PART 1 – JSPIN

Task 1

The request was to model a system composed by two producers, one priority queue and one consumer. So, the first thing to do was to create an mtype (Jspin representation for enum) composed by the producer's name (A or B) and the type of message. Another solution to represent the type of message to send could be the definition of a type with attribute "id" and the name of the producer, the first method is chosen for simplicity.

The model proposed can now concurrently receive or produce the messages as requested and the production mentioned above is possible only if exactly three acks are received. The resetting of variables is done using the "d_step" key word, this permits to perform all the specified operation at once but without inserting additional states.

An important request of the system was to give the priority to all the messages that the "priority queue" would receive from producer A. This condition is satisfied by checking if there are any messages in channel "pa"; if so, it atomically goes straight to the consumer. The consumer, as its last operation, will checks to whom send the acknowledgement (A or B) and so begin again the production cycle.

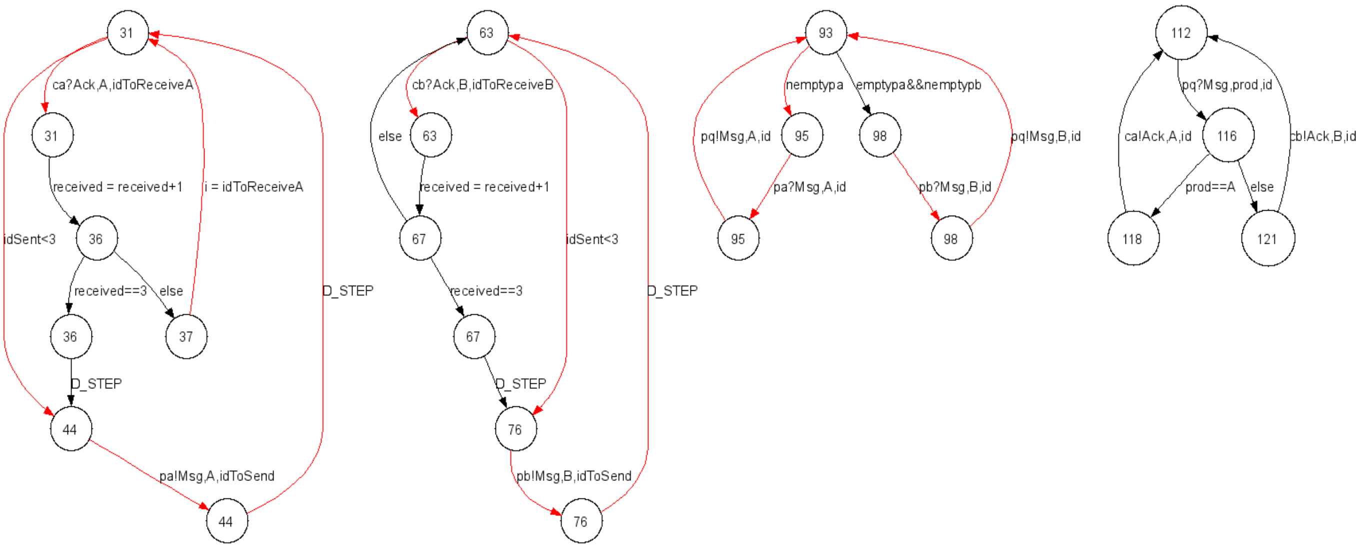
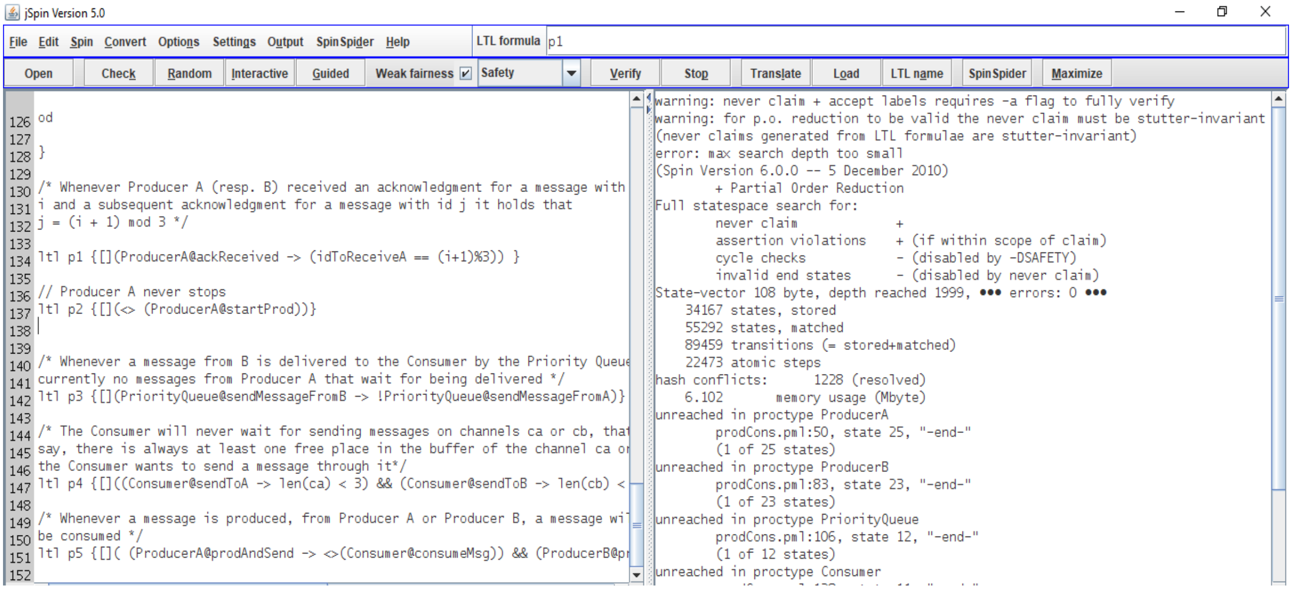


