Санкт-Петербургский государственный университет
Кафедра системного программирования

Сети Петри и их использование для верификации программ

Дмитрий Владимирович Павлушкин, Александра Юрьевна Лановая, Александр Алексеевич Лекомцев

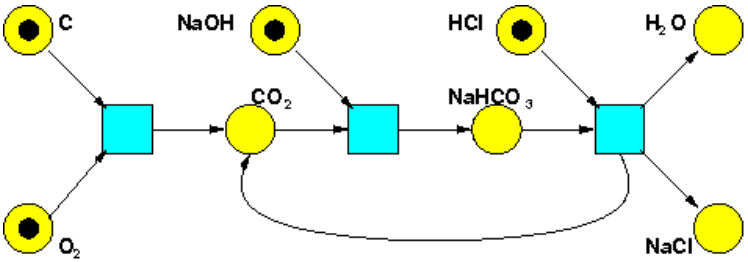
Математико-механический факультет, Программная инженерия, 1 курс

06.04.2023

Содержание доклада

- Краткая биография Карла Петри, с чего все начиналось
- Общая терминология
- Математическая модель
- Использование в программах для верификации

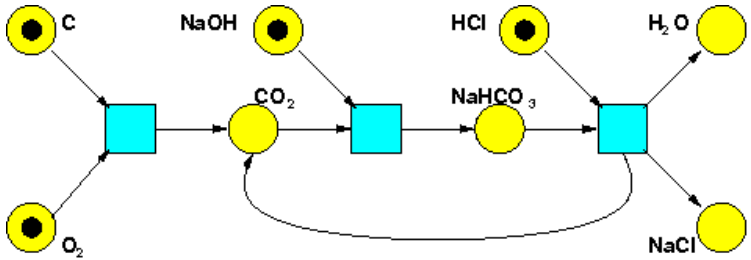
Химический пример



Термины:

- Кружочки - атомы или хим. соединение
- Черные точки - метки перехода
- Прямоугольники - хим. реакции.
- Стрелочки - направление перехода

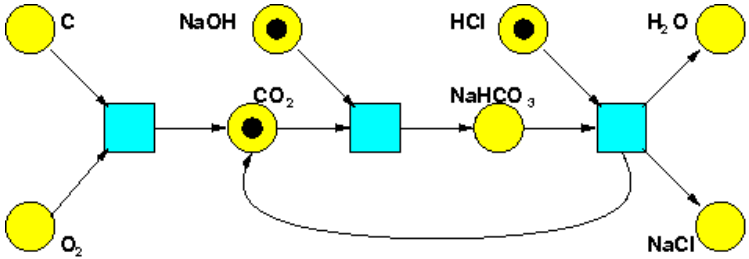
Химический пример



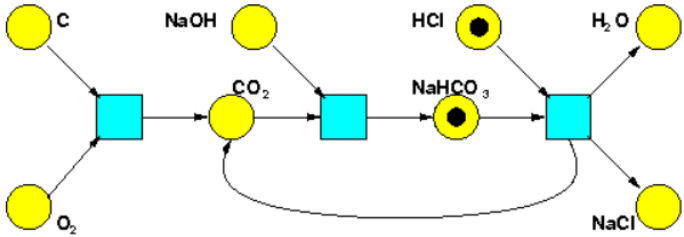
Правила:

- Пока не “соберутся” все маркеры перехода, ведущие к хим. реакции, она не запустится
- Как только они “собираются”, происходит хим. реакция, которая забирает у каждой молекулы по одной метке перехода и “отдает” по одной другому соединению. Или не одной, это зависит от того, какое число написано над стрелочкой. По умолчанию оно равно 1

