# Ben-Or consensus algorithm

## Properties explanation

1) **Agreement:** This property states that once all the processes have decided a value, then that value should be same for all of them. In our implementation, the agreement property checks for any two processes (suppose j & k) and if they decided for some values for these two processes then those decided values should be same in order to hold the agreement property true.

2) **Progress:** This property states that eventually all processes end up deciding some final value and after that the program would terminate. In our defined properties, progress property is defined as eventually all the processes should have the number of values of the decided set greater than 0. Based on our observation, we see that in case of same preference value for all the nodes, the progress property holds true and termination is achieved.

3) **MinorityReport:** This property states that if the initial input has a minority value 0 for a set of nodes, then it becomes impossible for the nodes to decide 0 eventually. But, in case of more faulty nodes, it becomes possible to decide the value 0 and that in turn makes the value 0 as not minority in the nodes anymore. In our defined properties, MinorityReport is defined as, for all the nodes in the processes, none of the decided values should be 0.
   e.g. For the given initial input <<0,1,1,1>>, Minority value is 0. So, if model checker suggests that if there is any possible states where all nodes decide on a value 0, then in that case the MinorityReport property gets violated i.e., for some reason, model will decide on 0 even if it is not in majority.

# Ben-Or consensus algorithm

**Observations:**

| TASK 1 | | | | | | |
|---|---|---|---|---|---|---|
| Scenario | F | N | INPUT | MAXROUND | Agreement | Progress |
| Scenario 1 | 0 | 4 | <1,1,1,1> | 4 | TRUE | TRUE |
| Scenario 2 | 0 | 4 | <0,0,0,0> | 4 | TRUE | TRUE |
| Scenario 3 | 0 | 4 | <0,0,1,1> | 4 | TRUE | FALSE (Temporal property violated) |
| Scenario 4 | 1 | 4 | <1,1,1,1> | 4 | TRUE | TRUE |
| Scenario 5 | 1 | 3 | <0,1,1> | 4 | TRUE | FALSE (Temporal property violated) |
| **TASK 2** | | | | | | |
| Scenario | F | N | INPUT | MAXROUND | Agreement | MinorityReport Violation |
| Scenario 1 | 0 | 4 | <0,1,1,1> | 4 | TRUE | Not violated |
| Scenario 2 | 1 | 4 | <0,1,1,1> | 4 | TRUE | Violated |

## TASK 1

## Scenarios explanation:

### Task 1: Scenario 1

**Agreement:** This property holds true for the given scenario. As agreement property states that for any two processes j & k, their decided values should be same. So, in our defined properties, the agreement property states that the cardinality(number of value) of both decided[j] and decided[k] should be 1 i.e. all the all the processes have decided for the same value.

**Progress:** This property holds true for the given scenario as eventually all the processes have ended up deciding some final value and after that the program have been terminated.

### Task 1: Scenario 2

**Agreement:** This property holds true for the given scenario. As agreement property states that for any two processes j & k, their decided values should be same. So, in our defined properties, the agreement property states that the cardinality(number of value) of both decided[j] and decided[k] should be 1 i.e. all the all the processes have decided for the same value.

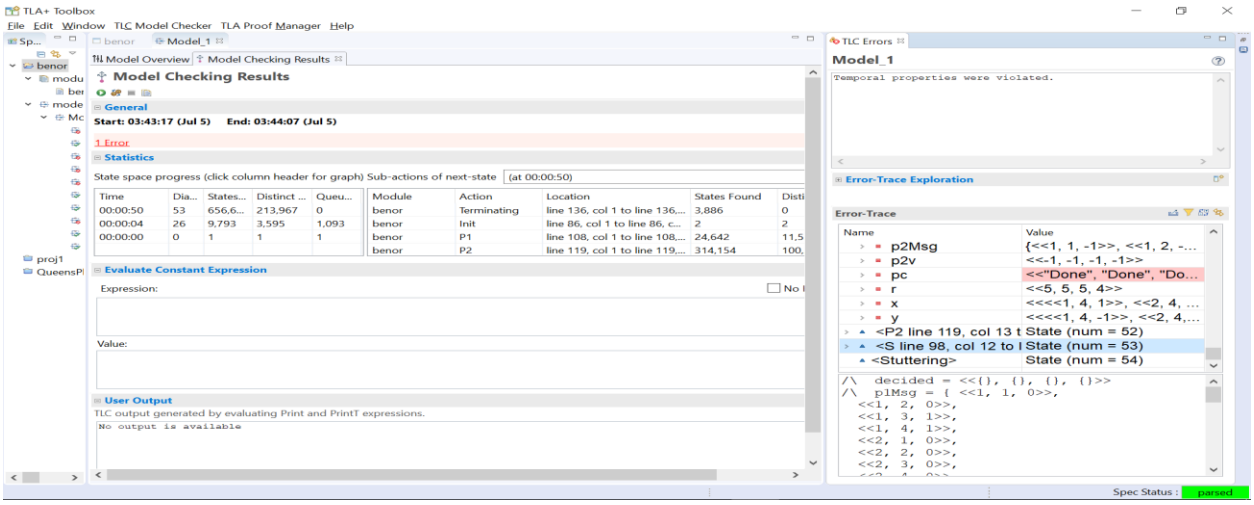
**Progress:** This property holds true for the given scenario as eventually all the processes have ended up deciding some final value and after that the program have been terminated.

