcIn(d_P.get(1),d_T.get(0),1));
    d_Out.add(new ArcOut(d_T.get(0),d_P.get(1),1));
    d_Out.add(new ArcOut(d_T.get(0),d_P.get(2),1));
    PetriNet d_Net = new PetriNet("SMOwithoutQueue"+name,d_P,d_T,d_In,d_Out);
    PetriP.initNext();
    PetriT.initNext();
    ArcIn.initNext();
    ArcOut.initNext();
    return d_Net;
}
```

(Лекція 15 ПЗ Петрі-об'єктного моделювання 14)

```
public static PetriObjModel getModel() throws ExceptionInvalidTimeDelay,
ExceptionInvalidNetStructure {
    ArrayList<PetriSim> list = new ArrayList<>();
    list.add(new PetriSim(NetLibrary.CreateNetGenerator(2.0)));
    list.add(new PetriSim(NetLibrary.CreateNetSMOwithoutQueue(1, 0.6,"First")));
    list.add(new PetriSim(NetLibrary.CreateNetSMOwithoutQueue(1, 0.3, "Second")));
    list.add(new PetriSim(NetLibrary.CreateNetSMOwithoutQueue(1, 0.4,"Third")));
    list.add(new PetriSim(NetLibrary.CreateNetSMOwithoutQueue(2, 0.1,"Forth")));
    list.add(new PetriSim(NetLibrary.CreateNetFork(0.15, 0.13, 0.3)));
    list.get(0).getNet().getListP()[1] = list.get(1).getNet().getListP()[0];
    list.get(1).getNet().getListP()[2] = list.get(5).getNet().getListP()[0];
    list.get(5).getNet().getListP()[1] = list.get(2).getNet().getListP()[0];
}
```

```

list.get(5).getNet().getListP()[2] = list.get(3).getNet().getListP()[0];
list.get(5).getNet().getListP()[3] = list.get(4).getNet().getListP()[0];
list.get(2).getNet().getListP()[2] = list.get(1).getNet().getListP()[0];
list.get(3).getNet().getListP()[2] = list.get(1).getNet().getListP()[0];
list.get(4).getNet().getListP()[2] = list.get(1).getNet().getListP()[0];
PetriObjModel model = new PetriObjModel(list);
return model;
}

```

31. Моделювання багатопоточних програм стохастичною мережею Петрі

(Лекція 16 Моделювання паралельних обчислень 6)

Проблеми паралельних програм:

- можливість виникнення взаємного блокування роботи потоків (deadlock), неможливість завершення роботи потоків (livelock), неможливість захоплення ресурсу потоком (starvation),
- помилка при спільному використанні ресурсу (memory consistency error, «гонка» потоків),
- сповільнення роботи потоків через синхронізацію дій.

Проблеми, що ускладнюють процес розробки та тестування паралельних програм:

- недетермінований порядок інструкцій, виконуваних потоками,
- залежність результату запуску паралельної програми від обчислювальних ресурсів, на яких вона запускається.

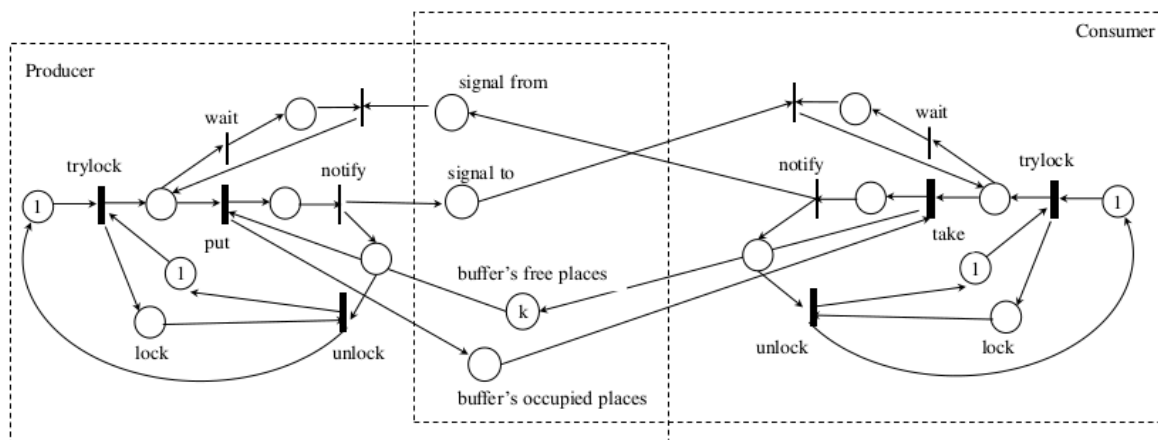
(Лекція 16 Моделювання паралельних обчислень 8)

Наприклад, м'ютекс – це певний ресурс, тобто позиція, операції trylock, unlock, catch – це переходи.

(Лекція 16 Моделювання паралельних обчислень 11)

Модель Producer – Consumer.

Приклад Guarded block [Oracle: The Java Tutorials]



32. Експериментальні методи дослідження моделей систем

(Модельовання Систем Посібник 225)

Експериментування є процес використання моделі з метою отримання і аналізу інформації про властивості модельованої системи.

