

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Построение операционной графовой модели программы
(ОГМП) и расчет характеристик эффективности ее выполнения
методом эквивалентных преобразований»

Студентка гр. 8304

Николаева М. А.

Преподаватель

Кирияничков В. А.

Санкт-Петербург

2022

Цель работы.

Построение операционной графовой модели программы (ОГМП) и расчет характеристик эффективности ее выполнения методом эквивалентных преобразований.

Задание.

Для задания из лабораторных работ 1-3 разработать операционную модель управляющего графа программы на основе схемы алгоритма. При выполнении работы рекомендуется для упрощения обработки графа исключить диалог при выполнении операций ввода-вывода данных, а также привести программу к структурированному виду.

Выбрать вариант графа с нагруженными дугами, каждая из которых должна представлять фрагмент программы, соответствующий линейному участку или ветвлению. При расчете вероятностей ветвлений, зависящих от распределения данных, принять равномерное распределение обрабатываемых данных в ограниченном диапазоне (например, $[0,100]$ - для положительных чисел или $[-100,100]$ - для произвольных чисел). В случае ветвлений, вызванных проверкой выхода из цикла, вероятности рассчитываются исходя априорных сведений о числе повторений цикла. Сложные случаи оценки вероятностей ветвлений согласовать с преподавателем.

В качестве параметров, характеризующих потребление ресурсов, использовать времена выполнения команд соответствующих участков программы. С помощью монитора `Sampler_v2` выполнить оценку времен выполнения каждого линейного участка в графе программы.

Полученную ОГМП, представить в виде графа с нагруженными дугами, у которого в качестве параметров, характеризующих потребление ресурсов на дуге ij , использовать тройку $\{P_{ij}, M_{ij}, D_{ij}\}$, где:

P_{ij} - вероятность выполнения процесса для дуги ij ;

M_{ij} - мат. ожидание потребления ресурса процессом для дуги ij ;

D_{ij} - дисперсия потребления ресурса процессом для дуги ij .

В качестве потребляемого ресурса в данной работе рассматривается время процессора, а оценками мат. ожиданий времен для дуг исходного графа следует принять времена выполнения операторов (команд), соответствующих этим дугам участков программы. Дисперсиям исходных дуг следует присвоить нулевые значения.

Получить описание полученной ОГМП на входном языке пакета CSA III в виде поглощающей марковской цепи (ПМЦ) – (англ.) AMC (absorbing Markov chain) и/или эргодической марковской цепи (ЭМЦ) - EMC (ergodic Markov chain).

С помощью предоставляемого пакетом CSA III меню действий выполнить расчет среднего времени и дисперсии времени выполнения как для всей программы, так и для ее фрагментов, согласованных с преподавателем. Сравнить полученные результаты с результатами измерений, полученными в работе 3.

Ход работы.

1. Текст программы (исходный).

Исходный код программы представлен в приложении А.

2. Профилирование.

Исходный код программы был структурирован, из него были убраны операции ввода/вывода. Код программы для профилирования, разделенной на функциональные участки, представлен в приложении Б.

3. Граф управления программы.

Был построен граф управления программы. Линейные участки программы были объединены в одну дугу для минимизации размеров графа. Результат представлен на рисунке 1.