# Ben-Or consensus algorithm

## Task 1: Scenario 3

**Agreement:** This property holds true for the given scenario. As agreement property states that for any two processes j & k, their decided values should be same. So, in our defined properties, the agreement property states that the cardinality(number of value) of both decided[j] and decided[k] should be 1 i.e. all the all the processes have decided for the same value.

**Progress:** Temporal property (Progress) is violated for this scenario. Since the initial input is <0,0,1,1>, the model checker finds any states where deciding on the same value becomes impossible for the processes and thus, they do not complete the progress property and does not terminate successfully.




## Task 1: Scenario 4

**Agreement:** This property holds true for the given scenario even in case of one failed node. As agreement property states that for any two processes j & k, their decided values should be same. So, in our defined properties, the agreement property states that the cardinality of both decided[j] and decided[k] should be 1 i.e. all the all the processes have decided for the same value and it works fine even in case of one failed node.

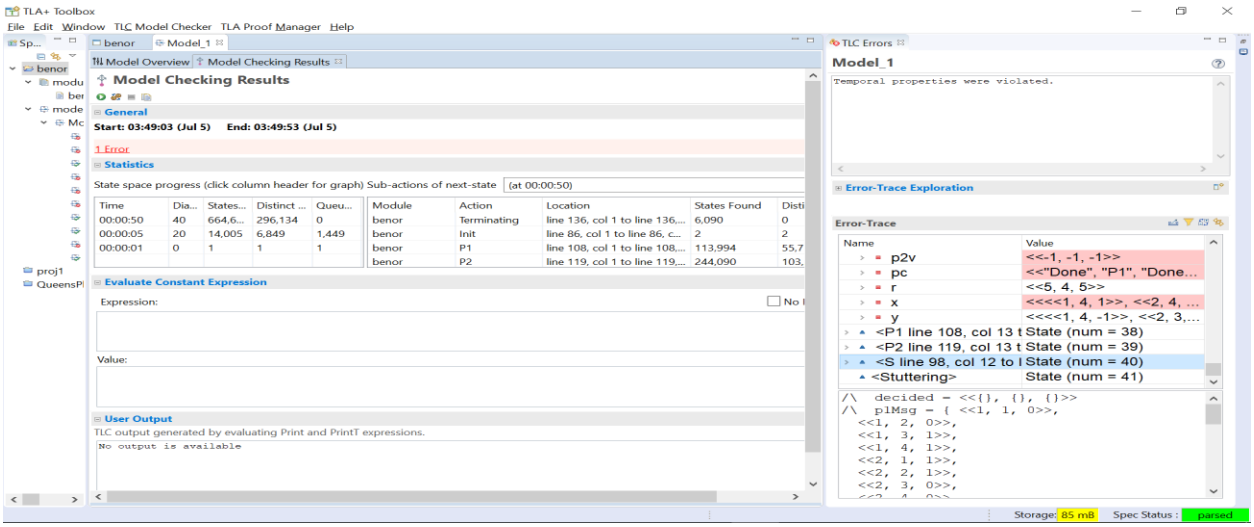
**Progress:** This property holds true for the given scenario as eventually all the processes have ended up deciding some final value and after that the program have been terminated.

# Ben-Or consensus algorithm

## Task 1: Scenario 5

**Agreement:** This property holds true for the given scenario even in case of one failed node. As agreement property states that for any two processes j & k, their decided values should be same. So, in our defined properties, the agreement property states that the cardinality of both decided[j] and decided[k] should be 1 i.e. all the all the processes have decided for the same value and it works fine even in case of one failed node.

**Progress:** Temporal property (Progress) is violated for this scenario. Since the initial input is <0,1,1>, the model checker finds any state where deciding on the same value becomes impossible after for the processes once a node is failed and thus, they do not complete the progress property and does not terminate successfully.




