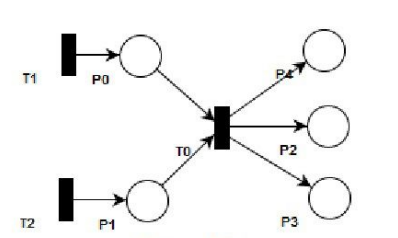
|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Расчётно-графическое задание | | |
| по дисциплине «Теория вычислительных процессов и структур» | | |
| **Сети Петри** | | |
|  | | |
|  | Бригада №6 | Ерощенко Артем, Кауфман Яна, Цыренова Сарюна |
| Группа ПМИ-33 |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватель | Дворецкая виктория константиновна |
|  |  |
| Новосибирск,2024 | | |

**1.** **Цель работы:**

**2. Задание:**

Для данной маркированной сети Петри, заданной графом, файл с вариантами заданий прикреплён.

****

Задание 1. Найти матрицы *D-*, *D+*, составную матрицу изменений *D*.

Задание 2. Построить дерево достижимости.

Задание 3. Провести анализ свойств сети Петри.

**3. Анализ задачи:**

**Входные данные: JSON файл сети Петри**

**Выходные данные:** матрицы D+, D-, D, дерево достижимости.

**Метод решения:** Программа считывает данные из **JSON файл сети Петри. Функция print\_destination\_tree реализует построение дерева достижимости для сети Петри на основе заданных параметров. В последствии, в консоль выводятся** матрицы D+, D-, D, дерево достижимости.

**4. Код программы:**

import json # To read json file of Petri network

import numpy as np # To manipulate matrixes

def print\_destination\_tree(start\_label: np.ndarray, network: dict, transitions: list, points: list, steps: int, cur\_step = 0, cur\_trans = ''):

if cur\_step >= steps:

return

if cur\_step > 0:

print('|' \* (cur\_step - 1) + '∟', end='')

print(cur\_trans, start\_label, sep='')

points\_to, labels\_in = [l for l\_ind, l in enumerate(points) if start\_label[l\_ind] != 0], [l for l in start\_label if l != 0]

for trans in transitions:

cur\_label = np.zeros(len(points), dtype=np.int32)

for i in network[trans]["inp"]: # Count every transition entry if it is in start\_label

if i[0] in points\_to:

i\_ind = points\_to.index(i[0])

cur\_label += np.array([(labels\_in[i\_ind] if i[0] == j else 0) for j in points], dtype=np.int32)

cur\_label\_max = np.max(cur\_label)

if cur\_label\_max > 0:

out\_label = np.zeros(len(points), dtype=np.int32) # Count every transition out

for out in network[trans]["out"]:

out\_label += np.array([int(out[0] == j) for j in points], dtype=np.int32)

# Recursively print out-labels

print\_destination\_tree(start\_label - cur\_label + out\_label \* cur\_label\_max, network, transitions, points, steps, cur\_step + 1, trans)

# Reading Petri network

with open('.\RGR\graph.json', 'r') as f:

petri = json.load(f)

# Some Global variables

points = petri["points"]

transitions = petri["transitions"]

points\_len, transitions\_len = len(points), len(transitions)

tree\_steps\_c = 5

start\_tree\_label = [1, 0, 0, 0, 0]

# Task 1: Destination matrix

# Fill destination matrixes

D\_minus, D\_plus = np.zeros((transitions\_len, points\_len), dtype=np.int32), np.zeros((transitions\_len, points\_len), dtype=np.int32)

for trans\_ind, trans in enumerate(transitions):

# For input matrix

for point in petri["network"][trans]["inp"]:

point\_ind = points.index(point[0])

D\_minus[trans\_ind, point\_ind] = 1

# For output matrix

for point in petri["network"][trans]["out"]:

point\_ind = points.index(point[0])

D\_plus[trans\_ind, point\_ind] = 1

# Print Destination matrix

print('D+:\n', D\_plus)

print('D-:\n', D\_minus)

print('D:\n', D\_plus - D\_minus)

# Task 2: Destination tree

# Print destianation tree for every label

start\_tree\_label = np.array(start\_tree\_label, dtype=np.int32)

print(f"Destination tree of steps: {tree\_steps\_c}:")

print\_destination\_tree(start\_tree\_label, petri["network"], transitions, points, tree\_steps\_c)

**5.Задание №1**

D- = D+= D =

**6.Задание №2**

Дерево достижимости: [1 0 0 0 0]

∟T0[0 0 1 1 1]

**7.Задание №3**

1. Сеть k-ограничена, так как выходные позиции не стремятся к бесконечности. Неважно сколько будет меток в p0,p1, в позиции p4,p2,p3 будет переходить по 1 метке. Ни одна из позиций не накапливает бесконечное количество меток. Ни в одной позиции сети при любой последовательности срабатываний переходов количество фишек не превышает некоторого K

2. Сеть безопасна, так как может быть ограничена 1 меткой. Если в позициях p0,p1 будет по 1 метке, то в позициях p4,p2,p3 будет содержать по 1 метке.

3. Сеть сохраняема, так как число циркулирующих в ней меток постоянна.

4. Сеть достижима, так как все позиции-вершины достижимы. Существует цепочка срабатываний переходов, ведущая из начального состояния p0,p1 в S(p4,p2,p3).

5. Сеть не живая, так как существуют два перехода t1,t2 для которых не существует входных дуг, и они не выполняются. Также переход t0 приводит к тупиковой разметке.

1

0