Figure 1Automata of proposed model

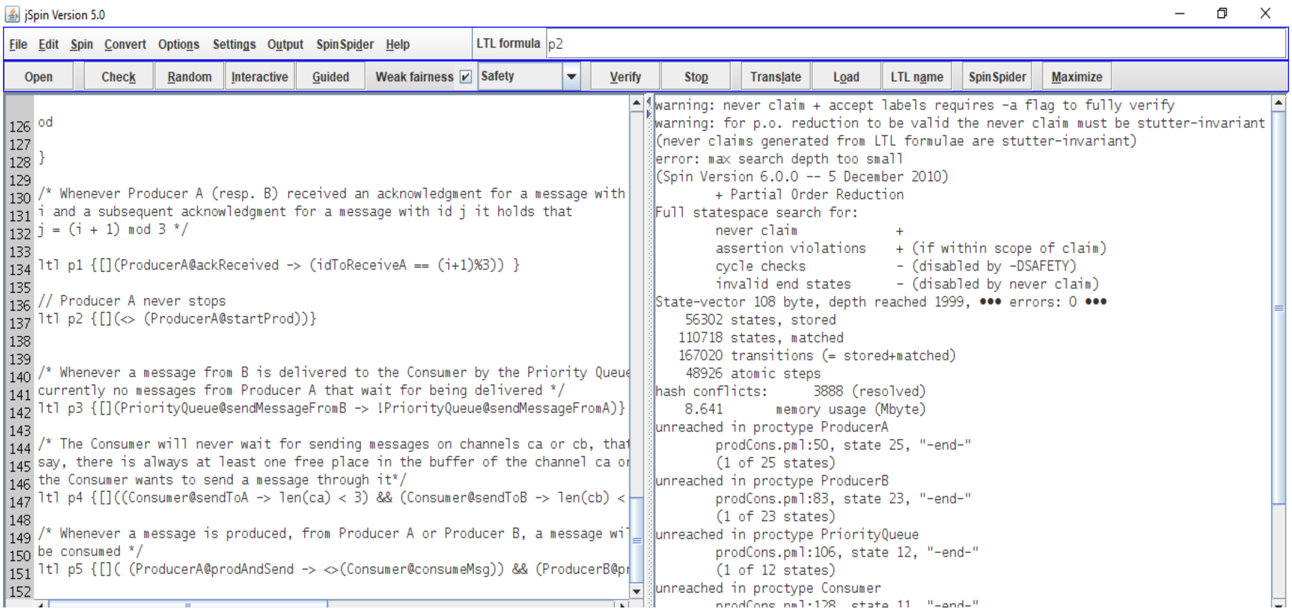
Note: ProducerA presents an additional operation on "else" condition than ProducerB because there is the updating of variable "I", useful for realization of P1 LTL condition.

