CSSE373 Formal Methods Milestone 3

By: Morgan Cook, Lucas Miller & Alia Robinson

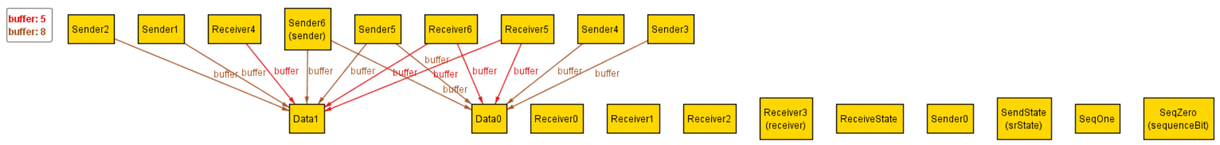
RDT2.1

We successfully modeled RDT2.1.

Property 1: There is at least one way for all data to be transferred.

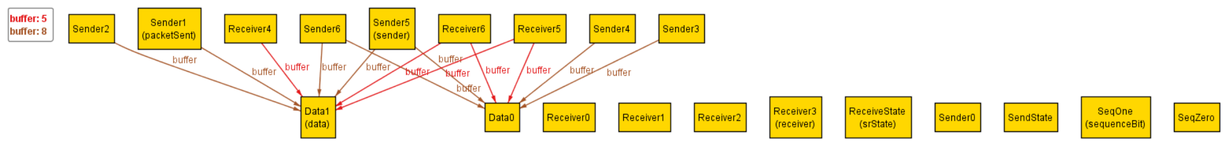
Our model was too big with this many states, so the following images are projected over both state and the packet that state is associated with.

State 0:



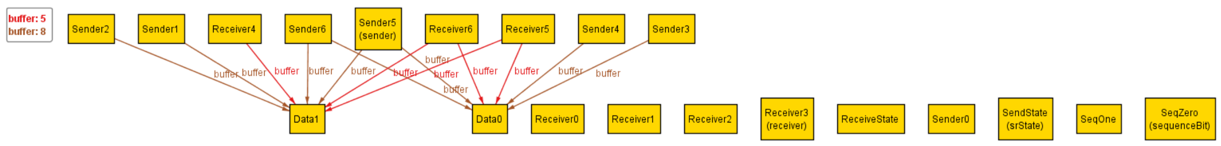
We are in the Send State. The current packet is AckPacket1 which has SeqZero.

State 1:



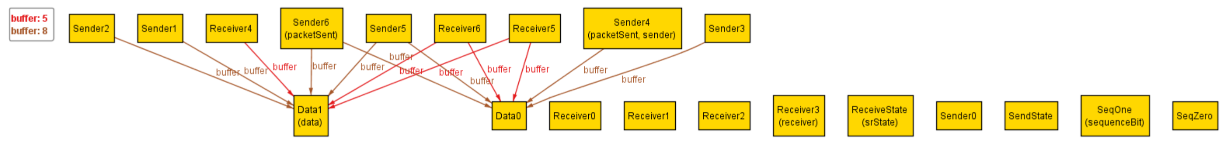
We are in the Receive State. The current packet is DataPacket3 which has SeqOne.

State 2:



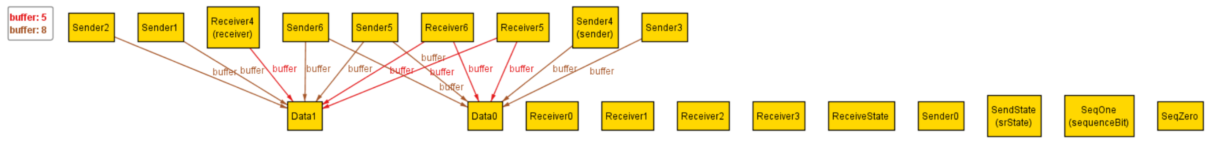
We are in the Receive State. The current packet is AckPacket1 which has SeqZero, which is incorrect because it is different from the seq bit of the data we just sent.