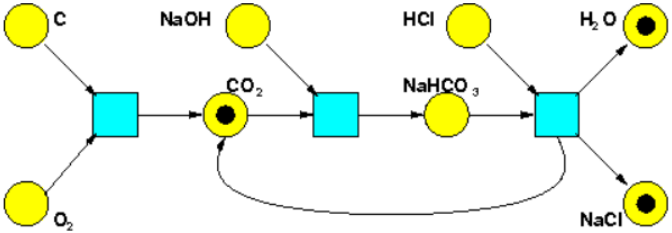
Химический пример



Химический пример



Химический пример

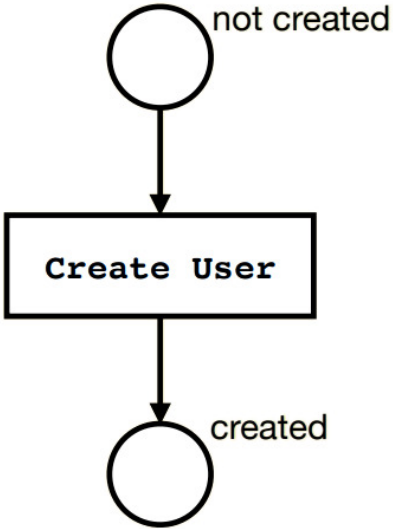


Биография Карла Петри

Карл Петри (1926-2010)

- 1939г. — Первое появление концепции "сетей Петри"
- 1941г. — Знакомится с трудами Конрада Цузе и начинает собирать собственный аналоговый компьютер
- 2008г. — Награда Компьютерного общества IEEE «Пионер компьютерной техники» (Computer Pioneer Award)

Общая терминология



— Позиции (places) хранят метки
— Дуги связывают позиции и переходы
— Переходы (transitions) перемещают метки

Общая терминология

- Переходы могут быть связаны только с позициями (и наоборот)
- Переход может **сработать** когда в каждой входной позиции есть по меньшей мере одна метка
- **Срабатывание** перехода забирает по одной метке из каждой входной позиции и добавляет по одной метке в каждую выходную позицию

Способы задания сетей Петри

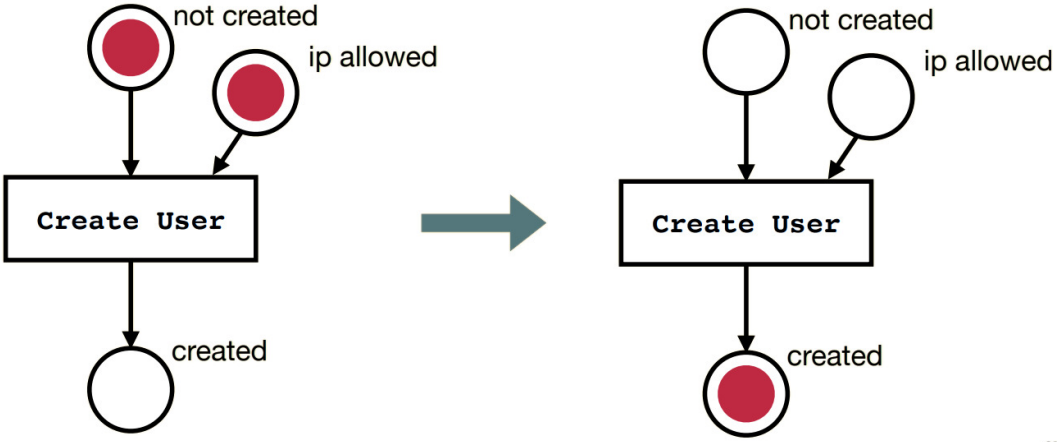
- Аналитический

  Перечисление элементов множеств позиций, переходов, перечисление элементов комплектов входной и выходной функций
- Графический

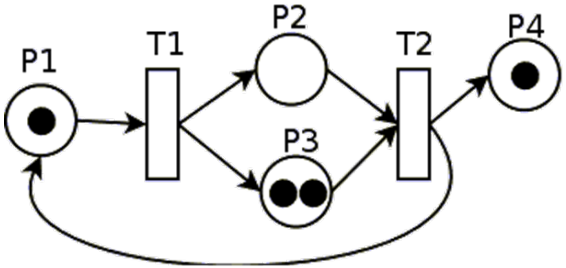
  Представляет собой двудольный, ориентированный мультиграф
- Матричный

  Матрица имеет n столбцов (по числу вершин мест) и k строк (по числу вершин переходов). Элементами являются нули и единицы. Единица - если имеется дуга от вершины места к вершине перехода, иначе ноль

Пример


not created
ip allowed
Create User
created
not created
ip allowed
Create User
created

