



**Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине:

Разработка PLM

Тема лабораторной работы **Реализация сети Петри**

Студент: **Абидоков Р. Ш.**
Группа: **РК6-11М.**
Преподаватель: **Жук Д. М.**

Москва, 2020

Оглавление

Задание на лабораторную работу	3
Описание программной реализации	3
Примеры работы	4

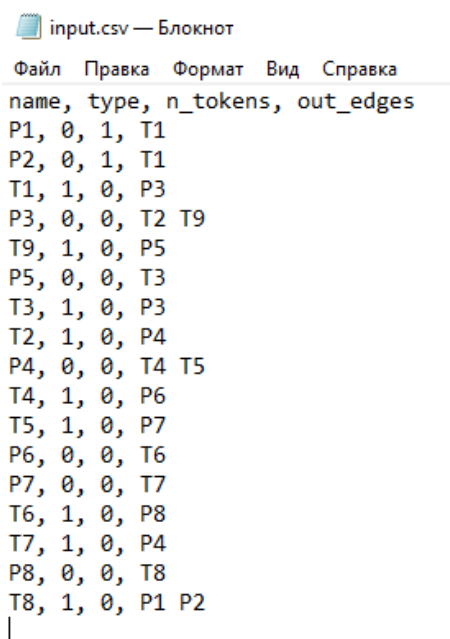
Задание на лабораторную работу

Реализовать программу, принимающую на вход структуру сети Петри и первоначальную разметку, проводящую моделирование работы сети и производящую построение графа достижимости с указанием тупиковых вершин. Разработать формат входных данных.

Описание программной реализации

Программа написана на языке Python 3.7.2 с использованием библиотек Numpy и Pandas для удобной работы с массивами и таблицами.

Входные данные считываются из .csv-файла, пример которого показан на Рис. 1. Структура файла следующая: первый столбец – имя вершины, второй столбец – тип вершины (0 для позиции, 1 для перехода), третий столбец – количество меток при начальной разметке, четвертый столбец – разделенные пробелами имена вершин, ребра от которых входят в текущую вершину.



```
input.csv — Блокнот
Файл  Правка  Формат  Вид  Справка
name, type, n_tokens, out_edges
P1, 0, 1, T1
P2, 0, 1, T1
T1, 1, 0, P3
P3, 0, 0, T2 T9
T9, 1, 0, P5
P5, 0, 0, T3
T3, 1, 0, P3
T2, 1, 0, P4
P4, 0, 0, T4 T5
T4, 1, 0, P6
T5, 1, 0, P7
P6, 0, 0, T6
P7, 0, 0, T7
T6, 1, 0, P8
T7, 1, 0, P4
P8, 0, 0, T8
T8, 1, 0, P1 P2
|
```

Рис. 1 Пример входного файла

Алгоритм работы программы следующий:

1. Происходит парсинг входного файла и разделение данных на позиции и переходы

2. Поскольку сеть Петри – двудольный граф, происходит построение двух листов смежности – в первом для каждого перехода хранятся позиции, к которым существуют выходные ребра, во втором – для каждого перехода позиции, из которых существуют выходные ребра.
3. Параллельно с построением листов смежности происходит формирование начального состояния в виде (c_1, c_2, \dots, c_n) , где $c_i \in \{0,1\}$ – количество меток в i -й позиции, n – количество позиций
4. Производится построение графа достижимости в виде листа смежности, где для каждого достижимого состояния хранятся состояния, переход к которым возможен из текущего.
Обход происходит методом, похожим на обход в ширину – из очереди берется состояние, для него находятся возможные переходы, если найденный переход не был рассмотрен ранее, он добавляется в лист смежности, а соответствующее ему следующее состояние – в очередь. Повторяется до тех пор, пока очередь не пуста.
5. Полученный список выводится в файл out_graph.csv
6. В консоль в качестве достижимых выводятся состояния, присутствующие в листе смежности, в качестве тупиковых – присутствующие в листе, но не имеющие выходных ребер

Полный исходный код программы приведен в приложенных файлах.

Примечание: программой не обрабатываются случаи, когда в позициях может быть больше одной метки, однако в общем такие ситуации возможны.

