

Лабораторная работа №12

Пример моделирования простого протокола передачи данных

Алиева Милена Арифовна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	17

Список иллюстраций

3.1	Декларации модели	7
3.2	Начальный граф	7
3.3	Состояние Send	8
3.4	Добавление промежуточных состояний	9
3.5	Декларации	10
3.6	Модель простого протокола передачи данных	11
3.7	Запуск модели	11

Список таблиц

1 Цель работы

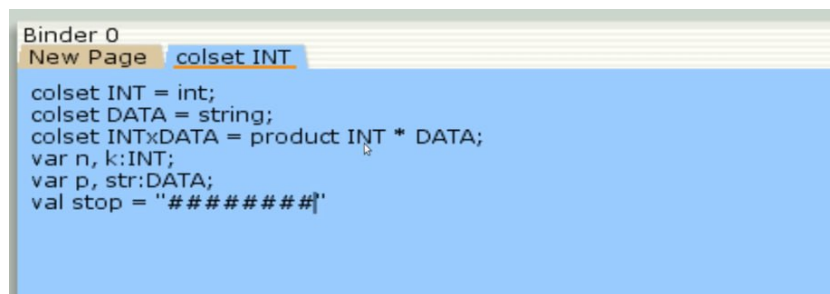
Реализовать простой протокол передачи данных в CPN Tools.

2 Задание

Реализовать простой протокол передачи данных в CPN Tools.

3 Выполнение лабораторной работы

1. Основные состояния: источник (Send), получатель (Receiver). Действия (переходы): отправить пакет (Send Packet), отправить подтверждение (Send ACK). Промежуточное состояние: следующий посылаемый пакет (NextSend).
Зададим декларации модели: (рис. 3.1)



```
Binder 0
New Page colset INT
colset INT = int;
colset DATA = string;
colset INTxDATA = product INT * DATA;
var n, k:INT;
var p, str:DATA;
val stop = "#####"
```

Рис. 3.1: Декларации модели

2. Построим начальный граф (рис. 3.2)



Рис. 3.2: Начальный граф

Состояние Send имеет тип INTxDATA и следующую начальную маркировку (в соответствии с передаваемой фразой): 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,"#####") (рис. 3.3)

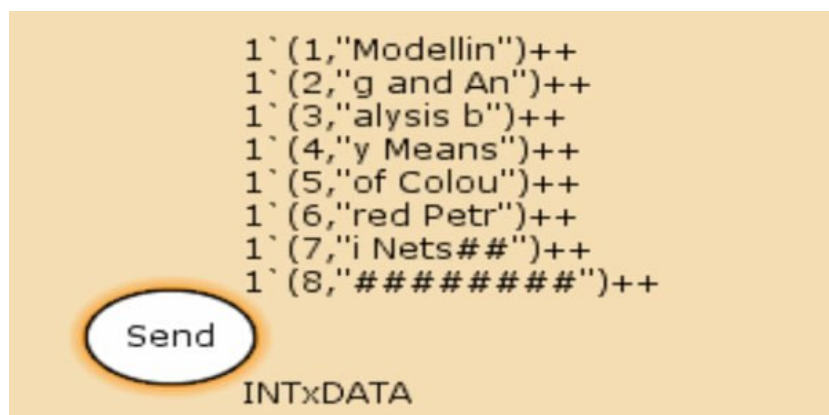


Рис. 3.3: Состояние Send

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 11. 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.

3. 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 NextRes 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 and also 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. 3.4)

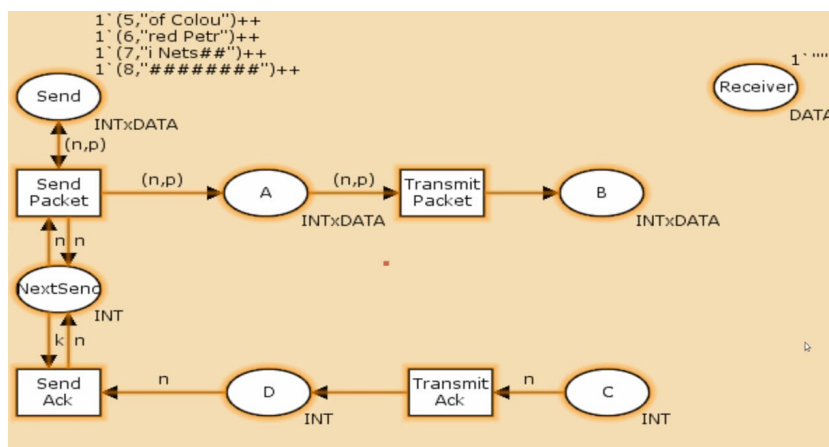



Рис. 3.4: Добавление промежуточных состояний

В декларациях задаём: (рис. 3.5)



```
▶ val stop  
▶ colset Ten0  
▶ colset Ten1  
▶ var s  
▶ var r  
▶ fun Ok
```