Task 2

- P1: The first property wants to know if the messages received by the producers respect the specified order (0,1,2). The variable “i” represents the previous message received. This is a safety property.

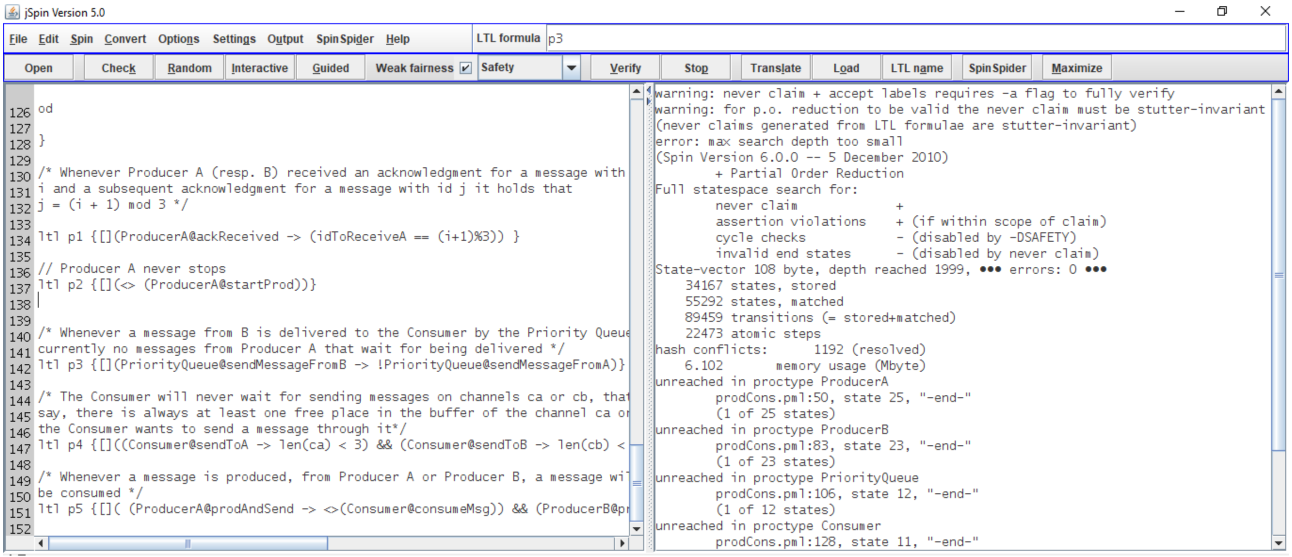


- P2: Liveness property. The property expressed specify that the task “startProd” will be reached infinitely often.

