

Лабораторная работа №12

Пример моделирования простого протокола передачи данных

Астраханцева А. А.

Содержание

1	Введение	4
1.1	Цель	4
1.2	Задание	4
2	Выполнение лабораторной работы	5
2.1	Упражнение	9
3	Выводы	16
	Список литературы	17

Список иллюстраций

2.1	Граф для модели простого протокола передачи данных	5
2.2	Задание деклараций	6
2.3	Задание деклараций	7
2.4	Модель простого протокола передачи данных	8
2.5	Запуск модели простого протокола передачи данных	9
2.6	Пространство состояний для модели простого протокола передачи данных	15

1 Введение

1.1 Цель

Реализовать простой протокол передачи данных в CPN Tools.

1.2 Задание

- Реализовать простой протокол передачи данных в CPN Tools.
- Вычислить пространство состояний, сформировать отчет о нем и построить граф.

2 Выполнение лабораторной работы

Для начала я построила весь граф целиком, а в дальнейшем написала декларации, задавала начальные значения и типы для состояний (рис. 2.1).

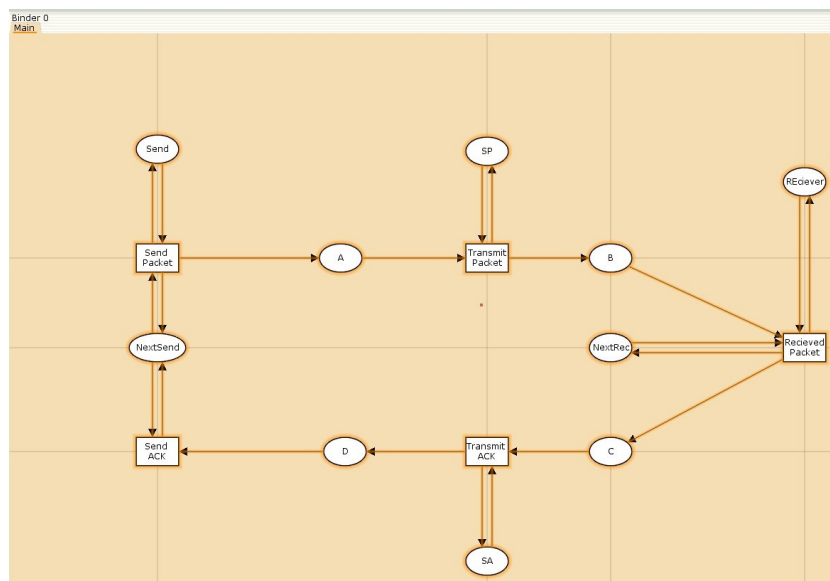


Рис. 2.1: Граф для модели простого протокола передачи данных

Основные состояния: источник (Send), получатель (Receiver). Действия (переходы): отправить пакет (Send Packet), отправить подтверждение (Send ACK). Промежуточное состояние: следующий посылаемый пакет (NextSend). [1] Зададим декларации модели (рис. 2.2).

```

▼ Declarations
  ▼ colset INT = int;
  ▼ colset DATA = string;
  ▼ colset INTxDATA = product INT * DATA;
  ▼ var n, k: INT;
  ▼ var p, str: DATA;
  ▼ val stop = "#####";
  ▼ Standard declarations
    ► colset UNIT
    ► colset BOOL
    ► colset STRING
  ► Monitors
    Main