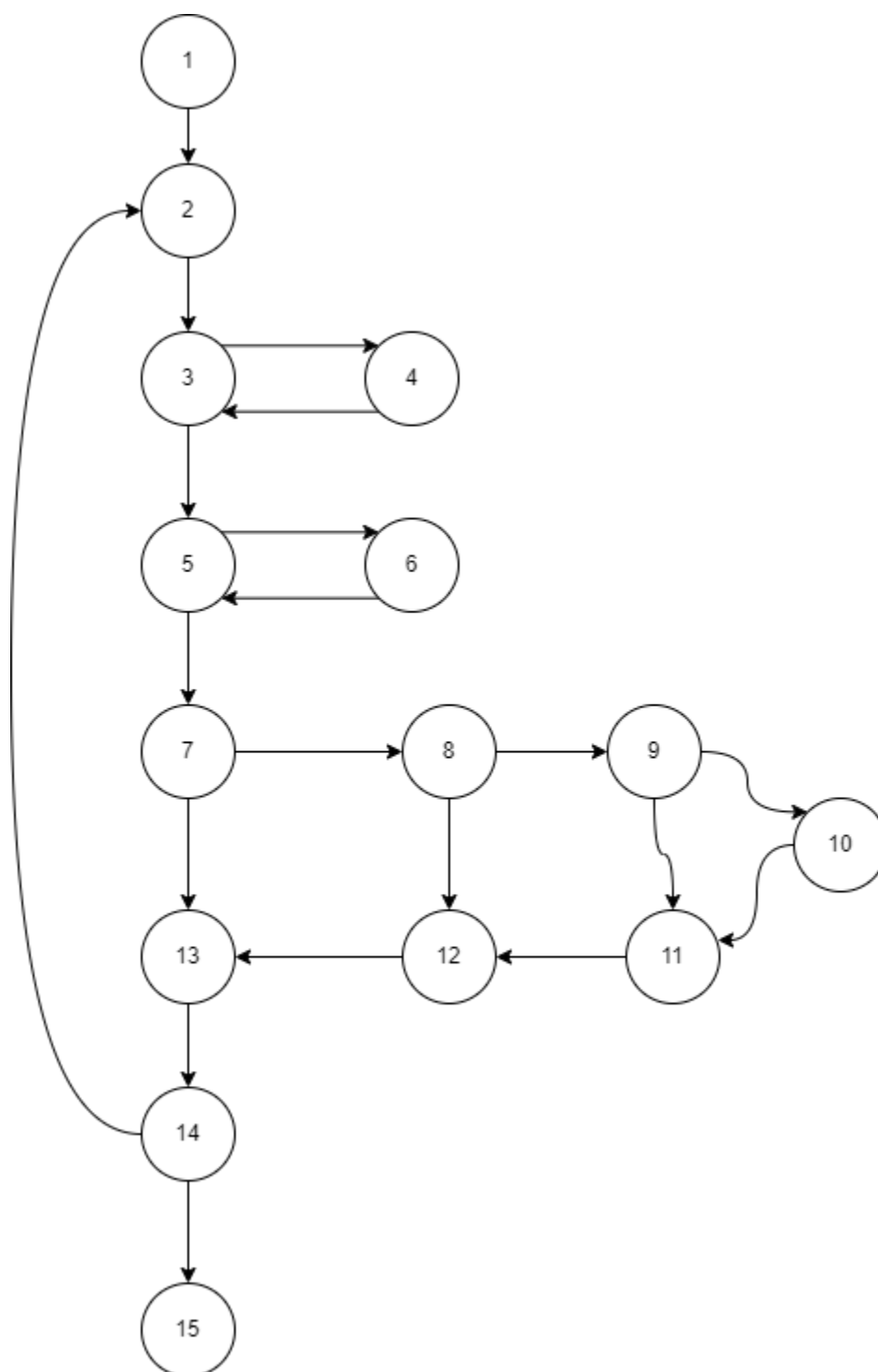


Рисунок 1 - Граф управления программы

4. Результаты профилирования.

В программу, разделенную на ФУ, из лабораторной работы №3 были добавлены некоторые метки для более полной картины измерений. Результаты профилирования представлены на рисунке 2.

исх	прием	общее время	кол-во проходов	среднее время
111	18	23.662	1	23.662
18	35	56.819	1	56.819
35	38	1.261	1	1.261
38	47	33.179	6	5.530
47	48	22.102	6	3.684
48	50	221.598	63	3.517
48	56	44.216	6	7.369
50	54	1265.856	63	20.093
54	48	164.335	63	2.608
56	61	24.681	6	4.113
61	62	22.560	6	3.760
62	64	100.895	21	4.805
62	71	37.783	6	6.297
64	69	422.861	21	20.136
69	62	117.436	21	5.592
71	73	13.575	6	2.263
73	96	18.068	3	6.023
73	76	14.756	3	4.919
96	98	9.897	6	1.649
98	38	12.392	5	2.478
98	100	10.470	1	10.470
76	78	9.432	3	3.144
78	92	60.359	2	30.179
78	82	16.432	1	16.432
92	94	3.630	3	1.210
94	96	4.319	3	1.440
82	84	4.602	1	4.602
84	92	1.709	1	1.709
100	113	29.091	1	29.091

Рисунок 2 – Результаты профилирования

5. Расчет вероятностей и затрат ресурсов для дуг управляющего графа.