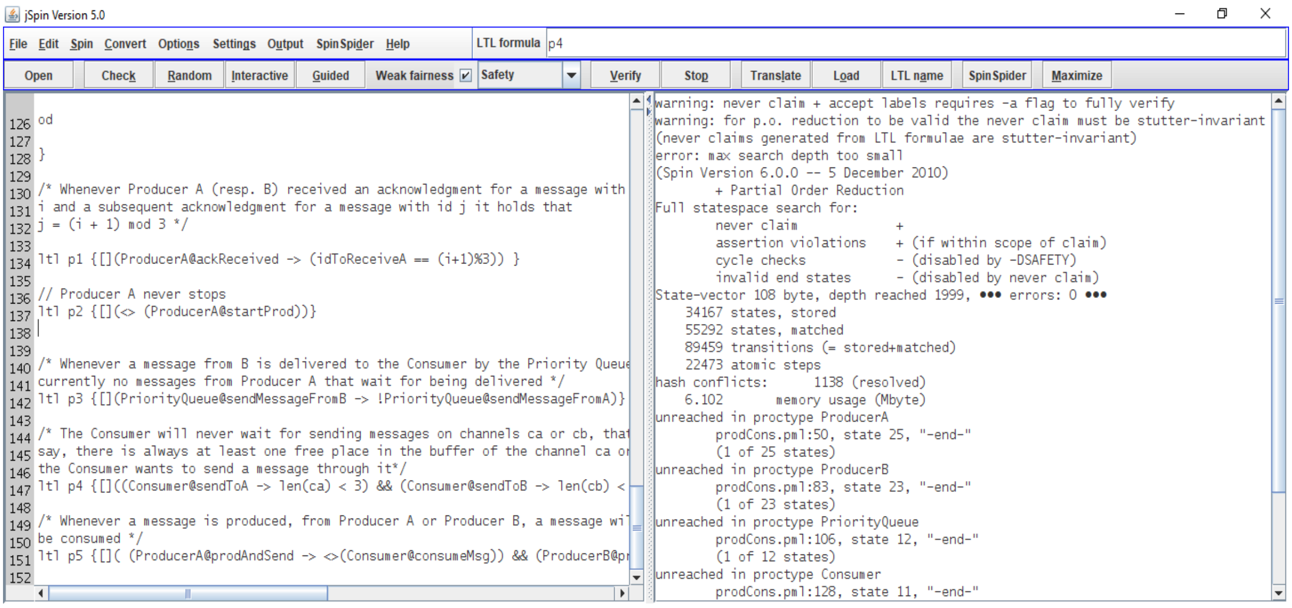


-

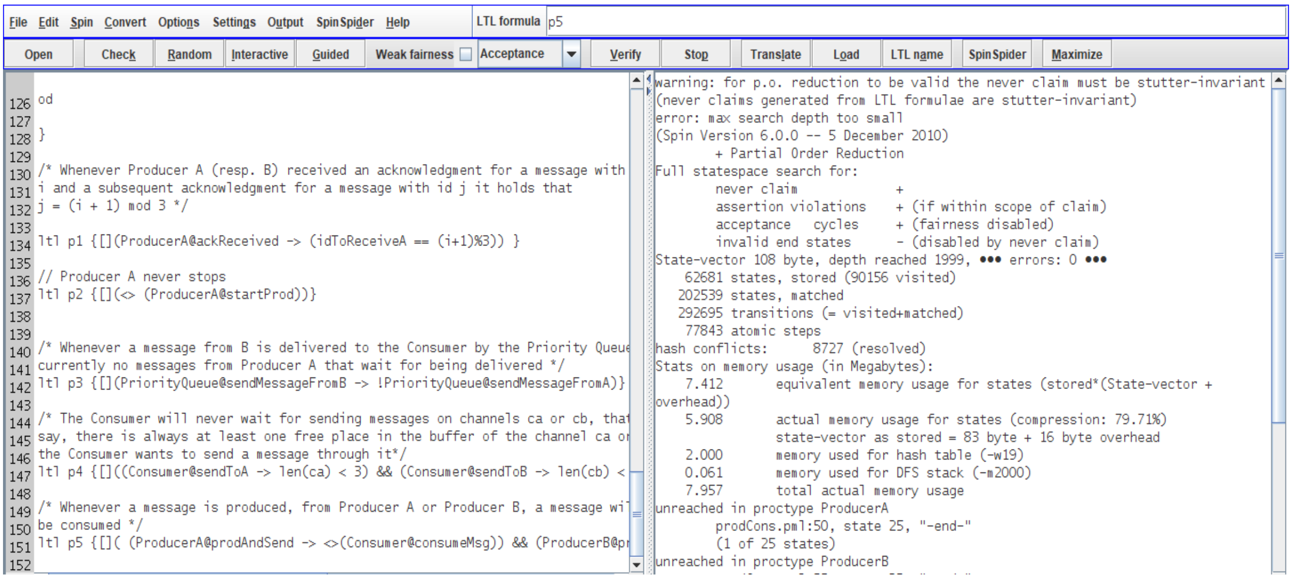
- P3: Safety property. The property checks that the “PriorityQueue” has not reached the state in which it can forwards messages from A if it can send messages from B instead.