```

Рис. 2.2: Задание деклараций

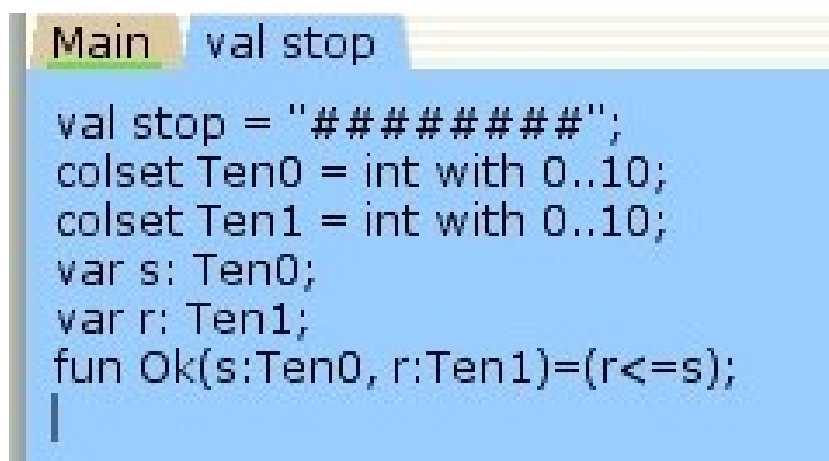
Состояние Send имеет тип INTxDATA и следующую начальную маркировку (в соответствии с передаваемой фразой).

Стоповый байт ("#####") определяет, что сообщение закончилось. Состояние Receiver имеет тип DATA и начальное значение 1' "" (т.е. пустая строка, поскольку состояние собирает данные и номер пакета его не интересует). Состояние NextSend имеет тип INT и начальное значение 1' 1. Поскольку пакеты представляют собой кортеж, состоящий из номера пакета и строки, то выражение у двусторонней дуги будет иметь значение (n, p). Кроме того, необходимо взаимодействовать с состоянием, которое будет сообщать номер следующего посылаемого пакета данных. Поэтому переход Send Packet соединяем с состоянием NextSend двумя дугами с выражениями n. Также необходимо получать информацию с подтверждениями о получении данных. От перехода Send Packet к состоянию NextSend дуга с выражением n, обратно – k.

Зададим промежуточные состояния (A, B с типом INTxDATA, C, D с типом INTxDATA) для переходов: передать пакет Transmit Packet (передаём (n, p)), передать подтверждение Transmit ACK (передаём целое число k). Добавляем

переход получения пакета (Receive Packet). От состояния Receiver идёт дуга к переходу Receive Packet со значением той строки (str), которая находится в состоянии Receiver. Обратно: проверяем, что номер пакета новый и строка не равна стоп-биту. Если это так, то строку добавляем к полученным данным. Кроме того, необходимо знать, каким будет номер следующего пакета. Для этого добавляем состояние NextRec с типом INT и начальным значением 1'1 (один пакет), связываем его дугами с переходом Receive Packet. Причём к переходу идёт дуга с выражением k, от перехода — if n=k then k+1 else k. Связываем состояния B и C с переходом Receive Packet. От состояния B к переходу Receive Packet — выражение (n,p), от перехода Receive Packet к состоянию C — выражение if n=k then k+1 else k. От перехода Receive Packet к состоянию Receiver: if n=k andalso p<>stop then str^p else str. (если n=k и мы не получили стоп-байт, то направляем в состояние строку и к ней прикрепляем p, в противном случае посылаем только строку). На переходах Transmit Packet и Transmit ACK зададим потерю пакетов. Для этого на интервале от 0 до 10 зададим пороговое значение и, если передаваемое значение превысит этот порог, то считаем, что произошла потеря пакета, если нет, то передаём пакет дальше. Для этого задаём вспомогательные состояния SP и SA с типом Ten0 и начальным значением 1'8, соединяем с соответствующими переходами(рис. 2.3):

В декларациях задаём(рис. 2.3):



```

Main  val stop
val stop = "#####";
colset Ten0 = int with 0..10;
colset Ten1 = int with 0..10;
var s: Ten0;
var r: Ten1;
fun Ok(s:Ten0, r:Ten1)=(r<=s);
|