Был произведен расчет вероятностей и затрат ресурсов для дуг графа. Результаты представлены в таблице 1.

Таблица 1 – Расчет вероятностей и затрат ресурсов для дуг.

	Номера строк	Количество проходов	Вероятность
$L_{1-2} = 81,742$	111-18;18-35;35-38	1	1
$L_{2-3} = 9,214$	38-47;47-48	6	1
$L_{3-4} = 23,61$	48-50;50-54	63	0,913
$L_{4-3} = 2,608$	54-48	63	1
$L_{3-5} = 15,242$	48-56;56-61;61-62	6	0,087
$L_{5-6} = 24,941$	62-64;64-69	21	0,78
$L_{6-5} = 5,592$	69-62	21	1
$L_{5-7} = 8,56$	62-71;71-73	6	0,22
$L_{7-8} = 4,919$	73-76	3	0,5

Таблица 1 (продолжение) – Расчет вероятностей и затрат ресурсов для дуг.

	Номера строк	Количество проходов	Вероятность
$L_{8-9} = 3,144$	76-78	3	1
$L_{8-12} = 0$	76-94	0	0
$L_{9-10} = 16,432$	78-82	1	0,33
$L_{9-11} = 30,179$	78-92	2	0,67
$L_{10-11} = 6,311$	82-84;84-92	1	1
$L_{11-12} = 1,21$	92-94	3	1
$L_{12-13} = 1,44$	94-96	3	1
$L_{7-13} = 6,023$	73-96	3	0,5
$L_{13-14} = 1,649$	96-98	6	1
$L_{14-2} = 2,478$	98-38	5	0,833
$L_{14-15} = 39,561$	98-100;100-113	1	0,167

6. Операционная графовая модель программы.

На основе таблицы 1 и рисунка 1 была построена операционная графовая модель программы. Результат представлен на рисунке 3.