- P4: Safety property. Whenever the consumer wants to send a message through the channel “ca” or “cb”, it will never wait because the size of these channels is exactly 3.



- P5: Liveness property. The property expressed says that if producer A (resp. B) has reached state “prodAndSend”, sooner or later will reach the state in which the messages are consumed (“consumeMsg”).



Part 2

Task 1

The model proposed for the simplified version of the previous system has one template for each component of the system itself: Producer, Consumer, Queue.

Producer and Consumer present similar behaviour; since both production and consumption have to start after producing and consuming a message, the time is reset after every transition. The Producer is composed by two states in which are distinguishable the starting of a production and its continuation; the first state manages the sending only if the id of the message is set to 0, so it starts or restart to produce again. The second state continue to produce until it reaches the maximum number of messages, when this happens it goes to the starting state. The Consumer has the same behaviour but with the exception that the transitions are activated only when receiving from the Queue.

The Queue has only one state, this is because it is always ready to receive or to send a message. To control the incoming messages a variable “idToReceive” has been added, so every time it synchronizes with the Producer this variable is incremented or reset inside method “enqueue”.

Task 2

Question 1 (Check that the system is deadlock free).

The system is deadlock free.

Question 2(Determine whether or not it is possible that the Queue becomes full).

The queue cannot become full because it can immediately free the messages, so the condition is not satisfied.

Question 3(Determine whether or not in every run sooner or later the Queue will become full).

Same as above.

Question 4(Determine whether or not the Producer is alive, i.e., whenever it produces a message, it will eventually produce another message).

This property is satisfied, since after every production another will begin.

Question 5(Now change the capacity N to 2 and repeat the checks of the previous 4 points. Is there any difference? Explain why).

Changing the capacity of the queue will not affect the results since the queue will free immediately the messages, even because the intersection between t1 and t2 is not empty.