1. На першому етапі планування дослідник задає:
 - 1.1. кількість варійованих факторів (факторами в теорії планування експериментів називають входні змінні, вплив яких на вихідну змінну, що називається відгуком моделі, досліджується);
 - 1.2. кількість рівнів кожного фактора (рівнями називають ті кількісні або якісні значення фактора, при яких можливо проводити експеримент);
 - 1.3. область змінювання факторів;
 - 1.4. необхідну точність та довірчу ймовірність вимірювання відгуку моделі.
2. На другому етапі приступають до тактичного планування експерименту, метою якого є визначення таких умов експерименту, що забезпечать вимір відгуку моделі з заданою точністю та довірчою ймовірністю. якщо про закон розподілу відгуку моделі нічого не відомо, то оцінка кількості прогонів, яка необхідна для забезпечення точності ε при довірчій ймовірності β , здійснюється за нерівністю Чебишева за формулою:
$$p = \frac{\sigma^2}{\varepsilon^2(1-\beta)}$$
. Наприклад, при довірчій ймовірності $\beta = 0.95$ та точності $\varepsilon = \sigma$ розраховуємо, що потрібно $p=20$ прогонів.

3. На третьому етапі приступають до стратегічного планування експериментів, метою якого є визначення такої серії експериментів, що забезпечить отримання бажаної інформації про відгук моделі при мінімальних витратах.
 - 3.1. Однофакторні плани експериментів дозволяють досліджувати вплив одного фактора на відгук моделі.
 - 3.2. Багатофакторні плани експериментів будують з урахуванням сумісного впливу факторів на відгук моделі.
 - 3.3. Якщо приймається, що фактор може неперервно змінюватись в заданому інтервалі, то цей фактор кількісний.
 - 3.4. Якщо приймається, що фактор має декілька дискретних значень, то цей фактор якісний.
 - 3.5. Для якісної оцінки впливу факторів використовують дисперсійний аналіз, а для кількісної оцінки впливу факторів - регресійний аналіз.

(Моделювання Систем Посібник 231) Регресійний аналіз впливу факторів.

(Моделювання Систем Посібник 237) Дисперсійний аналіз впливу факторів.

33. Еволюційні методи оптимізації імітаційних параметрів

(Моделювання Систем Посібник 260)

- Елементом популяції є набір параметрів, пошук яких здійснюється: $A = (r_1, g_1, r_2, g_2, \dots, r_n, g_n)$.
- Початкова популяція (генерування 0) формується з випадкових значень, розкиданих в області допустимих значень параметрів.
- Кожний елемент популяції запускається у «життя», тобто в імітаційну модель. Результатом такої життєдіяльності елемента популяції є відгук моделі. Набори параметрів, які виявились «неспроможними», тобто дістали в процесі імітації великі значення відгуку моделі, «гинуть» або знищуються. Таким чином за значенням відгуку моделі здійснюється відбір елементів популяції.
- Елементи популяції, що пройшли відбір, допускаються до схрещування. Схрещування здійснюється для випадково обраних пар елементів популяції склеюванням частин наборів параметрів. Нехай для схрещування обрані елементи популяції A_j та A_k . В результаті роботи оператора кросовера випадковим чином обираються компоненти, параметри яких в елементі-нащадку будуть прийняті такими, як в елементі A_j , інші компоненти елемента-нащадка приймають значення параметрів такі, як в елементі A_k .
- Мутація здійснюється додаванням випадкового відхилення до результату, який отриманий в результаті схрещування, наприклад, додаванням з рівною ймовірністю -1, 0 або 1.

- Кожна наступна популяція (генерування j) формується з елементів, що пройшли відбір на попередньому генеруванні (генерування j-1), та з елементів, що створені в результаті схрещування та мутації.
- У правилі зупинки еволюційного пошуку користувач задає точність визначення оптимального значення (перехід до наступної популяції не суттєво поліпшує оптимальне значення) та максимальну кількість генерувань.

34. Програмне забезпечення з імітаційного моделювання

(Лекція 1. Поняття моделі. Методи моделювання 2022 11)

- Неперервні моделі: Matlab, Simulink
- Дискретно-подійні моделі: VisSim, CPNTools, AnyLogic,
- ProModel, Simio <https://www.simio.com/index.php>
- Arena Simulation Software <https://www.arenasimulation.com/>

-
- GPSS
 - *(Моделювання Систем Посібник 265)*
 - *(Лекція 19 GPSS 1)*
 - PTRSIM
 - *(Моделювання Систем Посібник 273)*
 - Arena
 - *(Моделювання Систем Посібник 280)*

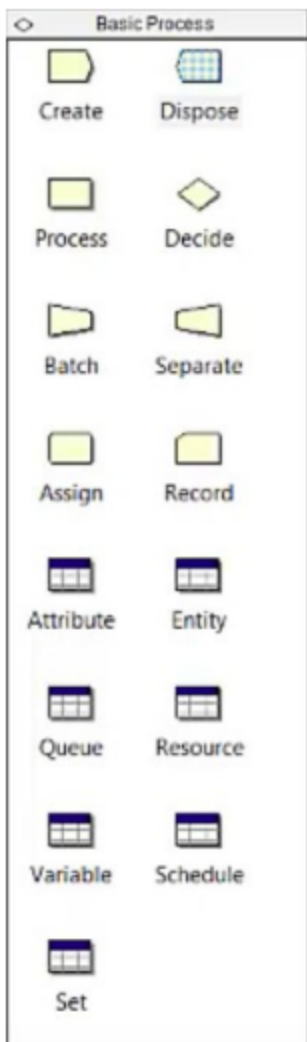
35. Моделювання в Arena Professional Software

(Лекція 18 Arena)

Конструктивні елементи Arena

- Розділені на групи Basic Process, Advanced Process та інші
- Кожна група містить набір модулів для побудови моделі. Модулі перетягуються у вікно робочого поля та з'єднуються у відповідності до логіки процесу, який моделюється
- Кожен блок містить набір параметрів, які має налаштувати користувач у вікні властивостей модул

Basic Process



Create-Модуль Create є відправною точкою для вимоги в імітаційній моделі. Вимоги - це об'єкти, які обробляються в системі. Створення вимоги модулем відбувається за розкладом, або ж ґрунтуючись на значенні часу між прибуттям вимог в модель. Залишаючи модуль, вимога починає оброблятися в системі. Тип створюваної вимоги визначається в цьому модулі.