Более строгое определение

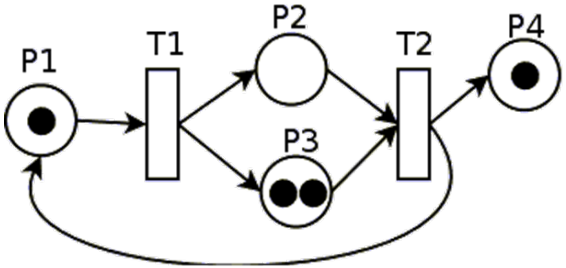


Сеть Петри - это двудольный ориентированный мультиграф A=(P,T,I,O), где:

- **P** – множество состояний;
- **T** – множество переходов;
- **I** – множество (картеж) входных дуг;
- **O** – множество (картеж) выходных дуг;

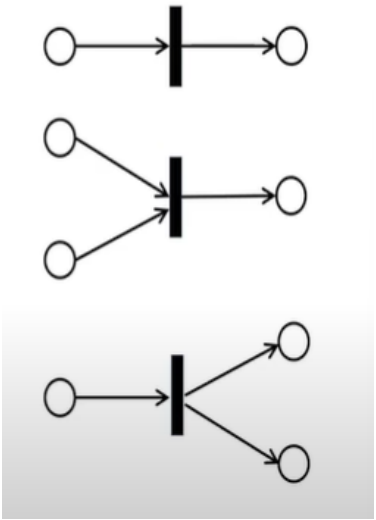
Маркировка СП – расстановка фишек в состояниях СП. Это вектор неотрицательных целых чисел, размерность которого равна числу состояний СП.

Более строгое определение



- Состояния (P): P=$\{P_1, P_2, P_3, P_4\}$
- Переходы (T): T=$\{t_1, t_2\}$
- Маркировка (μ): μ=$(1,0,2,1)^T$
- Входы (I): I(t_1)=$\{P_1\}$; I(t_2)=$\{P_3, P_2\}$
- Выходы (O): O(t_1)=$\{P_2, P_3\}$; O(t_2)=$\{P_1, P_4\}$

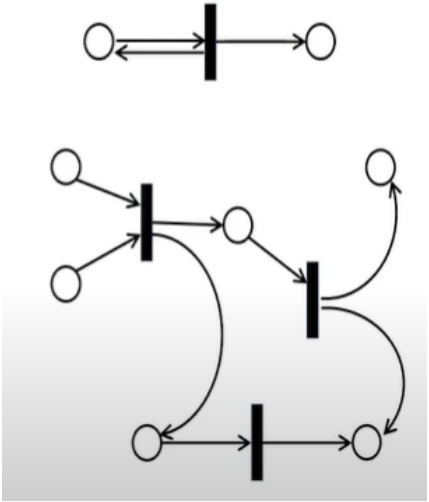
Выполнение сети Петри



Срабатывание перехода t - это удаление по одной метке из позиций p_i, из которых ведут рёбра в t и добавление метки в каждую позицию p_j, в которую ведут ориентированные рёбра из t

Переход t разрешен, если каждая позиция p_i, из которых ведут рёбра в t, содержит метку

Выполнение сети Петри



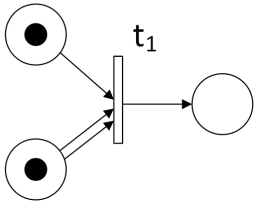
Срабатывание перехода предполагается мгновенным и может произойти в любой момент, когда переход разрешен

Единовременно может быть разрешен только один переход

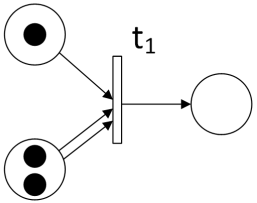
Если в некоторый момент времени не существует разрешенного перехода, то процесс, описывающий сеть Петри, завершается