State 3:



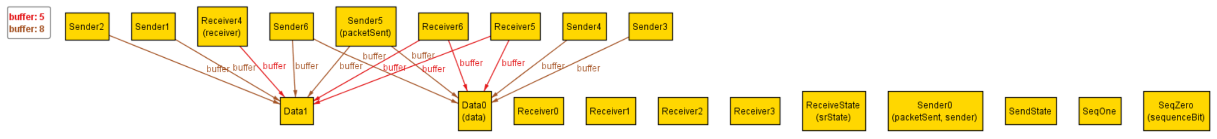
We are in the send state. The current packet is DataPacket0 which has SeqOne. Also, the receiver did not get the data because the AckPacket's sequence was different from the sent packet's sequence.

State 4:



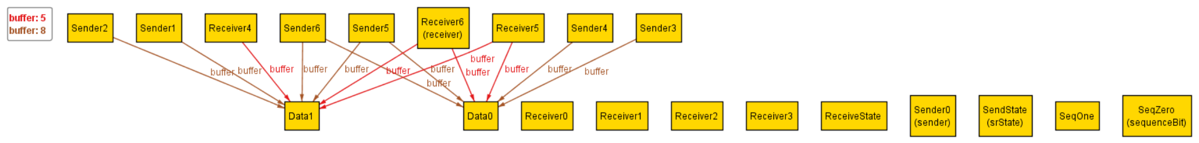
We are in the receive state. The current packet is AckPacket0 which has SeqOne which is correct because it is the same seq bit as the data just sent. The receiver now contains Data1 because it was successfully sent and received.

State 5:



We are in the send state. The current packet is DataPacket4 which has SeqZero.

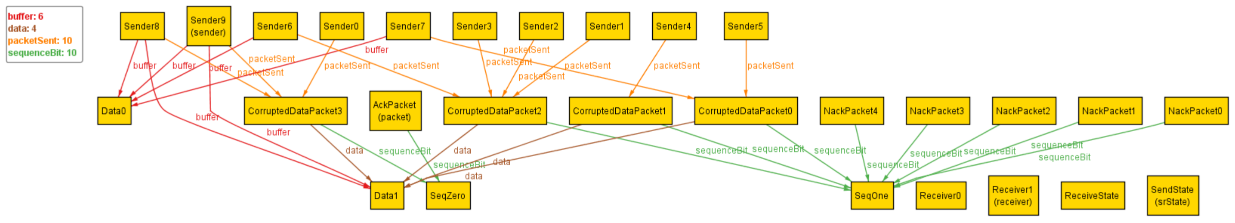
State 6:



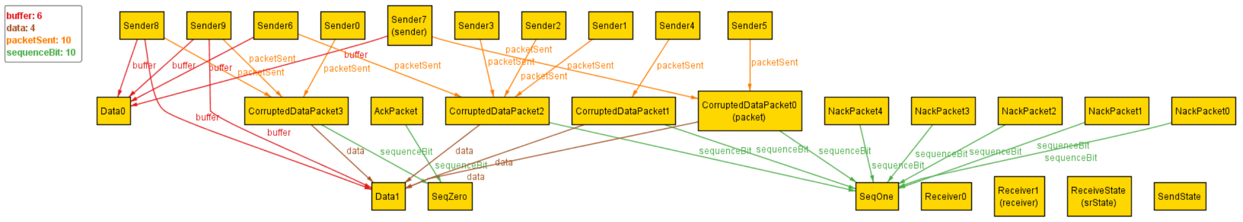
We are in the receive state. The current packet is AckPacket1 which has SeqZero. The receiver now contains both Data because both were successfully sent and received. This is the end state.

Property 2: It is not always possible for all data to be transferred

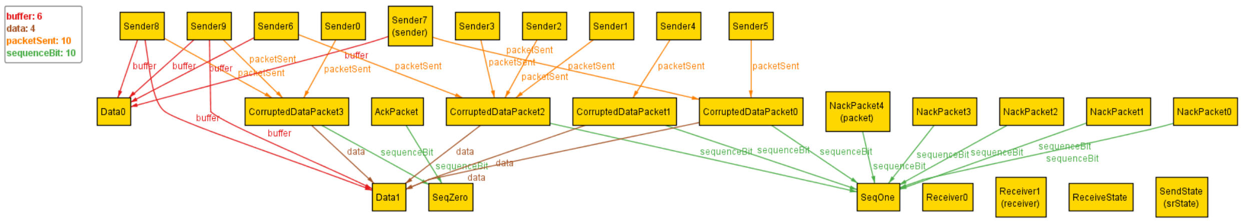
Alloy finds a counterexample, similar to the counterexample found for RDT2.0.