```

Рис. 2.3: Задание деклараций

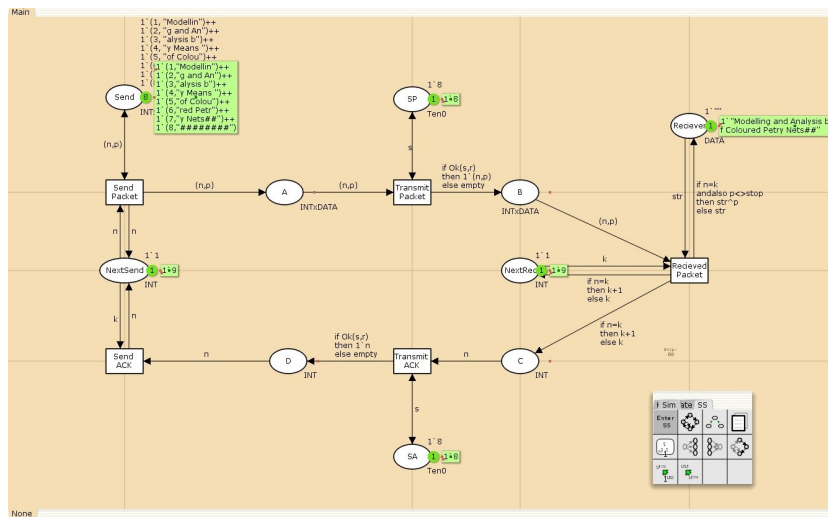


Рис. 2.5: Запуск модели простого протокола передачи данных

2.1 Упражнение

Вычислим пространство состояний. Прежде, чем пространство состояний может быть вычислено и проанализировано, необходимо сформировать код пространства состояний. Этот код создается, когда используется инструмент Войти в пространство состояний. Вход в пространство состояний занимает некоторое время. Затем, если ожидается, что пространство состояний будет небольшим, можно просто применить инструмент Вычислить пространство состояний к листу, содержащему страницу сети. Сформируем отчет о пространстве состояний и проанализируем его. Чтобы сохранить отчет, необходимо применить инструмент Сохранить отчет о пространстве состояний к листу, содержащему страницу сети и ввести имя файла отчета [3].

Из него можно увидеть:

- 8914 состояний и 131231 переходов между ними.
- Указаны границы значений для каждого элемента: промежуточные состояния A, B, C (наибольшая верхняя граница у A, так как после него пакеты отбрасываются. Вспомогательные состояния SP, SA, NextRec, NextSend,

Receiver(в них может находиться только один пакет) и состояние Send(в нем хранится только 8 элементов, так как мы задали их в начале и с ними никаких изменений не происходит).

- Указаны границы в виде мультимножеств.
- Маркировка home для всех состояний (в любую позицию можно попасть из любой другой маркировки).
- Маркировка dead равная 3115 [8914,8913,8912,8911,8910,...] – это состояния, в которых нет включенных переходов.

CPN Tools state space report for:

/home/openmodelica/Documents/cpntools/lab12.cpn

Report generated: Sat Apr 26 13:02:15 2025

Statistics

State Space

Nodes: 8914
Arcs: 131231
Secs: 300
Status: Partial

Scc Graph

Nodes: 4653
Arcs: 107318
Secs: 36

Boundedness Properties

Best Integer Bounds

	Upper	Lower
Main'A 1	19	0
Main'B 1	9	0
Main'C 1	6	0
Main'D 1	4	0
Main'NextRec 1	1	1
Main'NextSend 1	1	1
Main'Reciever 1	1	1
Main'SA 1	1	1
Main'SP 1	1	1
Main'Send 1	8	8