Примеры работы

В качестве примеров рассмотрены две сети Петри. В первом примере взята сеть, схема которой приведен на Рис. 2

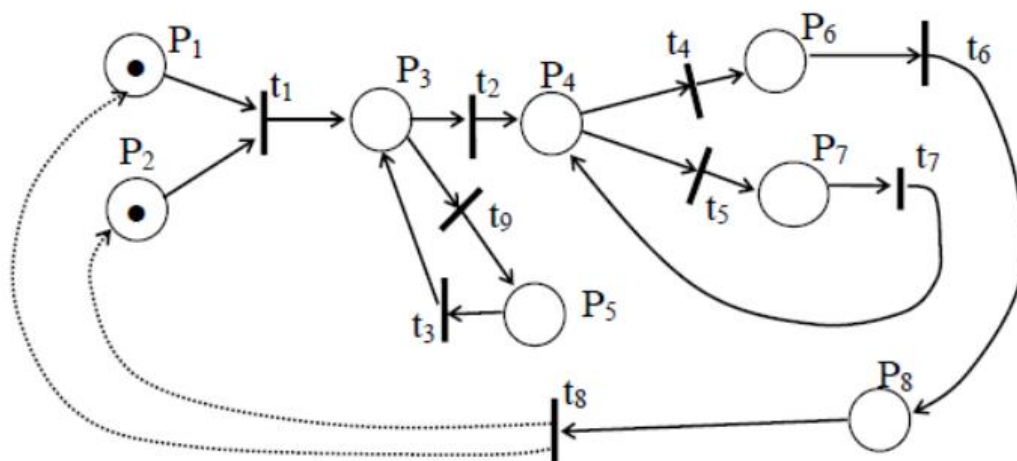


Рис. 2

Данная сеть не имеет тупиковых вершин, ее граф достижимости приведен на Рис.3

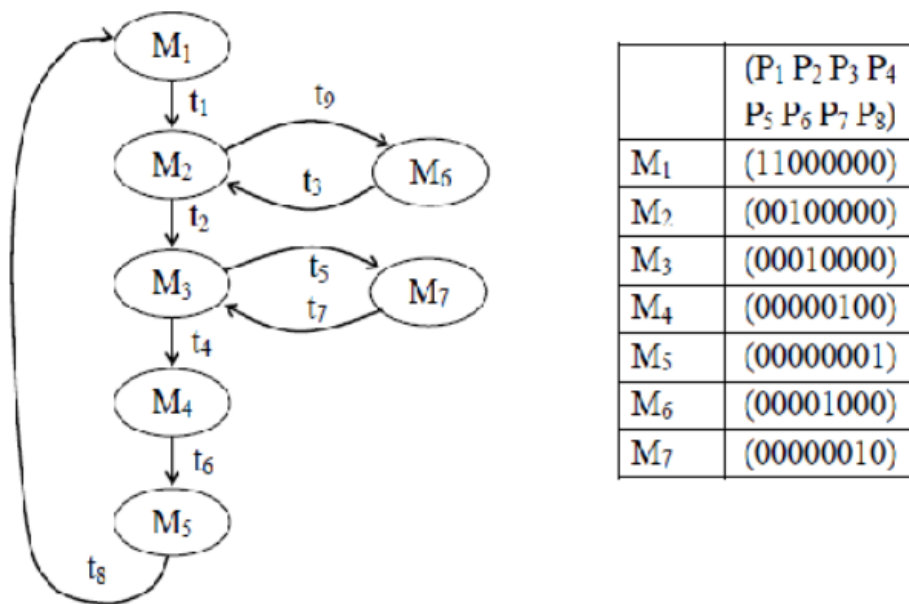


Рис. 3

Результат работы для данного примера приведен на Рис. 4 и 5

Достижимые состояния							
num	P1	P2	P3	P5	P4	P6	P7 P8
192	192					11000000	reachability
32	32					00100000	1
16	16					00010000	1
8	8					00001000	1
4	4					00000100	1
2	2					00000010	1
1	1					00000001	1
=====							
Недостижимые состояния							
num	P1	P2	P3	P5	P4	P6	P7 P8
255	255					11111111	reachability
254	254					11111110	0
253	253					11111101	0
252	252					11111100	0
251	251					11111011	0
..
7	7					00000111	0
6	6					00000110	0
5	5					00000101	0
3	3					00000011	0
0	0					00000000	0
[249 rows x 3 columns]							

Рис. 4

```

=====
Лист смежности
      state      next
0  11000000      00100000
1  00100000  00010000  00001000
6  00010000      00100000
2  00001000  00000100  00000010
4  00000100      00000001
3  00000010      00001000
5  00000001      11000000

=====
Тупиковые вершины
Series([], Name: state, dtype: object)

Process returned 0 (0x0)      execution time : 0.721 s
Для продолжения нажмите любую клавишу . . .

```

Рис. 5

Для проверки обнаружения тупиковых состояний в примере 2 из приведенной выше сети убран переход t8 и, соответственно, ребра, соединяющие его с P8, P1, P2. В этом случае в конце вывода программы приводится тупиковая вершина (Рис. 6)

```

=====
Лист смежности
      state      next
0  11000000      00100000
1  00100000  00010000  00001000
6  00010000      00100000
2  00001000  00000100  00000010
4  00000100      00000001
3  00000010      00001000
5  00000001

=====
Тупиковые вершины
5  00000001
Name: state, dtype: object

Process returned 0 (0x0)      execution time : 0.711 s
Для продолжения нажмите любую клавишу . . .

```

Рис. 6