Process-Модуль Process є основним модулем процесу обробки в імітаційній моделі. Окрім стандартного модуля Process, можна використовувати підмодель, додаючи їй особливу, визначену користувачем, ієрархічну логічну схему. У модулі можна також задавати додаткові вартісні і часові характеристики процесу обробки об'єкта. П

Decide-Модуль Decide дозволяє враховувати прийняття рішень в моделі. Модуль включає опції прийняття рішень заснованих на умові By Condition (наприклад, якщо тип вимоги Car) або заснованих на вірогідності By Chance (наприклад, 75% - true, а 25% - false). Умови можуть бути засновані на значенні атрибуту Attribute, значенні змінної Variable, типі об'єкта

Entity Type або засновані на виразі Expression. Якщо поставлена умова не виконується, то об'єкт залишатиме модуль через гілку False.

Batch-Модуль Batch відповідає за механізм угруповання в імітаційній моделі. Наприклад, зібрати необхідну кількість даних, перш ніж починати їх обробку, зібрати раніше розділені копії одної форми, з'єднати пацієнта і його лікарняну карту прийому до лікаря. Угруповання може бути постійним або тимчасовим. Тимчасово згруповані комплекти пізніше можуть бути роз'єднані за допомогою модуля Separate. Комплекти можуть складатися з будь-якої кількості вхідних об'єктів, визначеної користувачем, або ж об'єкт може об'єднуватися в комплект залежно від його атрибуту. Часові і вартісні характеристики об'єктів, які входять у групу, будуть рівні сумі характеристик вимог, які увійшли до групи

Separate-Модуль Separate може використовуватися як для створення копій вхідної вимоги, так і для розділення раніше згрупованих вимог. Наприклад, для роз'єднання раніше згрупованих комплектів документів, для паралельної обробки рахунків і документів по одному замовленню. Правило для розділення вартісних і часових характеристик копій вимоги і розділеної вимоги визначається користувачем. Коли тимгодини згрупована вимога прибуває в модуль, вони розкладаються на складену вимогу. вимога покидає модуль в тій же послідовності, в якій вони додавалися в комплект. Якщо модуль створює копії вимоги, то користувач може задати кількість дублікатів вимоги. У дубльованій вимоги значення атрибуту, а також анімаційна картинка такі ж, як і оригіналу. Оригінальна вимога також покидає модуль.

Assign-Модуль Assign призначений для завдання нового значення змінній, атрибуту вимоги типу вимоги, анімаційній картинці вимоги або іншої змінної в системі. Наприклад, для встановлення пріоритету для клієнтів, для привласнення номера наказу, що вийшов. У одному модулі можна зробити тільки одне призначення.

Record-Модуль Record призначений для збору статистики в імітаційній моделі. Наприклад, для підрахування, яка кількість замовлень була виконана із запізненням, підрахування кількості роботи, що здійснюється за одну годину. Модуль може збирати різні типи статистики, включаючи час між виходами вимоги з модуля, статистику вимоги (час циклу, вартість), статистику за період часу (період часу від заданої точки до теперішнього моменту). Також доступний кількісний тип статистики.

Dispose-Модуль Dispose є вихідною точкою з імітаційної моделі. Наприклад, клієнти покидають відділ, закінчення бізнес-процесу. Статистика про вимогу може збиратися до того моменту поки вона не вийде з системи.

36. Побудова ієрархічних моделей в Arena

(Джерело ????)

Ієрархічне моделювання в **Arena** — це підхід, який дозволяє розділяти складну систему на підсистеми, організовані в ієрархічній структурі.

Основні концепти побудови ієрархічних моделей в Arena:

1. Субмоделі (Submodels):

- a. Субмоделі в Arena — це частини загальної моделі, які можна виділити в окремі блоки.
 - b. Вони дозволяють спростити структуру складної моделі, групуючи елементи, що відповідають за одну функціональну частину системи.
 - c. **Приклад:** У моделі виробничого процесу можна виділити окремі субмоделі для закупівлі матеріалів, виробництва та відвантаження продукції.
- 2. Використання блоків (Modules):
 - a. Arena надає можливість створювати ієрархії через використання модулів у вікні **Project Navigator**.
 - b. Основні модулі, які можуть використовуватись для побудови ієрархічних моделей:
 - i. **Basic**
 - ii. **Advanced Process:** Для більш складних процесів і логіки.
