# Valid states



# State transitions



close() call

Bid() call

newAuction() call

Bid() call

close() call

1. **Inactive to Open**

*stateBefore*

*f(e,args)*

*stateAfter*

*assert(stateBefore==INACTIVE && stateAfter==OPEN =>f.selector==newAuction(..)*

*assert(stateBefore==INACTIVE => stateAfter==OPEN || stateAfter==INACTIVE)*

1. **Open state transition**

stateBefore

f(e,args)

stateAfter

assert(stateBefore==OPEN => stateAfter==OPEN || stateAfter==BID\_PLACED || stateAfter==INACTIVE)

assert(stateBefore==OPEN && stateAfter==BID\_PALCED => f.selector==bid(..))

assert(stateBefore==OPEN && stateAfter==INACTIVE => f.selector==close())

1. **BID\_PLACED state transition**

stateBefore

f(e,args)

stateAfter

assert(stateBefore==BID\_PLACED=> stateAfter==BID\_PLACED || stateAfter==INACTIVE)

assert(stateBefore==BID\_PLACED && stateAfter==INACTIVE => f.selector==close(..)

1. **Valid State transition**

stateBefore

f(e,args)

stateAfter

assert(stateBefore is valid => stateAfter is valid)

# Variable transitions

* 1. Total Supply cannot decrease
  2. Prize can only decrease
  3. Payment remains constant throughout the life of a auction

# High-Level properties

* 1. Total Supply>=User Balance
  2. Sum of balances <= total supply
  3. Owner!=0
  4. Sum of all open auction prizes + totalSupply <= MAX\_UINT256

*If the intention is to make it possible to close all open auctions to be closed then the above property would be needed, else we’d end up with potentially multiple open auctions which would never be closed since there is no burn function in the contract.*

# Unit tests

* 1. **Mint()** should increase the total supply and mint account balance by the minting amount
  2. **transferTo()** should increase balance[to] and decrease balance[msg.sender] by transfer amount and sum of all balances should remain less than total supply.
  3. **newAuction()**

require(auctionStateBefore == INACTIVE)

newAuction()

assert(auctionStateAfter == OPEN)

* + 1. **bid()**

require(auctionStateBefore == OPEN|| auctionStateBefore==BID\_PLACED)

bid()

assert(auctionStateAfter == BID\_PLACED)

* 1. close()

require(auctionStateBefore == OPEN|| auctionStateBefore==BID\_PLACED)

close()

assert(auctionStateAfter ==INACTIVE)