Выполнение сетей Петри

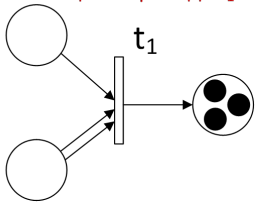
Запрещенный переход



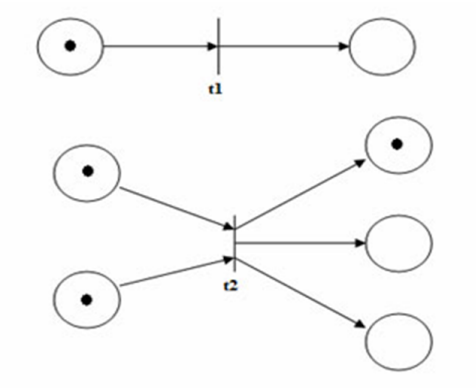
Разрешенный переход



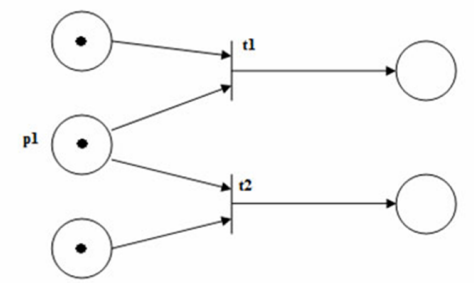
После активации перехода t_1



Виды событий



Одновременное событие



Конфликт (зависит от последовательности выполнения переходов)

Проверка критериев сети Петри при их моделирование

- **Ограниченность** — число меток в любой позиции сети не может превысить некоторого значения K
- **Безопасность** — СП называется безопасной, если каждая позиция содержит не более одной метки
- **Консервативность** — СП называется консервативной, если общее количество меток во всех позициях всегда постоянно
- **Достижимость** — возможность перехода сети из одного заданного состояния (характеризуемого распределением меток) в другое
- **Живучесть** — возможность срабатывания любого перехода при функционировании моделируемого объекта

Некоторые виды сетей Петри

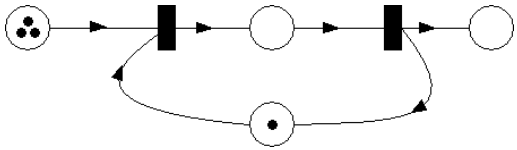
- **Временная** сеть Петри — переходы обладают весом, определяющим продолжительность срабатывания (задержку)
- **Стохастическая** сеть Петри — задержки являются случайными величинами
- **Цветная** сеть Петри — метки могут быть различных типов, обозначаемых цветами, тип метки может быть использован как аргумент в функциональных сетях
- **Ингибиторная** сеть Петри — возможны ингибиторные дуги, запрещающие срабатывания перехода, если во входной позиции, связанной с переходом ингибиторной дугой, находится метка
- **Иерархическая** сеть — содержит немгновенные переходы, в которые вложены другие, возможно, также иерархические, сети

Физическая интерпретация при моделировании компьютерных сетей

- Позиции — место хранения данных
- Метки — передаваемые данные
- Переходы — обработка данных

Конечные автоматы vs Сети Петри

Конечный Автомат



Сети Петри

Механизмы синхронизации процессов

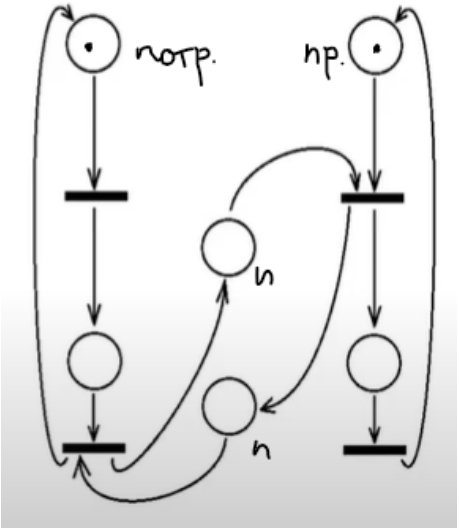
Взаимодействие процессов требует распределения ресурсов между ними. Чтобы система работала правильно распределением ресурсов необходимо управлять. Это управление в основном сводится к задачам синхронизации взаимодействия процессов

Задача о производителе/потребителе

Процесс производитель порождает компоненту информации и размещает ее в буфер. Процесс потребитель ждет пока компонента информации разместится в буфере, после этого он может ее удалить из буфера и использовать
Как правило используется буфер ограниченной емкости, то есть имеется возможность разместить в нем не более n компонентов данных