 - iii. **Transfer:** Організація руху матеріалів.
 - iv. **Submodel:** Створення ієрархічних блоків для вкладених моделей.
- 3. Створення субмоделей (Submodels):
 - a. Кроки створення:
 - i. Виберіть групу елементів моделі, які логічно можна об'єднати в одну підсистему.
 - ii. У меню **Submodel** створіть новий модуль.
 - iii. Перенесіть вибрані елементи у вікно нової субмоделі.
 - iv. Додайте необхідні вхідні та вихідні точки (Input/Output Connections).
 - b. Субмоделі можна використовувати багаторазово в різних частинах моделі.

37. Елементи анімації програмного забезпечення Arena

(Джерело????)

Entity-Представляє об'єкт, який рухається через модель (наприклад, клієнт, товар, машина). Анімація показує переміщення сутності між блоками або затримку в черзі.

Queue-ідображає накопичення сутностей, які чекають обслуговування. Анімація показує, скільки сутностей перебуває в черзі, а також порядок їх надходження.

Resource-Відображає доступність або зайнятість ресурсів (наприклад, машини, персоналу). Анімація змінює стан ресурсу (вільний/зайнятий) і може показувати час обслуговування.

Transporter-Візуалізує транспортний засіб, який переміщує сутності між локаціями. Анімація показує рух транспортера, його навантаження та розвантаження.

Path-Відображає маршрути, якими рухаються сутності чи транспортери. Анімація показує переміщення сутностей або транспортерів вздовж заданих шляхів.

Station-Відображає місця, де сутності можуть зупинятися для виконання певних дій (обслуговування, сортування). Анімація показує прибуття й відправлення сутностей зі станції.

Timer-Відображає плин часу в моделі симуляції. Анімація показує поточний симуляційний час або реальний час, який пройшов з початку моделювання. Це допомагає відстежувати тривалість виконання процесів і співвідносити її з подіями у моделі.

Date-Відображає поточну симуляційну дату (або реальну, якщо вказано). Анімація оновлює значення дати залежно від плину часу в моделі, дозволяючи враховувати часові періоди для подій, які мають прив'язку до календаря (наприклад, робочі зміни, графік виконання завдань).

Plot-Відображає зміну змінних або параметрів системи у вигляді графіків. Анімація показує динаміку параметрів моделі (наприклад, довжина черги, використання ресурсу).

(Моделювання Систем Посібник 291)

Анімація об'єкта обслуговування виконується за допомогою параметру Entity Picture, що задається в блоці Entity. Анімаційну картинку об'єкта обслуговування можна вибрати із переліку запропонованих картинок або із бібліотеки анімаційних картинок. При запуску моделі користувач спостерігатиме рух об'єктів обслуговування від одного блоку моделі до іншого.

Анімація ресурсу виконується за допомогою панелі Resours. У вікні, що з'явиться після виклику панелі Resours, потрібно вибрати ресурс із списку ресурсів моделі. Потім обрати картинку для зображення стану «вільний» у списку картинок і перемістити її до вікна Idle. Так само обрати картинку для зображення стану «зайнятий» і перемістити її до вікна Busy. Якщо бажаної картини в запропонованому списку не знайдено, можна скористатись бібліотеками картинок, які містяться у папці C:\Program Files\Rockwell Software\Arena у файлах з розширенням.plb. Після вибору усіх картинок ресурсу як зайнятого, так і вільного, анімаційна картинка розташовується курсором миші у потрібному користувачу місці. Розтягуванням анімаційної картини користувач може змінювати її розміри.

Анімація черги виконується за допомогою панелі Queues. Після заповнення діалогового вікна картинка розміщується користувачем на моделі.

Візуальне зображення гістограми виконується за допомогою панелі Histograms. Зображення гістограми сильно залежить від масштабу, який обрав користувач, тому вибирати його потрібно ретельно.

Візуальне зображення значення змінної виконується за допомогою панелі Variables. Скористатись цією візуалізацією можна тільки якщо у моделі була введена змінна типу Variable.

Візуальне графічне зображення динаміки змінювання виконується за допомогою панелі Plot. Може бути виведена, наприклад, динаміка змінювання кількості об'єктів у черзі, динаміка змінювання часу обслуговування об'єктів у моделі і т.і.

38. Моделювання в CPNTools

(“Моделирование телекоммуникационных систем в CPN Tools”)

Ось тут детальна інформація, але російською. Вона досить коротка і гарно викладена:

<http://daze.ho.ua/cpnmp-ru.pdf>

(Джерело???)

CPN Tools – це програмне забезпечення для моделювання та аналізу систем, яке базується на кольорових мережах Петрі (Colored Petri Nets). Воно розроблене в Університеті Ольборга для моделювання складних систем із паралелізмом і взаємодією компонентів.

Ключові аспекти:

- **Кольорові мережі Петрі:** розширення класичних мереж Петрі, де маркери мають типи (кольори), що дозволяє описувати складні дані.
- **Графічний інтерфейс:** моделі створюються інтерактивно за допомогою інструментів редагування графів.
- **Модульний підхід:** модель можна розбивати на підмоделі для зручності розробки й аналізу.

Основні функції:

- **Моделювання:** візуальне представлення процесів у вигляді графа для опису поведінки систем.
- **Симуляція:** динамічний аналіз поведінки моделі для виявлення помилок або оптимізації процесів.