Best Upper Multi-set Bounds

Main'A 1	19`(1,"Modellin")++
14`(2,"g and An")++	
10`(3,"alysis b")++	
5`(4,"y Means ")	
Main'B 1	9`(1,"Modellin")++
7`(2,"g and An")++	
5`(3,"alysis b")++	
2`(4,"y Means ")	
Main'C 1	6`2++
4`3++	
3`4++	
1`5	
Main'D 1	4`2++

```

3`3++
2`4++
1`5
    Main'NextRec 1      1`1++
1`2++
1`3++
1`4++
1`5
    Main'NextSend 1     1`1++
1`2++
1`3++
1`4++
1`5
    Main'Reciever 1     1`""++
1`"Modellin"++
1`"Modelling and An"++
1`"Modelling and Analysis b"++
1`"Modelling and Analysis by Means "
    Main'SA 1          1`8
    Main'SP 1          1`8
    Main'Send 1         1`(1,"Modellin")++
1`(2,"g and An")++
1`(3,"alysis b")++
1`(4,"y Means ")++
1`(5,"of Colou")++
1`(6,"red Petr")++
1`(7,"y Nets##")++
1`(8,"#####")

```

Best Lower Multi-set Bounds

Main'A 1	empty
Main'B 1	empty
Main'C 1	empty
Main'D 1	empty
Main'NextRec 1	empty
Main'NextSend 1	empty
Main'Reciever 1	empty
Main'SA 1	1`8
Main'SP 1	1`8
Main'Send 1	1`(1,"Modellin")++

```
1`(2,"g and An")++  
1`(3,"alysis b")++  
1`(4,"y Means ")++  
1`(5,"of Colou")++  
1`(6,"red Petr")++  
1`(7,"y Nets##")++  
1`(8,"#####")
```

Home Properties

Home Markings

None

Liveness Properties

Dead Markings

3115 [8914,8913,8912,8911,8910,...]

Dead Transition Instances

None

Live Transition Instances

None

Fairness Properties

Main'Recieved_Packet 1 No Fairness

Main'Send_ACK 1 No Fairness

Main'Send_Packet 1 Impartial

Main'Transmit_ACK 1 No Fairness

Main'Transmit_Packet 1 Impartial

Сформируем начало графа пространства состояний, так как их много(рис. 2.6):

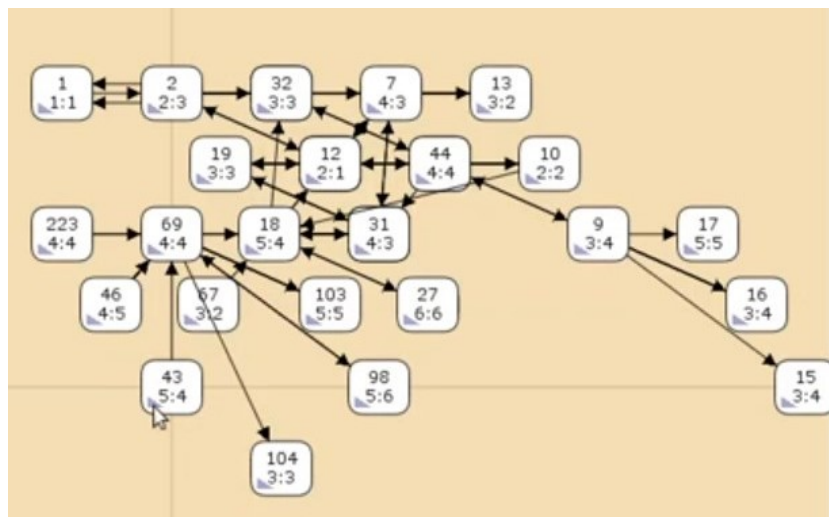


Рис. 2.6: Пространство состояний для модели простого протокола передачи данных

3 Выводы

В процессе выполнения данной лабораторной работы я реализовала простой протокол передачи данных в CPN Tools и проведен анализ его пространства состояний.

Список литературы

1. В. К.А., С. К.Д. Руководство к лабораторной работе №11. Моделирование информационных процессов. Модель системы массового обслуживания M|M|1. 2025. С. 10.
2. Modeling with Coloured Petri Nets. 2018.
3. Beaudouin-Lafon M. и др. Editing and Simulating Coloured Petri Nets. University of Aarhus, 2000.