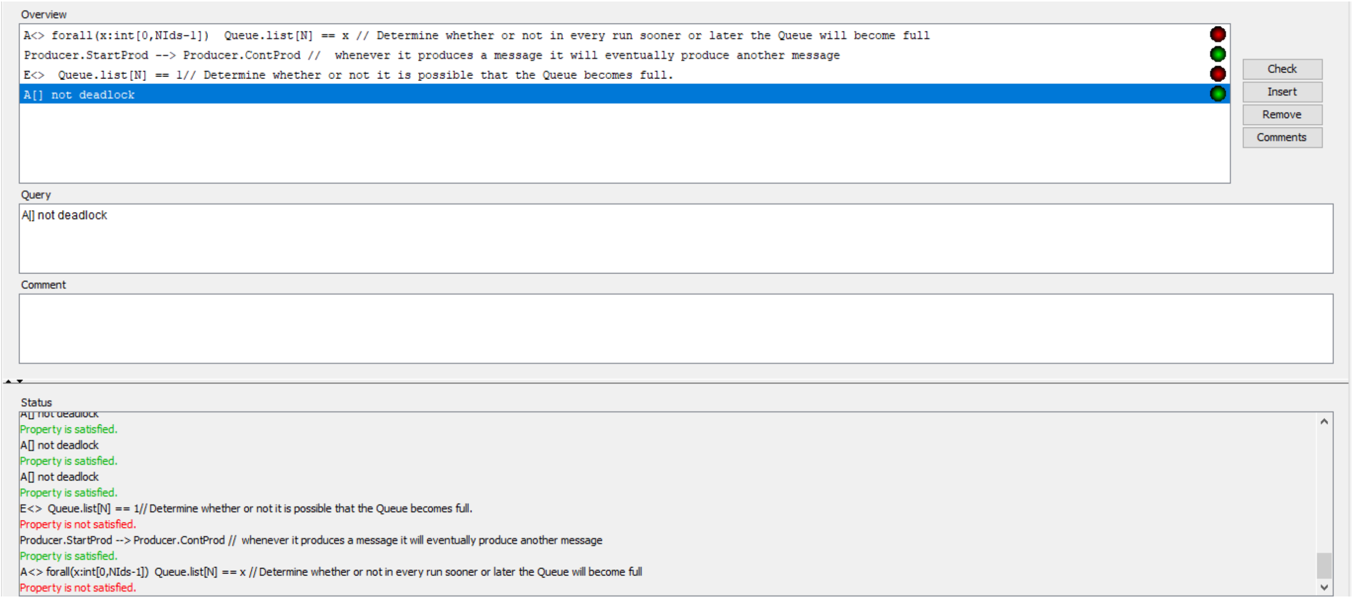


Figure 2 - Results from requested conditions

Question 6(Now change the time interval for the Consumer from [2, 3] to [3, 4]and check that the system can reach a deadlock situation. Why is it so? Is it possible to find a value for the capacity of the Queue in order to avoid the deadlock? Justify your answers).

The system can actually reach a deadlock situation. This is because when entering the queue, it has to wait for the Consumer to be into [3,4] range; so, the producer that is already producing another message waits too long (since the queue becomes full in this case) and cannot perform actions anymore.

Finally, is not possible to avoid the deadlock changing the capacity of the queue since sooner or later the Producer will wait for the Queue to become free.

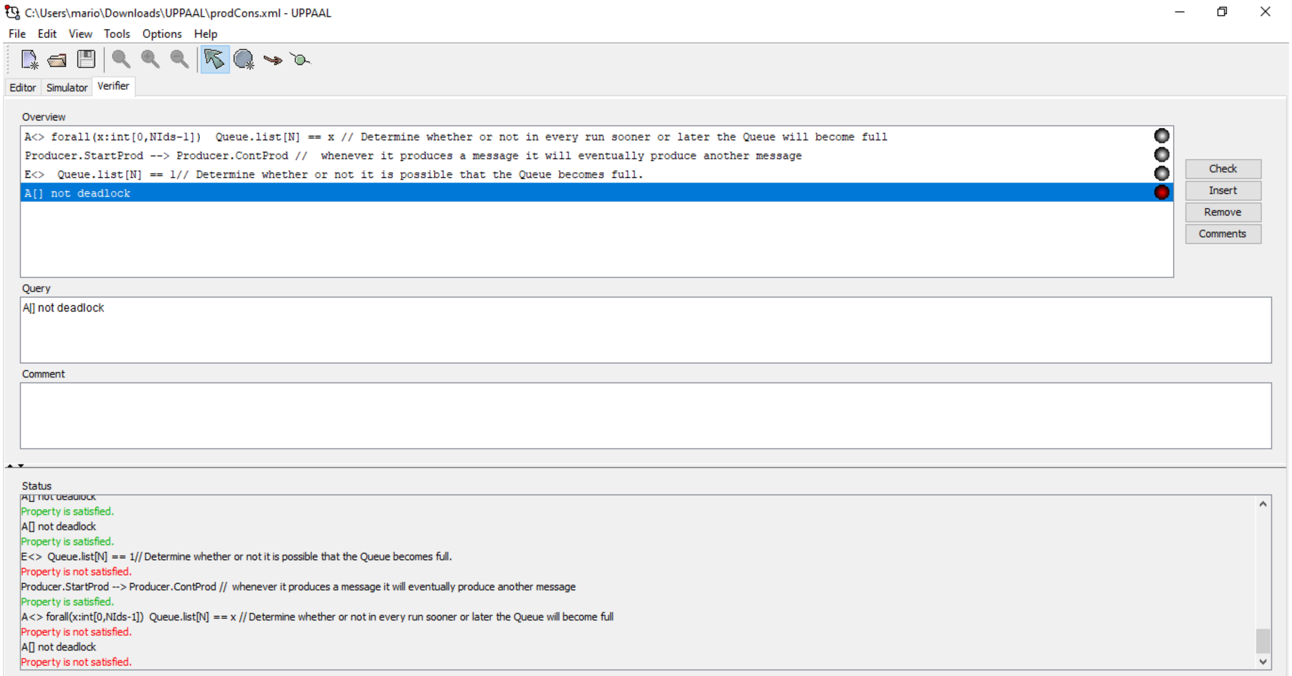


Figure 3 - Changing the consumption time the system is not deadlock free.