- **Аналіз:** підтримує перевірку властивостей моделі, таких як досяжність станів, коректність та відсутність блокувань.

Особливості:

Підтримка **інтерактивної симуляції**, що дозволяє досліджувати поведінку моделі в реальному часі.

Використання мови ML для визначення функцій і типів даних у моделі.

Зручне середовище для роботи з великими та складними моделями.

39. Моделювання в GPSS

(Моделювання Систем Посібник 265, Лекція 19 GPSS 1)

GPSS (General Purpose Simulation System) – це система моделювання складних об'єктів загального призначення, розроблена Джефрі Гордоном у 1960 році. Спочатку підтримувалася компанією IBM.

Основні поняття:

- **Транзакти** – вимоги, що обслуговуються в мережі масового обслуговування.
- **Оператори** – команди, що описують процеси виникнення, затримки, обробки та виходу транзактів.
- **Структура оператора:** мітка, назва, змінні (наприклад, **ADVANCE 20, 5** для затримки).

Основні оператори:

- **GENERATE:** створює транзакти з заданими інтервалами часу.
- **TERMINATE:** видаляє транзакти з системи.
- **QUEUE/DEPART:** реєструє збільшення/зменшення довжини черги.
- **SEIZE/RELEASE:** зайняття або звільнення пристрою транзактом.
- **ADVANCE:** затримка транзакта на визначений час.
- **TRANSFER:** умовний або безумовний перехід до іншого оператора.

Особливості: GPSS підтримує моделювання складних систем, таких як черги, ресурси, маршрутизація та статистика, використовуючи мову, близьку до низькорівневої.

40. Загальні складові сучасного програмного забезпечення з імітаційного моделювання систем

(Моделювання Систем Посібник 265)

40.1. GPSS (General Purpose Simulation System)

(Моделювання Систем Посібник 268)

GPSS—це одна з найвідоміших мов імітаційного моделювання, розроблена Джеффрі Гордоном приблизно у 1960 році та спочатку підтримувана компанією IBM. Вона призначена для моделювання складних дискретних систем, зокрема мереж масового обслуговування. Сучасна версія GPSS World розроблена компанією Minuteman Software і підтримує об'єктно-орієнтоване моделювання.

GPSS дозволяє моделювати системи, що містять черги, пристрої, генератори випадкових подій, а також розширену систему команд для маніпуляції з транзактами (елементами, що проходять через модель). Основні оператори GPSS включають:

- GENERATE—створює нові транзакти (елементи системи).
- QUEUE та DEPART—використовуються для обліку довжини черг.
- SEIZE та RELEASE—керують зайняттям і звільненням ресурсів.
- ADVANCE—задає час затримки транзакту.
- TRANSFER—переміщує транзакти між етапами моделювання.

GPSS підтримує анімацію процесів і аналіз даних. Вона оснащена функціями статистичного аналізу, такими як розрахунок довірчих інтервалів та дисперсійний аналіз. Система підтримує автоматичне виконання оптимізаційних експериментів, що дозволяє досліджувати поведінку складних систем.

40.2. PTRSIM (Petri Net Simulation System)

(Моделювання Систем Посібник 274)

PTRSIM—це система імітаційного моделювання, створена в 2006 році на кафедрі комп'ютерних технологій Черкаського державного технологічного університету. Вона базується на концепції мереж Петрі, що є універсальним засобом формалізації дискретно-подійних процесів. Основна ідея системи—автоматизація імітаційного моделювання процесів, які неможливо ефективно змоделювати в традиційних середовищах (GPSS, Arena).

Процес моделювання в PTRSIM включає такі етапи:

1. Побудова формалізованої моделі засобами мереж Петрі.
2. Введення моделі в імітаційну систему.
3. Перевірка моделі шляхом запуску візуалізації.
4. Виконання моделювання при різних параметрах.
5. Аналіз результатів та побудова графіків.
6. Формулювання висновків для оптимізації системи.

PTRSIM підтримує мережі Петрі з часовими затримками, інформаційними зв'язками, випадковими подіями та умовними переходами. Унікальні можливості системи включають гнучке управління зв'язками між об'єктами, що дозволяє точно відображати складні процеси, наприклад паралельну обробку даних, системи управління запасами, логістичні процеси.

Графічний інтерфейс PTRSIM дозволяє користувачам зручно створювати моделі, змінювати їх, анімаційно відстежувати перебіг імітації. Вбудовані інструменти аналізу забезпечують обчислення середнього завантаження системи, статистичних характеристик черг та ресурсів, а також побудову графіків зміни параметрів у часі.

40.3. Arena—сучасний пакет імітаційного моделювання

(Моделювання Систем Посібник 280)

Arena—це потужна комерційна система імітаційного моделювання, створена в 1998 році компанією Systems Modeling Corporation. Вона базується на мові моделювання SIMAN і використовує Cinema Animation для візуалізації процесів. Arena дозволяє моделювати широкий спектр систем: виробничі лінії, складську логістику, банківські системи, транспортні процеси та обслуговування клієнтів.

Інтерфейс Arena включає три основні області:

- Робоче поле, де будуються моделі.
- Параметри модулів, що дозволяють налаштовувати процеси.
- Панель проекту, що містить основні інструменти для управління моделями.