Задача о производителе/потребителе

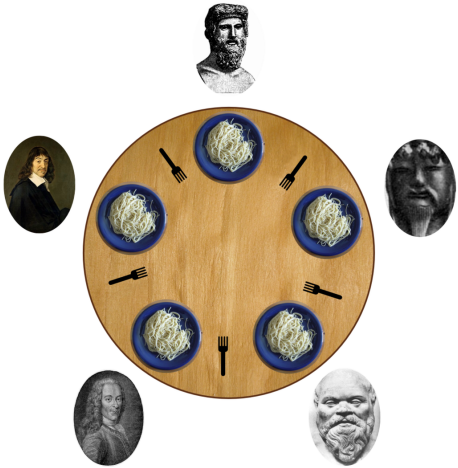
Буферу сопоставлены две позиции:
B — указывает на количество компонентов данных размещенных в буфере, но еще не использованных
B^0 — количество свободных мест в буфере для размещения компонентов данных



Недостатки

- Моделируют управление потоком, но не сам поток данных
- Сложность описания одновременных действий, происходящих во время вычислительного процесса
- Физическая интерпретация перехода в сетях Петри весьма сомнительна

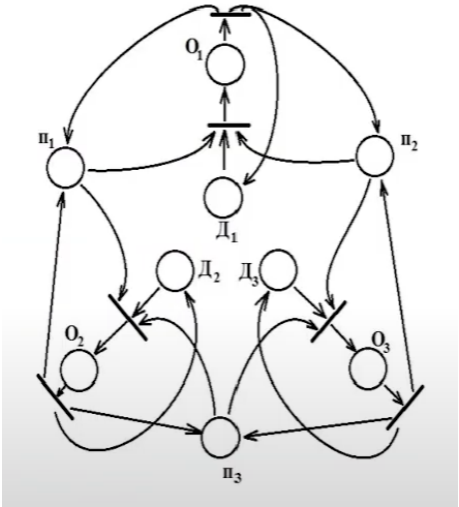
Задачка



Пять безмолвных философов сидят вокруг круглого стола, перед каждым философом стоит миска с рисом и по бокам палочки для еды. Палочка лежит на столе между каждой парой ближайших философов.

Каждый философ может либо есть, либо размышлять. Количество риса не ограничено. Философ может есть только тогда, когда держит две палочки — взятую справа и слева

Решение



Пять безмолвных философов сидят вокруг круглого стола, перед каждым философом стоит миска с рисом и по бокам палочки для еды. Палочка лежит на столе между каждой парой ближайших философов.

Каждый философ может либо есть, либо размышлять. Количество риса не ограничено. Философ может есть только тогда, когда держит две палочки — взятую справа и слева

Тестирование на основе сетей Петри

1. Создаем модель системы, используя сеть Петри
2. Генерируем на основе модели тесты

Создание модели системы

Помимо моделей Петри, модели различных систем можно строить также на основе:

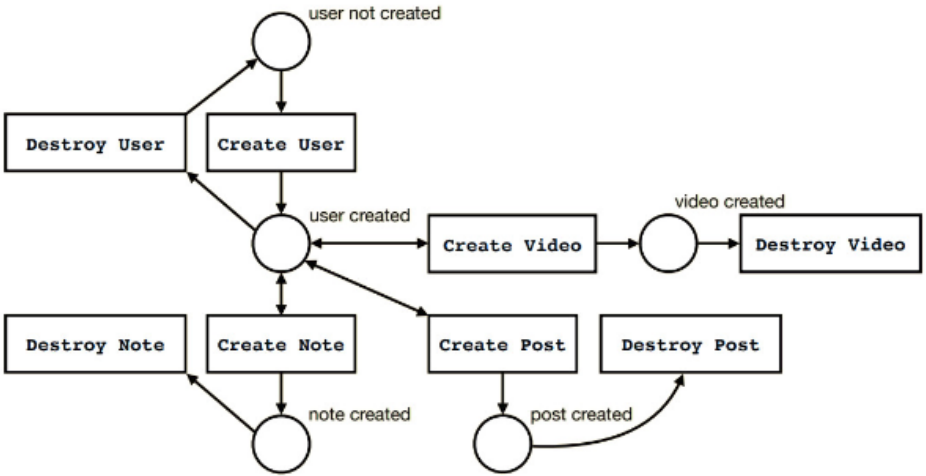
- Потокового графа
- Процессной сети Кана (KPN)
- Конечных автоматов

Где всё это строить?