# TASK 2

## Task 2: Scenario 1

**Agreement:** This property holds true for the given scenario. As agreement property states that for any two processes j & k, their decided values should be same. So, in our defined properties, the agreement property states that the cardinality of both decided[j] and decided[k] should be 1 i.e. all the all the processes have decided for the same value.

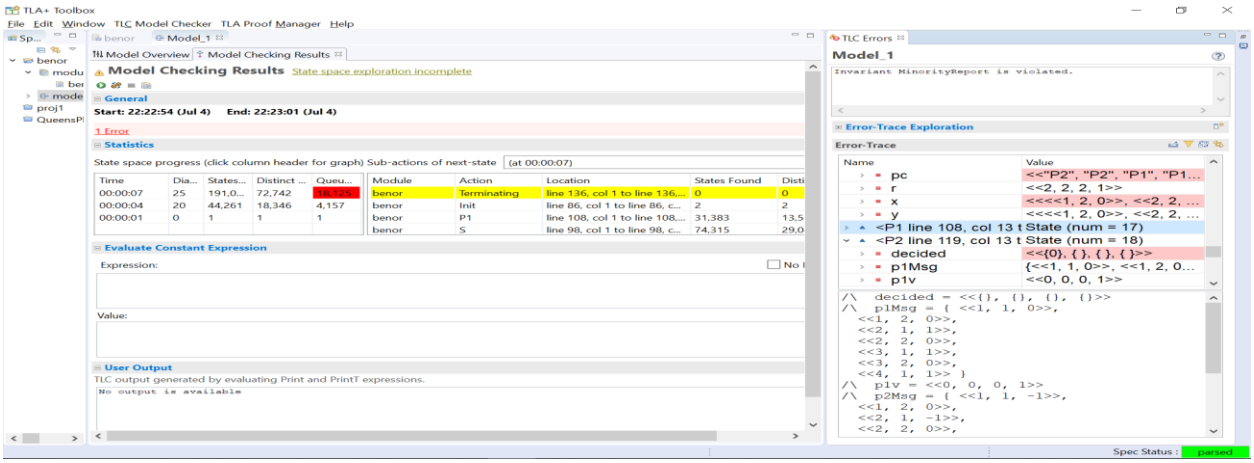
**MinorityReport:** For the given scenario and values, the MinorityReport property is violated. In state 17, we see that there is no decision with 0. But, when we move to state 18, it seems that one of decided value is 0 in this case and eventually all the nodes have decided value 0 in state 25. Thus, it violates our defined MinorityReport property where it says that none of the nodes should have decided values as 0.