Основні компоненти моделювання:

- Create—створення об'єктів (клієнтів, деталей, завдань).
- Process—процес обробки, використання ресурсів.
- Decide—прийняття рішень (розгалуження логіки).
- Batch—угруповання об'єктів.
- Assign—призначення атрибутів.
- Dispose—вихід об'єктів із системи.

Arena підтримує детальну анімацію та графічний аналіз, що дозволяє користувачам візуально відстежувати роботу системи. Вона включає засоби оптимізації та статистичні інструменти (Input Analyzer, Output Analyzer), які допомагають налаштовувати параметри моделі для досягнення найкращих результатів.

Особливості Arena:

- Використання графічного інтерфейсу для моделювання.
- Автоматична генерація коду мовою SIMAN.
- Вбудована підтримка розподілених систем та мультипроцесорного моделювання.
- Гнучка система звітування, що дозволяє аналізувати продуктивність моделі, завантаженість ресурсів, час очікування в чергах.

Arena широко застосовується у виробництві, логістиці, фінансах, обслуговуванні клієнтів та інших сферах, де необхідне моделювання бізнес-процесів і оптимізація робочих навантажень.

40.4. Порівняння GPSS, PTRSIM та Arena

- GPSS—традиційна мова для дискретного моделювання, особливо ефективна для аналізу мереж масового обслуговування, виробничих ліній та черг. Вимагає знання програмування, має потужні статистичні засоби.
- PTRSIM—спеціалізована система, що базується на мережах Петрі, відмінно підходить для моделювання процесів управління, багатопотокових систем, технологічних процесів. Має зручний візуальний редактор і розширені можливості для паралельного моделювання.
- Arena—універсальне програмне середовище, що використовується в різних сферах бізнесу, виробництва, транспорту, фінансів. Має розвинений графічний інтерфейс, підтримку статистичного аналізу та оптимізацію моделей.

Таким чином, GPSS і PTRSIM більше підходять для інженерних та дослідницьких завдань, тоді як Arena є гнучким комерційним рішенням для бізнесу, логістики та виробництва.

41. Паралельні обчислення в імітаційному моделюванні

41.1. Прискорення експериментальних досліджень

(Лекція 17 Паралельний алгоритм імітації 2)

- кілька прогонів однакової моделі виконують одночасно для отримання більш точного результату (див. Тактичне планування експериментів):
 - прогони виконуються в окремих потоках, основний метод очікує завершення роботи всіх потоків, потім виконує обчислення середнього значення
- кілька моделей з різними параметрами виконують одночасно (див. стратегічне планування експериментів, оптимізація еволюційними методами):
 - основний метод здійснює обчислення параметрів, при яких виконується моделювання, та обробку результатів імітації

- імітація моделей для різних наборів параметрів здійснюється в окремих потоках

41.2. Графічний інтерфейс

(Лекція 17 Паралельний алгоритм імітації 3)

- Динамічне відтворення результатів моделювання одночасно з обчисленням моделі:
 - в одному потоці відбувається імітація, інший з затримкою виводить поточні значення вихідних характеристик
- Анімація моделювання одночасно з обчисленням моделі:
 - перемальовування елементів моделі виконується з затримкою

41.3. Прискорення виконання алгоритму імітації

(Лекція 17 Паралельний алгоритм імітації 4)

Варіанти розпаралелювання:

- Відшукати незалежні частини моделі, що можна виконувати одночасно
 - Проте виявлення таких частин є самостійною складною задачею
- Виконувати розпаралелювання в межах операцій, що часто повторюються. Наприклад, для мережі Петрі можна виконувати одночасно перевірку запуску переходів та вхід маркерів в переходи
 - Накладні витрати на створення потоків (або задач) перевищують виграш в часі, отриманий за рахунок одночасного виконання задач
- Виконувати імітацію окремих фрагментів моделі одночасно
 - Через використання спільної змінної часу алгоритм тільки сповільнюється

(Лекція 17 Паралельний алгоритм імітації 6)

- Виконувати імітацію окремих фрагментів моделі одночасно не використовуючи спільну змінну часу

42. Паралельний алгоритм імітації Петрі-об'єктної моделі

42.1. Варіант розпаралелювання Петрі-об'єктної моделі

- Фрагменти моделі – це Петрі-об'єкти, з яких вона складається
- Кожний Петрі-об'єкт відтворює своє функціонування з урахуванням подій в Петрі-об'єктах, з якими він пов'язаний.