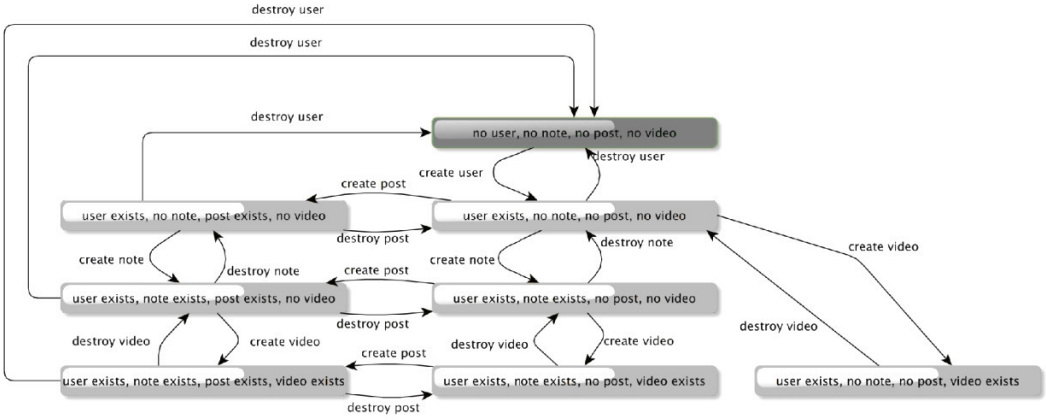
Для этой задачи придумали Desing/CPN (ЯП - ML, 1989)
Позже расширили функциональность и создали CPN tools (2000-2010)

- https://cpntools.org
- http://daze.ho.ua/cpnmpru.pdf

Модель на основе сетей Петри



Модель на основе конечных автоматов



Парсинг в json

```
{
   "places": [
      {
         "guid": "e57c0830-148f-11e8-bb7f-adf9986e1043
         "identifier": "user created"
      }
   ],
   "transitions": [
      {
         "guid": ‘‘df65a6e0-148f-11e8-bb7f-adf9986e1043"
         "identifier": "Create User
      }
   ],
   "arcs": [
      {
         "from_guid": "e57c0830-148f-11e8-bb7f-adf9986e1043
         "to_guid": ‘‘df65a6e0-148f-11e8-bb7f-adf9986e1043"
      }
   ]
}
```

Загрузка модели

Мы умеем хранить .json (.cpn) модель
Что дальше?

Загрузка модели

Реализовать свою библиотеку или поискать решение, придуманное за вас.

- Ruby
- Kotlin
- Python
- C/C++

Пример #1

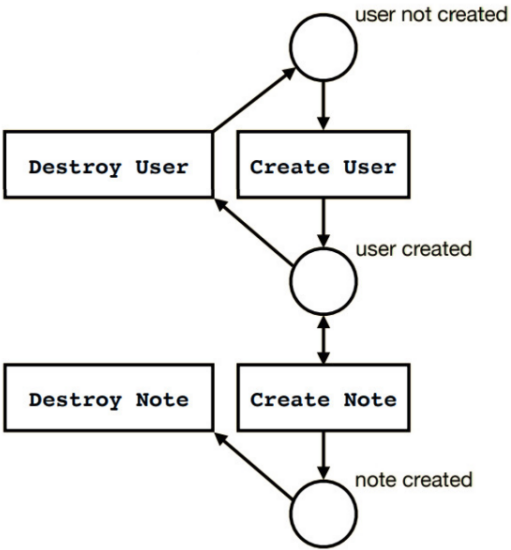
non-UI CPN Tools on C++

```
CCPNet *cpnet = new CCPNet();
cpnet->addPlace("P1 "int CValue("int 1, "int 10));
cpnet->addPlace("P2 "int");
cpnet->addPlace("P3 "int");
cpnet->addTransition("T1_2");
cpnet->transitionAddFromPlaceAndValue("T1_2 "P1 CValue("int 1, "int 10));
cpnet->transitionAddToPlaceAndValue("T1_2 "P2 CValue("int 1, "int 10));
cpnet->addTransition("T2_3");
cpnet->transitionAddFromPlaceAndValue("T2_3 "P2 CValue("int 1, "int 10));
cpnet->transitionAddToPlaceAndValue("T2_3 "P3 CValue("int 1, "int 10));
cpnet->initAllTransitions();
cpnet->fireToEnd();

cout ≪ cpnet->places2String();
```