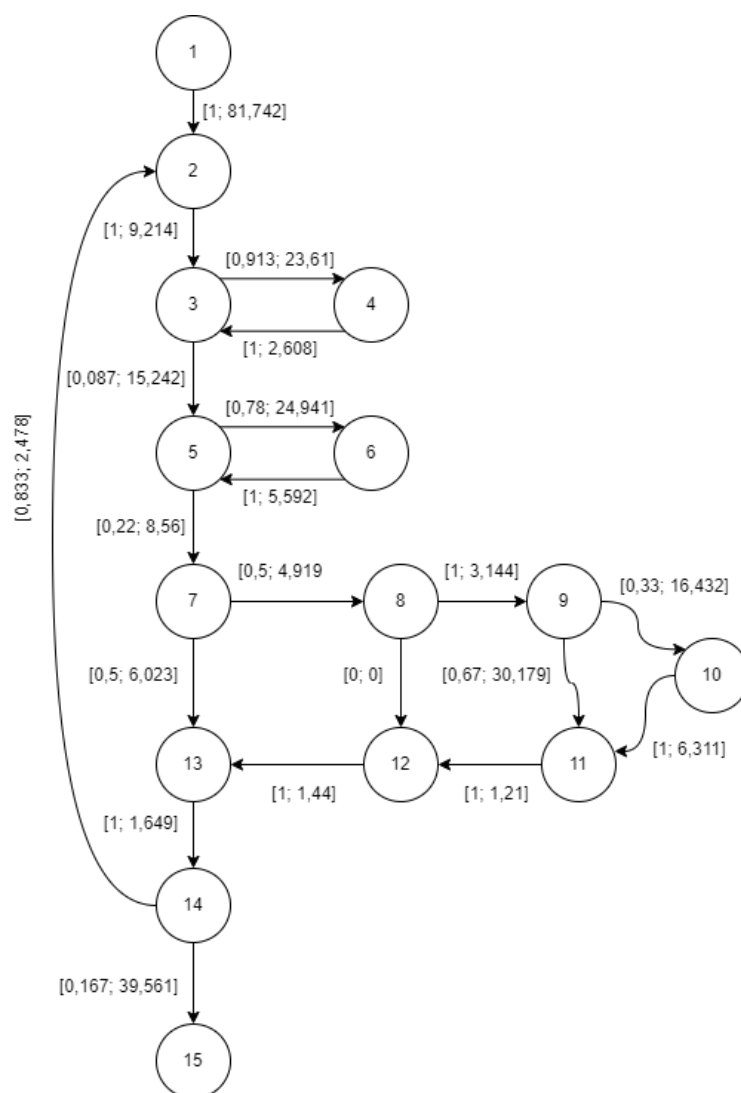


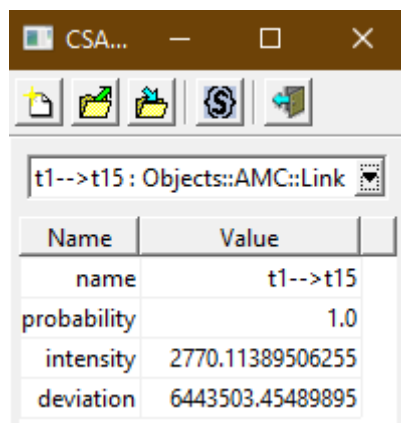
Рисунок 3 - ОГМП

7. Описание модели model.xml.

ОГМП была записана в виде xml-файла. Описание модели представлено в приложении В.

8. Результаты.

Программа была запущена, граф был рассчитан. Результаты работы программы представлены на рисунке 4.



The screenshot shows a window titled 'CSA...' with a toolbar containing icons for file operations and a search function. Below the toolbar is a text field containing 't1-->t15 : Objects::AMC::Link'. Below this is a table with two columns: 'Name' and 'Value'.

Name	Value
name	t1-->t15
probability	1.0
intensity	2770.11389506255
deviation	6443503.45489895

Рисунок 4 - Результат работы программы

Согласно расчётам программы, среднее время выполнения составляет 2770,113 мкс. В пункте 4 данного отчёта приведен результат профилирования программы с использованием SAMPLER_v2, где суммарное время выполнения составило 2767,976 мкс. В итоге, разница между результатами составляет менее 0.1 %.

Выводы.

В ходе выполнения лабораторной работы была построена операционная графовая модель заданной программы, нагрузочные параметры которой были оценены с помощью профилировщика SAMPLER_v2 и методом эквивалентных преобразований с помощью пакета CSA III были вычислены математическое ожидание и дисперсия времени выполнения как для всей программы, так и для фрагментов программы. Результаты сравнения этих характеристик с полученными в ходе выполнения лабораторной работы №3 согласуются.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <stdio.h>
#include <math.h>
#include <stdbool.h>

float fx (float x)
{
    return (1.0 / x);
}

float romb (float lower, float upper, float tol)
{
    int nx [16];
    float t [136];

    bool done = false;
    bool error = false;
    int pieces = 1;
    nx[1] = 1;
    float delta_x = (upper - lower) / pieces;
    float c = (fx(lower) + fx(upper)) * 0.5;
    t[1] = delta_x * c;
    int n = 1;
    int nn = 2;
    float sum = c;

    float fotom,x;
    int l,i,j,k,nt,ntra;
    do
    {
        n = n+1;
        fotom = 4;
        nx[n] = nn;
        pieces = pieces * 2;
        l = pieces - 1;
        delta_x = (upper - lower) / pieces;

        int ll = (l+1)/2;
        for(int ii = 1; ii <= ll; ii++)
        {
            i = ii * 2 - 1;
            x = lower + i * delta_x;
            sum = sum + fx(x);
        }

        t[nn] = delta_x * sum;

        ntra = nx[n-1];
        k = n-1;

        for(int m = 1; m <= k; m++)
```

```

    {
        j = nn+m;
        nt = nx[n - 1] + m - 1;
        t[j] = (fotom * t[j - 1] - t[nt]) / (fotom-1.0);
        fotom = fotom * 4;
    }

    if (n > 4)
    {
        if (t[nn + 1] != 0.0) {
            if ((fabs(t[ntra+1]-t[nn+1])<=fabs(t[nn+1]*tol))
                || (fabs(t[nn-1]-t[j])<=fabs(t[j]*tol)))
            {
                done = true;
            } else
            if (n>15) {
                done = true;
                error = true;
            }
        }
        nn = j+1;
    } while (!done);