- Інформація про події надходить від об'єктів і розглядається як зовнішня

42.2. Паралельний алгоритм імітації Петрі-об'єктної моделі

(Лекція 17 Паралельний алгоритм імітації 7)

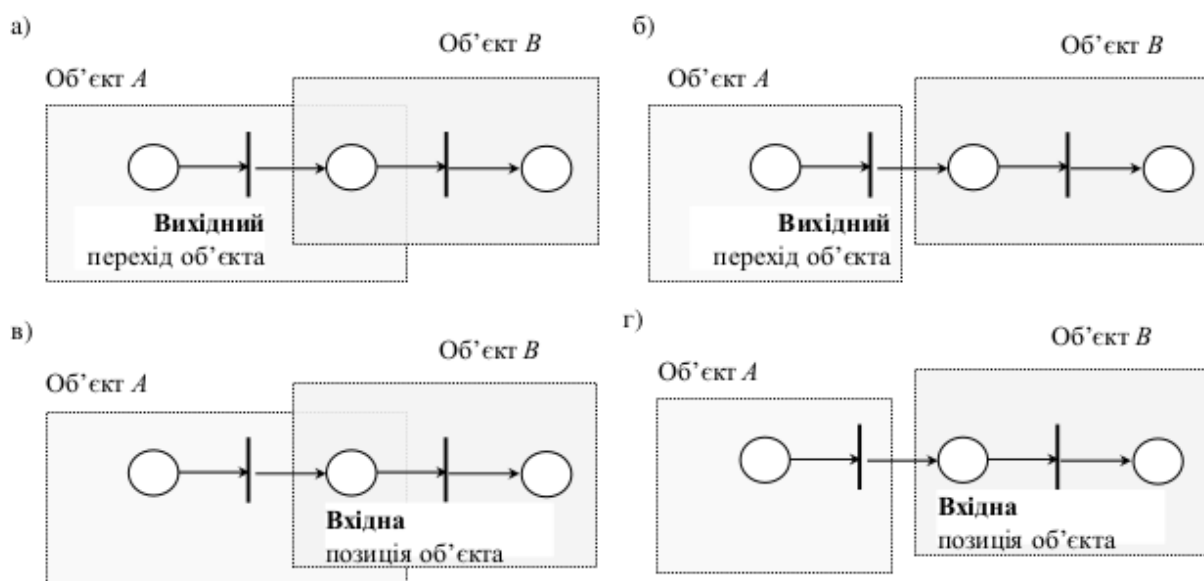
Означення:

- Перехід мережі Петрі-об'єкта є вихідним, якщо при його запуску здійснюється вихід у спільну позицію з іншим об'єктом або здійснюється вихід у позицію іншого об'єкта (рис.1 а,б).

- Позиція об'єкта є вхідною, якщо вона є спільною з позицією іншого об'єкта або у неї здійснюється вихід з переходу іншого об'єкта (рис.1 в,г).
- Якщо об'єкт А має вихідний перехід, з якого здійснюється вихід маркерів у позицію об'єкта В, то об'єкт В є next-об'єктом для об'єкта А
- Якщо об'єкт В має позицію, в яку здійснюється вихід маркерів з вихідного переходу об'єкта А, то об'єкт А є previous-об'єктом для об'єкта В.
- Безпечним інтервалом імітації об'єкта є інтервал часу між двома послідовними виходами маркерів з вихідного переходу його previous-об'єкта.

42.3. Взаємодія Петрі-об'єктів

(Лекція 17 Паралельний алгоритм імітації 8)



- а) вихід маркерів з переходу об'єкта у спільну позицію його next-об'єкта;
 б) вихід маркерів з переходу об'єкта у позицію його next-об'єкта;
 в) вхід маркерів у спільну позицію об'єкта з його previous-об'єкта;
 г) вхід маркерів у позицію об'єкта з переходу його previous-об'єкта.

42.4. Правила паралельного функціонування Петрі-об'єктів

(Лекція 17 Паралельний алгоритм імітації 9)

- Кожний Петрі-об'єкт здійснює імітацію в локальному часі в окремому потоці.
- Кожний об'єкт, який має next-об'єкт, передає йому інформацію про моменти виходу маркерів з об'єкта і призупиняє своє функціонування при значному накопиченні такої інформації.
- Кожен об'єкт, який має previous-об'єкт, здійснює імітацію в межах до наступної події входу в нього маркерів з previous-об'єкта, поступово просуваючи свій локальний

час до повного вичерпання накопиченої інформації про вхідні події, або очікує надходження інформації про вхідні події зі свого previous-об'єкта.

- Об'єкт, який має previous-об'єкт, при досягненні моменту часу входу маркерів в об'єкт, відновлює вихід маркерів у спільну позицію з переходу previous-об'єкта та продовжує імітацію.
- Об'єкт, який має next-об'єкт, при досягненні моменту виходу маркерів з об'єкта, призупиняє вихід маркерів у позицію next-об'єкта. Цей вихід відновить next-об'єкт у відповідний момент часу свого функціонування.
- Об'єкт, який вичерпав час моделювання, передає повідомлення про це next-об'єкту, якщо такий є, і завершує свою роботу. Разом з завершенням роботи об'єкта завершує роботу і потік, ним генерований.
- Об'єкт, який отримав від previous-об'єкта сигнал про завершення його роботи і, водночас, вичерпав усі накопичені події, припиняє свою роботу.

42.5. Правила паралельного функціонування Петрі-об'єктів

(Лекція 17 Паралельний алгоритм імітації 10)

- зберігає моменти часу надходження маркерів з інших об'єктів, що не опрацьовані на поточний момент часу;
- в межах від одного моменту зовнішньої події до наступної гарантовано не виникає зовнішніх подій, отже, об'єкт може виконувати імітацію своїх внутрішніх подій;
- порожній стан буфера означає, що наступний момент зовнішньої події невідомий (на поточний момент виконання алгоритму імітації);
- стан буфера, в якому останній його елемент є нескінченність (максимально допустиме число), означає, що previous-об'єкт завершив свою роботу і надходження зовнішніх подій не очікується.