Пример #2

```
class UserNoteNet < Rhizome::Net
   runner { Runner.new('user-note.bpf', marking: ['user not created']) }
   transition('Create User') { @user = User.create! }
   transition('Destroy User') { @user.destroy }
   transition('Create Note') { @note = Note.create!(user: @user) }
   transition('Destroy Note') { @note.destroy}
   place('user not created') do
      empty { expect(User.count).to be> 0 }
      filled { expect(User.count).to eq(0) }
   end
   place('user created') do
      empty { expect(User.count).to eq(0) }
      filled { expect(User.count).to be > 0 }
   end
   place('note created') do
      empty { expect(Note.count).to eq(0) }
      filled { expect(Note.count).to be > 0 }
   end

end
```

Схема программы



Обход сети

Существуют разные алгоритмы обхода, рассмотрим один из них:

1. Даем каждому переходу одинаковую вероятность (4 перехода = 0,25 каждому)
2. Генерируем случайное число - порог вероятности выбора перехода
3. Выбираем первый переход с подходящей под порог вероятностью, который может сработать
4. Срабатываем переход
5. Проверяем пустые и заполненные позиции
6. Уменьшаем вероятность сработанного перехода в 2 раза
7. Если все переходы сработали хотя бы раз, останавливаемся. Если нет, повторяем спункта 2

Успешные прогоны программы

```
$ bin/rhizome
Traversing user_note net:
  Initial marking: user not created
  Executing Create User
     New marking: user created
    Transitions left: Destroy User, Create Note, Destroy Note
  Executing Create Note
     New marking: user created, note created
    Transitions left: Destroy User, Destroy Note
  Executing Destroy Note
     New marking: user created
    Transitions left: Destroy User
  Executing Destroy User
     New marking: user not created
Finished traversing user_note net.
```

Успешные прогоны программы

```
$ bin/rhizome
Traversing user_note net:
  Initial marking: user not created
  Executing Create User
     New marking: user created
    Transitions left: Destroy User, Create Note, Destroy Note
  Executing Destroy User
     New marking: user not created
    Transitions left: Create Note, Destroy Note
  Executing Create User
     New marking: user created
    Transitions left: Create Note, Destroy Note
  Executing Create Note
     New marking: user created, note created
    Transitions left: Destroy Note
  Executing Destroy Note
     New marking: user created
Finished traversing user_note net.
```

Баг или фича?

```
$ bin/rhizome
Traversing user_note net:
  Initial marking: user not created
  Executing Create User
     New marking: user created
    Transitions left: Destroy User, Create Note, Destroy Note
  Executing Create Note
     New marking: user created, note created
    Transitions left: Destroy User, Destroy Note
  Executing  Destroy User
```

```
ActiveRecord::InvalidForeignKey: PG::ForeignKeyViolation: ERROR: update or delete on
table "users"violates foreign key constraint "fk_rails_83d9817008" on table "notes"
DETAIL: Key (id)=(1) is still referenced from table ‘‘notes".
```

Преимущества

Преимущества тестирования на основе сетей Петри относительно обычного тест фреймворка:

- Меньше кода (время на написание и поддержку тестов сократилось)
- Тестовое покрытие увеличилось
- Время выполнения осталось прежним

Ссылки

- Chemistry Petri net
- Пример со слайдов
- Биография Петри
- Обедающие философы
- Способы задания сетей Петри
- Хорошая лекция про сети Петри
- Тестирование на основе сетей Петри
- Конечные автоматы и сети Петри