    return (t[j]);
}

int main()
{
    const float tol = 1.0E-4;
    float lower = 1.0;
    float upper = 9.0;
    float sum = romb(lower,upper,tol);

    return 0;
}

```

ПРИЛОЖЕНИЕ Б.

КОД ПРОГРАММЫ С РАЗДЕЛЕНИЕМ НА ФУ

```
1 #include <stdio.h>
2 #include <math.h>
3 #include "sampler.h"
4
5 typedef int bool;
6 #define true 1
7 #define false 0
8
9
10 float fx (float x)
11 {
12     return (1.0 / x);
13 }
14
15
16 float romb (float lower, float upper, float tol)
17 {
18     SAMPLE;
19     int nx [16];
20     float t [136];
21
22     bool done = false;
23     bool error = false;
24     int pieces = 1;
25     nx[1] = 1;
26     float delta_x = (upper - lower) / pieces;
27     float c = (fx(lower) + fx(upper)) * 0.5;
28     t[1] = delta_x * c;
29     int n = 1;
30     int nn = 2;
31     float sum = c;
32
33     float fotom,x;
34     int l,i,j,k,nt,ntra;
35     SAMPLE;
36     do
37     {
38         SAMPLE;
39         n = n+1;
40         fotom = 4;
41         nx[n] = nn;
42         pieces = pieces * 2;
43         l = pieces - 1;
44         delta_x = (upper - lower) / pieces;
45
46         int ll = (l+1)/2;
47         SAMPLE;
48         for(int ii = 1; SAMPLE, ii <= ll; ii++)
```

```

49     {
50     SAMPLE;
51     i = ii * 2 - 1;
52     x = lower + i * delta_x;
53     sum = sum + fx(x);
54     SAMPLE;
55     }
56     SAMPLE;
57     t[nn] = delta_x * sum;
58
59     ntra = nx[n-1];
60     k = n-1;
61     SAMPLE;
62     for(int m = 1; SAMPLE, m <= k; m++)
63     {
64         SAMPLE;
65         j = nn+m;
66         nt = nx[n - 1] + m - 1;
67         t[j] = (fotom * t[j - 1] - t[nt]) / (fotom-1.0);
68         fotom = fotom * 4;
69         SAMPLE;
70     }
71     SAMPLE;
72
73     SAMPLE;
74     if (n > 4)
75     {
76         SAMPLE;
77         if (t[nn + 1] != 0.0) {
78             SAMPLE;
79             if ((fabs(t[ntra+1]-t[nn+1])<=fabs(t[nn+1]*tol))
80             || (fabs(t[nn-1]-t[j])<=fabs(t[j]*tol)))
81             {
82                 SAMPLE;
83                 done = true;
84                 SAMPLE;
85             } else
86             if (n>15) {
87                 SAMPLE;
88                 done = true;
89                 error = true;
90                 SAMPLE;
91             }
92             SAMPLE;
93         }
94         SAMPLE;
95     }
96     SAMPLE;
97     nn = j+1;
98     SAMPLE;
99 } while (!done);
100 SAMPLE;
101

```

```
102     return (t[j]);
103 }
104
105 int main(int argc, char **argv)
106 {
107     sampler_init(&argc, argv);
108     const float tol = 1.0E-4;
109     float lower = 1.0;
110     float upper = 9.0;
111     SAMPLE;
112     float sum = romb(lower,upper,tol);
113     SAMPLE;
114     return 0;
115 }
```

ПРИЛОЖЕНИЕ В.

ОПИСАНИЕ МОДЕЛИ .XML