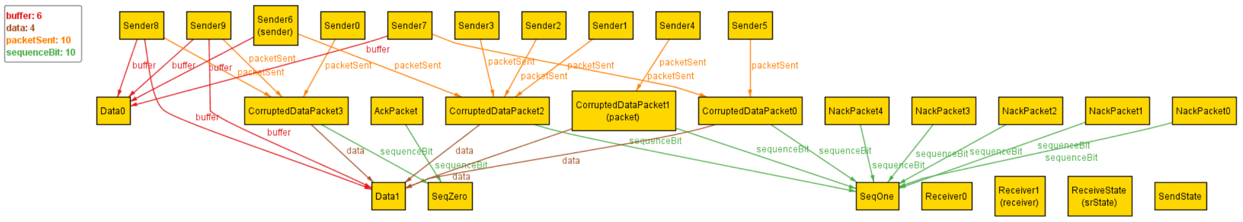
State 0: The initial state.



State 1: This is the send state. The sender sends a corrupted data packet holding Data1.



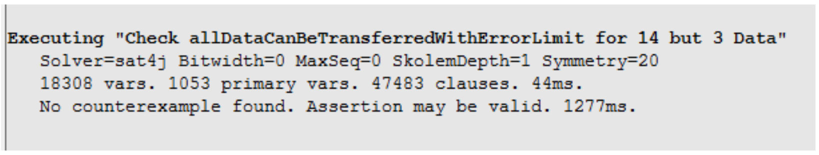
State 2: This is the receive state. The receiver sends back a NAK packet to indicate that the data was corrupted.



State 3: This is the send state. The sender sends another corrupted data packet holding Data1.

After the states shown here, the cycle of sending corrupted packets and responding with NAK packets can continue infinitely. Therefore, it is possible for some of the data to never reach the receiver when using this protocol.

Extra property: It is always possible to send all data from the sender buffer to the receiver buffer, given that there can be no more than one send/receive error in the wire.



```
Executing "Check allDataCanBeTransferredWithErrorLimit for 14 but 3 Data"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   18308 vars. 1053 primary vars. 47483 clauses. 44ms.
   No counterexample found. Assertion may be valid. 1277ms.
```

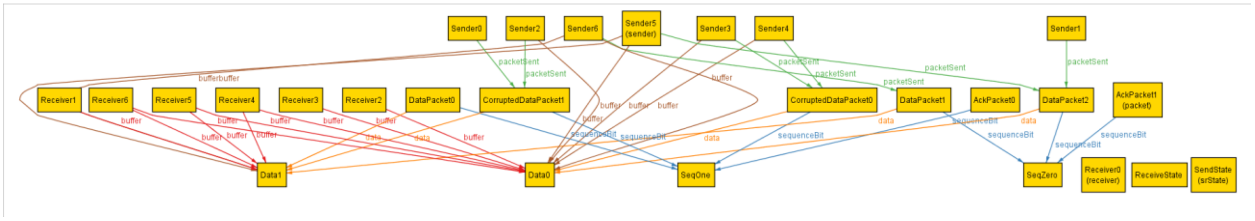
When the constraint is added that there can be no more than one error for each data packet, Alloy finds no counterexample.

RDT2.2

We successfully modeled RDT2.2.

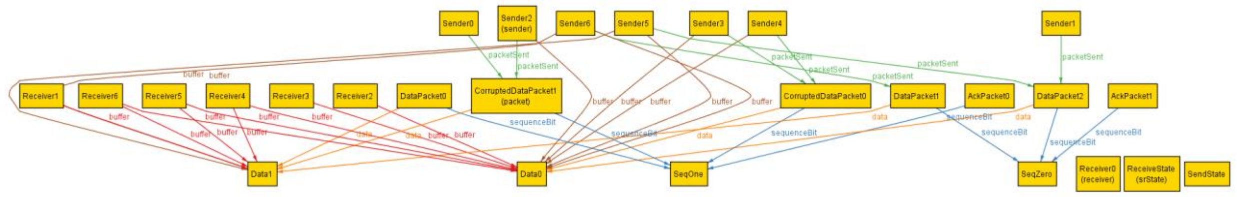
Property 1: There is at least one way for all data to be transferred.