42.6. Локальний час об'єкта просувається за такими правилами

(Лекція 17 Паралельний алгоритм імітації 11)

Локальний час об'єкта просувається за такими правилами:

- якщо час найближчої події менший за межу безпечного інтервалу, то просунути локальний час у момент найближчої події, виконати (внутрішні) події, для яких час збігається з поточним моментом, та продовжити імітацію об'єкта;
- інакше
 - якщо межа безпечного інтервалу менша за межу інтервалу моделювання
 - просунути локальний час на межу безпечного інтервалу та виконати зовнішню подію (відновлення маркерів у вхідних позиціях),
 - інакше просунути локальний час на межу безпечного інтервалу та
 - завершити імітацію об'єкта

42.7. Межа безпечного інтервалу

(Лекція 17 Паралельний алгоритм імітації 12)

Межа безпечного інтервалу визначається за наступними правилами:

- якщо немає previous-об'єкта, то межа встановлюється в час моделювання;
- інакше
 - якщо буфер зовнішніх подій не порожній і перша подія менша за час моделювання, то межа встановлюється в момент першої зовнішньої події;
 - інакше - в час моделювання.

42.8. Об'єкт

(Лекція 17 Паралельний алгоритм імітації 13)

Об'єкт знаходиться в очікуванні (призупиняє своє функціонування), якщо і тільки якщо:

- він знаходиться у stop-стані: кількість маркерів недостатня для запуску переходів і момент виходу маркерів з переходу більший за границю безпечного інтервалу і, якщо є previous-об'єкт, перша подія в буфері менша за границю безпечного інтервалу;
- є previous-об'єкт і буфер зовнішніх подій порожній;
- є next-об'єкт і його буфер зовнішніх подій перевищує заданий ліміт.

Об'єкт розблоковується (припиняє очікування) якщо і тільки якщо:

- його previous-об'єкт здійснив вихід у позицію, спільну з даним об'єктом;
- його next-об'єкт подав сигнал про відновлення роботи об'єкта у разі, якщо він вичерпав буфер подій до заданого ліміту.

Об'єкт завершує імітацію, якщо досягнутий кінець інтервалу моделювання.

42.9. Синхронізація виходу маркерів у спільну позицію

(Лекція 17 Паралельний алгоритм імітації 14)

Синхронізація виходу маркерів у спільну позицію відбувається таким чином:

- у момент здійснення виходу маркерів з вихідного переходу Петрі-об'єкта вихід маркерів у позицію next-об'єкта не здійснюється, замість цього поточний момент часу запам'ятовується в буфері зовнішніх подій next-об'єкта;
- коли локальний час об'єкта досягає першого значення буфера подій, здійснюється вхід маркерів у вхідну позицію об'єкта, а перше значення з буфера зовнішніх подій видаляється.

42.10. Умова завершення роботи потоків

(Лекція 17 Паралельний алгоритм імітації 15)

Якщо в буфері об'єкта перший елемент нескінченність, це означає, що його previous-об'єкт завершив імітацію на інтервалі імітації і зовнішніх подій більше не очікується.

Об'єкт продовжує імітацію до завершення інтервалу моделювання без очікування зовнішніх подій.

42.11. Узагальнення

(Лекція 17 Паралельний алгоритм імітації 16)

Алгоритм, описаний вище, виходить з таких припущень: модель має хоча б один об'єкт, для якого не заданий previous-об'єкт, і хоча б один об'єкт, для якого не заданий next-об'єкт. В реальних умовах це відповідає відкритій системі.

- Якщо об'єкт має декілька previous-об'єктів, то для кожного з них утворюється свій буфер подій. Момент найближчої події можна визначити тільки, коли надійшла інформація про наступні події від усіх previous-об'єктів. Тому робота такого об'єкта призупиняється, якщо хоч один буфер подій порожній, і відновлюється, якщо усі буфери подій не порожні. Щоб запобігти взаємному блокуванню об'єктів, потрібно відслідковувати стан, в якому об'єкт і його previous-об'єкти одночасно знаходяться в стані очікування (коли хоч один з їх буферів подій є порожнім). В цьому випадку потрібно визначити найближчу подію для об'єкта і його previous-об'єктів, просунути локальний час в момент найближчої події і продовжити моделювання.
- Якщо об'єкт має декілька next-об'єктів, то робота такого об'єкта припиняється, якщо досягає ліміту буфер подій хоч в одному з його next-об'єктів, і відновлюється, якщо в усіх next-об'єктах кількість подій в буфері подій менша за назначений ліміт.

42.12. Взаємодія потоків паралельного алгоритму імітації Петрі-об'єктної моделі

(Лекція 17 Паралельний алгоритм імітації 17)

Потоки А, В, С ілюструють дію потоків, що запускають імітацію Петрі-об'єктів А, В, С таких, що:

- А є previous-об'єктом для В,
- В є next-об'єктом для А і previous-об'єктом для С,
- С є next-об'єктом для В.

Пара повідомлень <<new event>> та <<empty>> інформує об'єкт про надходження нової події від previous-об'єкта та, навпаки, що таких подій немає. Повідомлення <<no event>> сповіщає next-об'єкт про завершення імітації об'єктом, а значить про припинення надходження подій від previous-об'єкта.

Пара повідомлень <<limit true>>, <<limit false>> інформує об'єкт про перевищення ліміту буферу зовнішніх подій його next-об'єкту або, навпаки, про те, що такого перевищення немає.