```
<model type = "Objects::AMC::Model" name = "model">
  <node type = "Objects::AMC::Top" name = "t1"></node>
  <node type = "Objects::AMC::Top" name = "t2"></node>
  <node type = "Objects::AMC::Top" name = "t3"></node>
  <node type = "Objects::AMC::Top" name = "t4"></node>
  <node type = "Objects::AMC::Top" name = "t5"></node>
  <node type = "Objects::AMC::Top" name = "t6"></node>
  <node type = "Objects::AMC::Top" name = "t7"></node>
  <node type = "Objects::AMC::Top" name = "t8"></node>
  <node type = "Objects::AMC::Top" name = "t9"></node>
  <node type = "Objects::AMC::Top" name = "t10"></node>
  <node type = "Objects::AMC::Top" name = "t11"></node>
  <node type = "Objects::AMC::Top" name = "t12"></node>
  <node type = "Objects::AMC::Top" name = "t13"></node>
  <node type = "Objects::AMC::Top" name = "t14"></node>
  <node type = "Objects::AMC::Top" name = "t15"></node>
  <link type = "Objects::AMC::Link" name = "t1-->t2" probability
= "1.0" intensity = "81.742" deviation = "0.0" source = "t1" dest =
"t2"></link>
  <link type = "Objects::AMC::Link" name = "t2-->t3" probability
= "1.0" intensity = "9.214" deviation = "0.0" source = "t2" dest =
"t3"></link>
  <link type = "Objects::AMC::Link" name = "t3-->t4" probability
= "0.913" intensity = "23.61" deviation = "0.0" source = "t3" dest =
"t4"></link>
  <link type = "Objects::AMC::Link" name = "t4-->t3" probability
= "1.0" intensity = "2.608" deviation = "0.0" source = "t4" dest =
"t3"></link>
  <link type = "Objects::AMC::Link" name = "t3-->t5" probability
= "0.087" intensity = "15.242" deviation = "0.0" source = "t3" dest =
"t5"></link>
  <link type = "Objects::AMC::Link" name = "t5-->t6" probability
= "0.78" intensity = "24.941" deviation = "0.0" source = "t5" dest =
"t6"></link>
  <link type = "Objects::AMC::Link" name = "t5-->t7" probability
= "0.22" intensity = "8.56" deviation = "0.0" source = "t5" dest =
"t7"></link>
  <link type = "Objects::AMC::Link" name = "t6-->t5" probability
= "1.0" intensity = "5.592" deviation = "0.0" source = "t6" dest =
"t5"></link>
  <link type = "Objects::AMC::Link" name = "t7-->t8" probability
= "0.5" intensity = "4.919" deviation = "0.0" source = "t7" dest =
"t8"></link>
  <link type = "Objects::AMC::Link" name = "t7-->t13"
probability = "0.5" intensity = "6.023" deviation = "0.0" source = "t7"
dest = "t13"></link>
```

```

    <link type = "Objects::AMC::Link" name = "t8-->t9" probability
= "1" intensity = "3.144" deviation = "0.0" source = "t8" dest =
"t9"></link>
    <link type = "Objects::AMC::Link" name = "t8-->t12"
probability = "0.0" intensity = "0.0" deviation = "0.0" source = "t8"
dest = "t12"></link>
    <link type = "Objects::AMC::Link" name = "t9-->t10"
probability = "0.33" intensity = "16.432" deviation = "0.0" source =
"t9" dest = "t10"></link>
    <link type = "Objects::AMC::Link" name = "t9-->t11"
probability = "0.67" intensity = "30.179" deviation = "0.0" source =
"t9" dest = "t11"></link>
    <link type = "Objects::AMC::Link" name = "t10-->t11"
probability = "1.0" intensity = "6.311" deviation = "0.0" source = "t10"
dest = "t11"></link>
    <link type = "Objects::AMC::Link" name = "t11-->t12"
probability = "1.0" intensity = "1.21" deviation = "0.0" source = "t11"
dest = "t12"></link>
    <link type = "Objects::AMC::Link" name = "t12-->t13"
probability = "1.0" intensity = "1.44" deviation = "0.0" source = "t12"
dest = "t13"></link>
    <link type = "Objects::AMC::Link" name = "t13-->t14"
probability = "1.0" intensity = "1.649" deviation = "0.0" source = "t13"
dest = "t14"></link>
    <link type = "Objects::AMC::Link" name = "t14-->t2"
probability = "0.833" intensity = "2.478" deviation = "0.0" source =
"t14" dest = "t2"></link>
    <link type = "Objects::AMC::Link" name = "t14-->t15"
probability = "0.167" intensity = "39.561" deviation = "0.0" source =
"t14" dest = "t15"></link>
</model>

```