State 0:



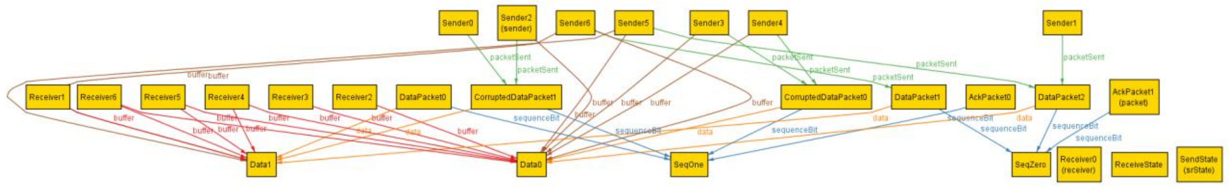
We are currently in the send state. The current packet is AckPacket which has SeqZero.

State 1:



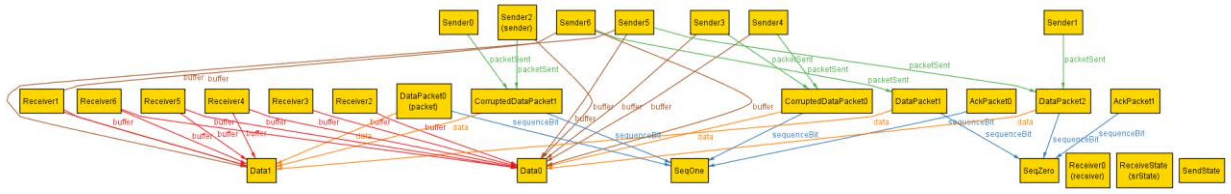
We are currently in the receive state. The current packet is CorruptedDataPacket which has SeqOne.

State 2:



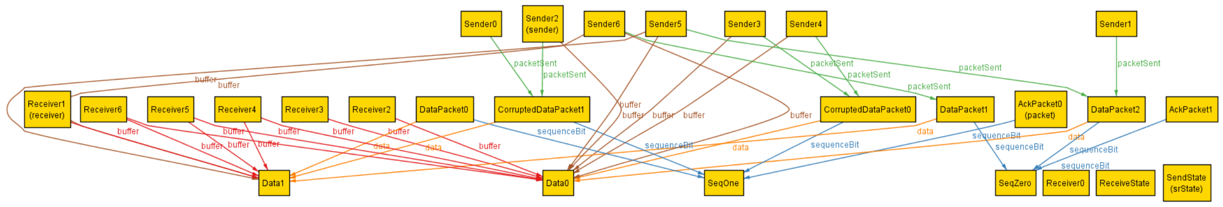
We are currently in the send state. The current packet is AckPacket1 which has SeqZero. This means that the data was not sent correctly. Therefore, the data is not held by the receiver.

State 3:



We are currently in the receive state. The current packet is DataPacket0 which has SeqOne.

State 4:

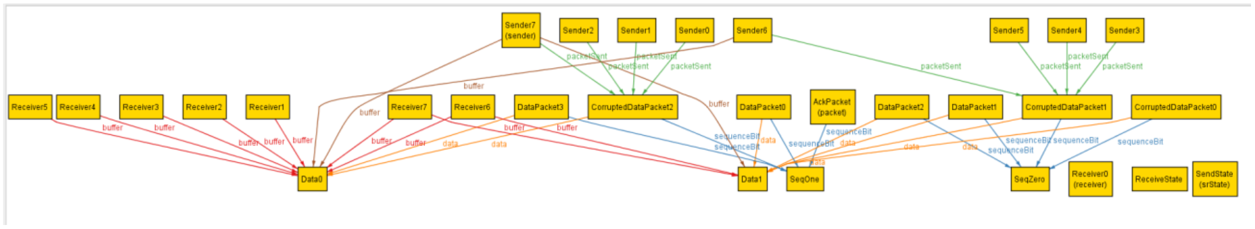


We are currently in the send state. The current packet is AckPacket0 which has SeqOne. Because the Ack is sending the correct sequence bit, the data was sent and received correctly; all data is now held in the receiver's buffer.