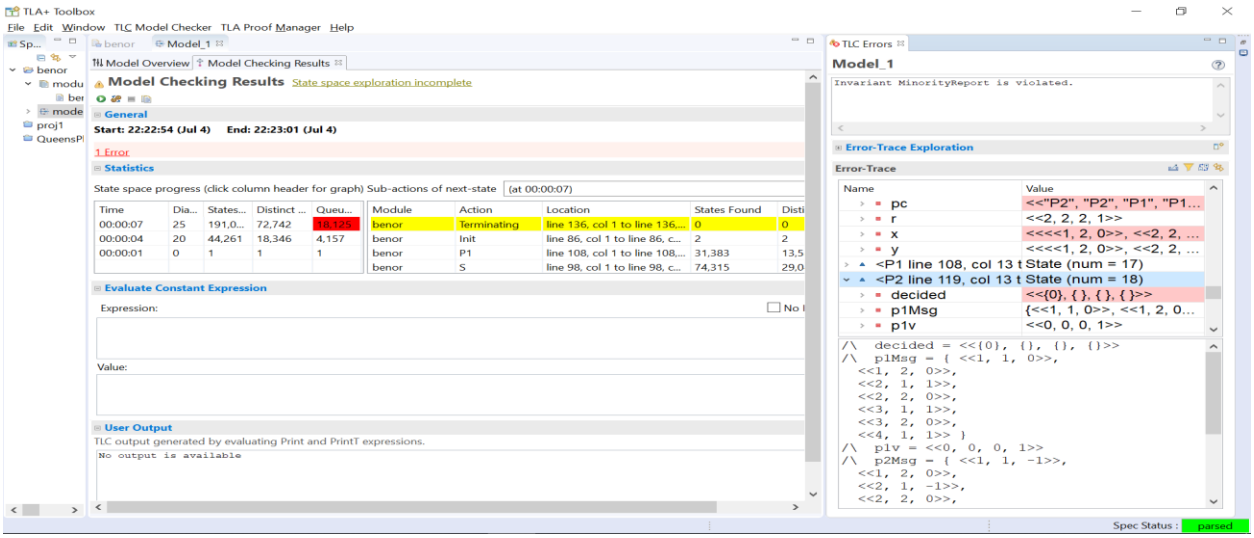
# Ben-Or consensus algorithm

## Task 2: Scenario 2

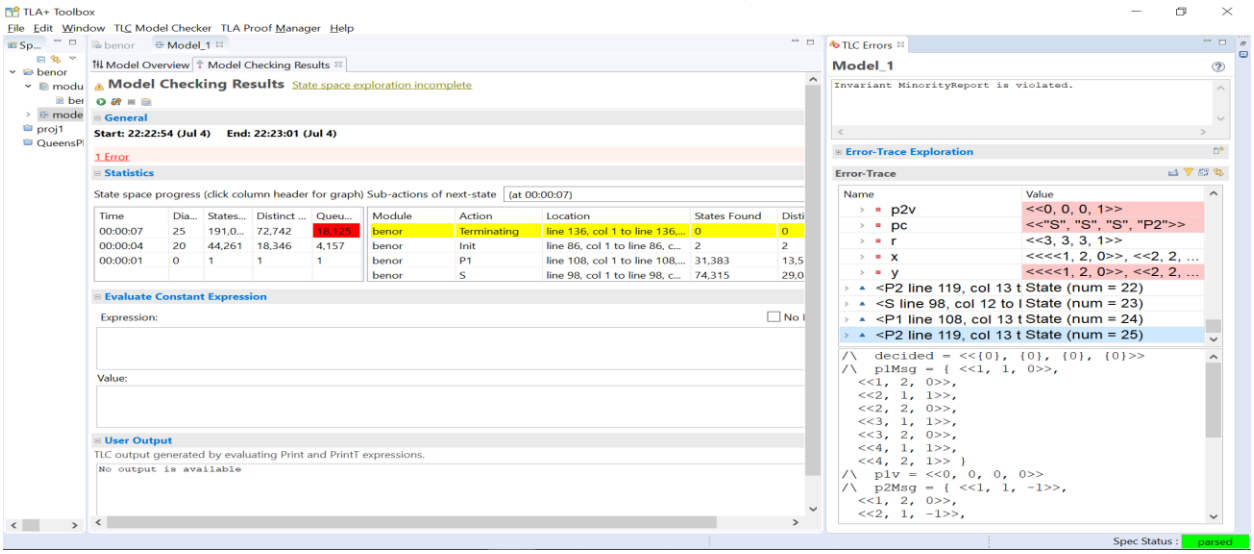
**Agreement:** This property holds true for the given scenario even in case of one failed node. As agreement property states that for any two processes j & k, their decided values should be same. So, in our defined properties, the agreement property states that the cardinality of both decided[j] and decided[k] should be 1 i.e. all the all the processes have decided for the same value and it works fine even in case of one failed node.

**MinorityReport:** For the given scenario and values, the MinorityReport property is violated. Due to faulty node, model checker keeps assigning random values to the nodes to achieve majority value. In state 17, we see that there is no decision with 0 yet. But, when we move to state 18, it seems that one of decided value is 0 in this case and eventually all the nodes have decided value 0 in state 25 as majority value. Thus, it violates our defined MinorityReport property where it says that none of the nodes should have decided values as 0 since value 0 was minority. Even if one of the nodes failed, the other nodes have decided on the value 0.



# Ben-Or consensus algorithm




**Final Observation:**

Based on our observations, we see that the agreement property never got violated for the given scenarios even in case of failure of nodes. The remaining nodes have decided on some final value to be in agreement with others and complete the program.

## BenOr Algorithm Conclusion:

Ben-Or algorithm is a randomized consensus algorithm having a binary protocol method in order to come to a conclusion to whether commit or abort. It was a first protocol to achieve consensus with probabilistic termination in a model and always guaranteed to work. It can tolerate up to f<n/2 crash failures.

Ben-Or algorithm is basically a leaderless algorithm as every replica sends their values to all other replicas. It has 2 phases in every round and works similarly as Paxos. For phase 1, replicas send their values to all other replicas and then wait for all non-faulty replicas to send their messages. After they receive messages from all non-faulty processes (n-f => f is the number of faulty replicas), they check for a value which was proposed by the majority (n/2 => n is the number of replicas). If majority has proposed any value, then replica proposes the same value for the phase 2. Otherwise, it proposes an undefined value. For the second phase, again replicas wait for all non-fault replicas. Then they check the messages from at least f+1 process and see if they are proposing the same value. In case of same proposed value, then replica goes with that value. In case if there is any value, it proposes the same value for the first phase of next round by checking that the proposed value is not undefined. A random value either 0 or 1 is being proposed for the next round in case if there was no value.

This protocol is not very efficient and in particular the expected number of rounds to reach agreement may be exponential and thus require more computational power.