Рис. 3.5: Декларации

4. Таким образом, получим модель простого протокола передачи данных. Пакет последовательно проходит: состояние Send, переход Send Packet, состояние A, с некоторой вероятностью переход Transmit Packet, состояние B, попадает на переход Receive Packet, где проверяется номер пакета и если нет совпадения, то пакет направляется в состояние Received, а номер пакета передаётся последовательно в состояние C, с некоторой вероятностью в переход Transmit ACK, далее в состояние D, переход Receive ACK, состояние NextSend (увеличивая на 1 номер следующего пакета), переход Send Packet. Так продолжается до тех пор, пока не будут переданы все части сообщения. Последней будет передана стоппоследовательность. (рис. 3.6)

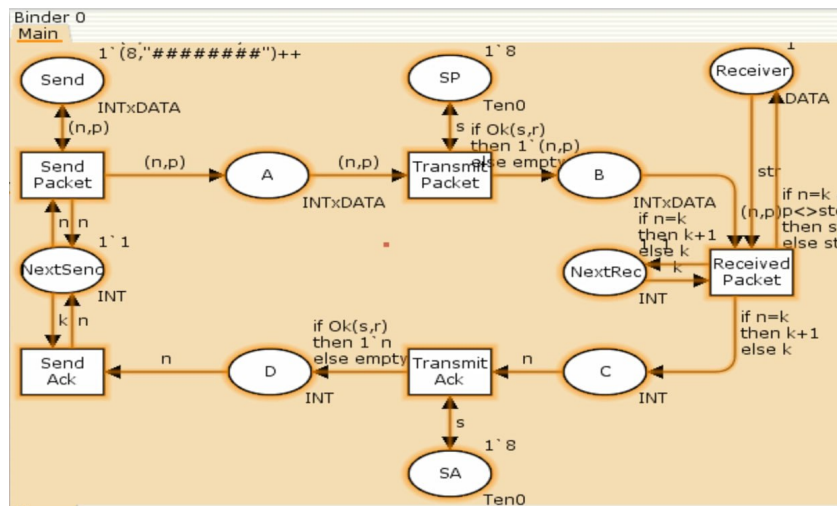


Рис. 3.6: Модель простого протокола передачи данных

5. Сохраним модель и перезапустим. Увидим, что всё отображается корректно, запустим (рис. 3.7)

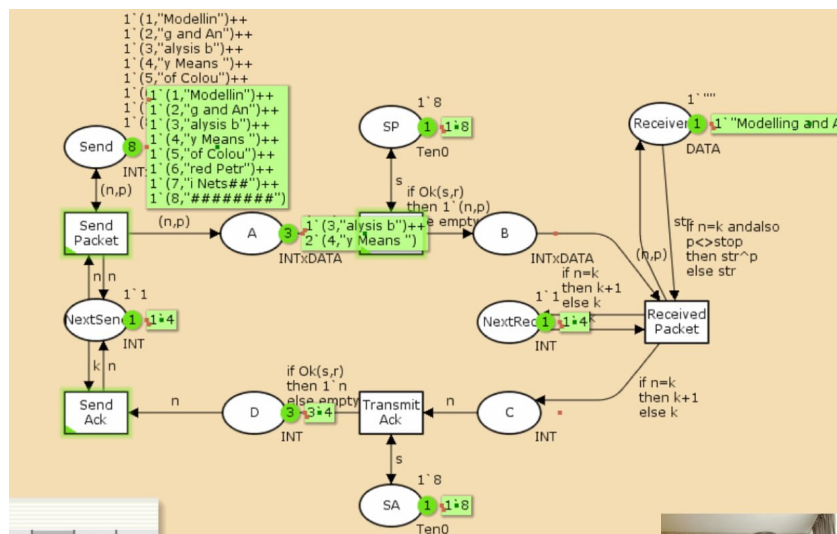


Рис. 3.7: Запуск модели

6. Отчёт о пространстве состояний:

Statistics

State Space

Nodes: 13341
Arcs: 206461
Secs: 300
Status: Partial

Scc Graph

Nodes: 6975
Arcs: 170859
Secs: 14

Boundedness Properties

Best Integer Bounds

	Upper	Lower
Main'A 1	20	0
Main'B 1	10	0
Main'C 1	6	0
Main'D 1	5	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	20`(1,"Modellin")++
15`(2,"g and An")++	
9`(3,"alysis b")++	
4`(4,"y Means ")	
Main'B 1	10`(1,"Modellin")++
7`(2,"g and An")++	
4`(3,"alysis b")++	
2`(4,"y Means ")	
Main'C 1	6`2++
5`3++	
3`4++	
1`5	
Main'D 1	5`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	
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

4675 [9999,9998,9997,9996,9995,...]

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

Видим, что у нас 13341 состояний и 206461 переходов между ними. Можем также проанализировать границы значений для промежуточных состояний A, B, C - наибольшая верхняя граница у A, затем у состояния B верхняя граница - 10. У вспомогательных состояний SP, SA, NextRec, NextSend, Receiver - 1, так как в них может находиться только один пакет, в состоянии Send - 8, так как в нем хранится только 8 элементов, как мы задавали в начале (никаких изменений с ним не происходило).

4 Выводы

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