Property 2: It is not always possible for all data to be transferred

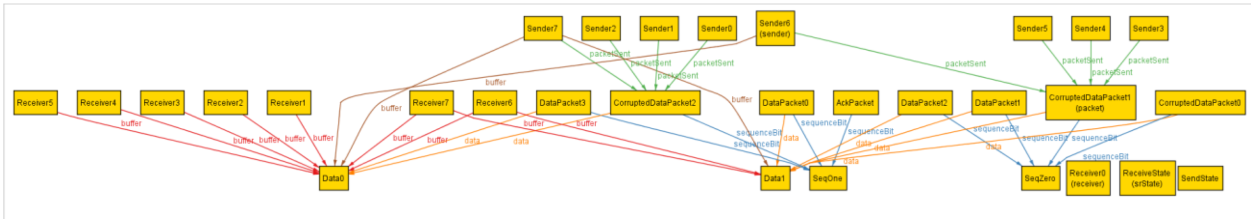
Alloy finds a counterexample, similar to the counterexample found for RDT2.1.

State 0:



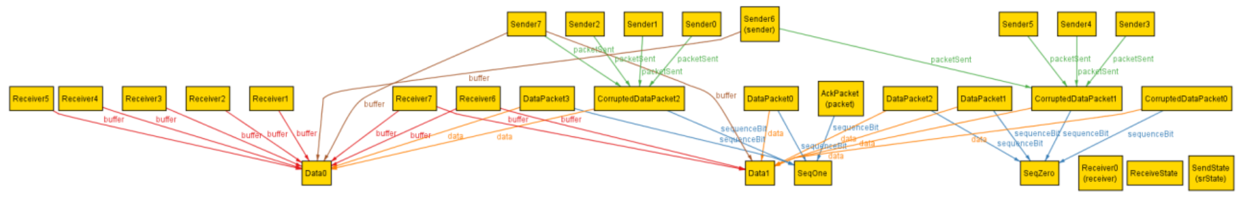
We start in the send state. The current packet is AckPacket which has SeqOne.

State 1:



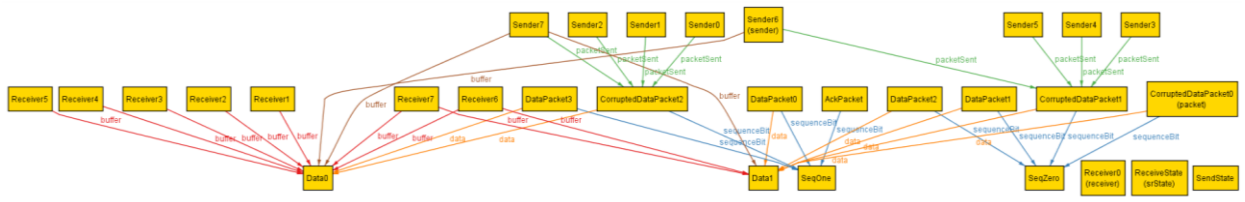
We are in the receive state. The current state is CorruptedDataPacket which has SeqZero.

State 2:



We are in the send state. The current packet is AckPacket which has SeqOne. Because this packet has a different sequence from the last sent packet, the data transfer failed and the data is not in the receiver's buffer.

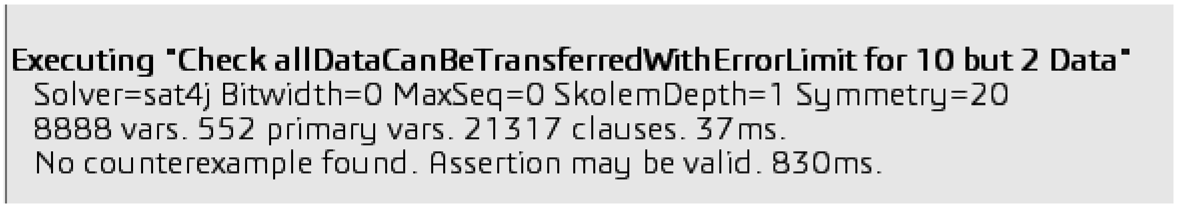
State 3:



We are in the receive state. The current packet is CorruptedDataPacket0 which has SeqZero.

Therefore, this property does not hold because this process of constantly sending corrupted data can continue infinitely.

Extra property: It is always possible to send all data from the sender buffer to the receiver buffer, given that there can be no more than one send/receive error in the wire.



```
Executing "Check allDataCanBeTransferredWithErrorLimit for 10 but 2 Data"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   8888 vars. 552 primary vars. 21317 clauses. 37ms.
   No counterexample found. Assertion may be valid. 830ms.
```

When the constraint is added that there can be no more than one error for each data packet, Alloy finds no counterexample.