0.1 Implementation Details

Algorithm 1 Escalate Algorithm

```
procedure ESCALATE($leftBound, rightBound$)    ▷ Function called when the query has yielded a false positive
    CREATEBOUND($leftBound, true$)
    CREATEBOUND($rightBound, false$)
    MARKEMPTY($leftBound, rightBound$)    ▷ Marks the range created as empty
end procedure
```

Algorithm 2 Escalate Algorithm

```
procedure CREATEBOUND($bound, isLeft$)
    $node \leftarrow$ NAVIGATE($bound$)    ▷ Get the information of the leaf node that contains the bound
    if ($isLeft$ and $node.left == bound$) or ($!isLeft$ and $node.right == bound$) or $node.leafValue == false$ then
        return    ▷ A leaf with the needed left or right bound already exists or the leaf with the bound is already empty
    end if
    while $true$ do
        SPLIT($node$)
        if $isLeft$ and $node.rightChild.left == bound$ then
            return
        end if
        if $!isLeft$ and $node.leftChild.right == bound$ then
            return
        end if
        if CONTAINS($node.leftChild, bound$) then    ▷ Continue splittling in the leaf that contains the bound
            $node \leftarrow node.leftChild$
        else
            $node \leftarrow node.rightChild$
        end if
    end while
end procedure
```

Algorithm 3 Deescalate Algorithm

```
procedure DEESCALATE($targetSize$)
    $startIdx \leftarrow 0$
    while $size \geq targetSize$ do
        TRUNCATE($startIdx$)
    end while
end procedure
```

Algorithm 4 Deescalate Algorithm

```
procedure TRUNCATE($currIdx$)
    if ISLEAF($currIdx$) then
        DECREMENTUSED($currIdx$)
        return
    end if
    $leftChild \leftarrow$ GETLEFTCHILD($currIdx$)
    $rightChild \leftarrow$ GETRIGHTCHILD($currIdx$)
    if ISLEAF($leftChild$) and ISLEAF($rightChild$) then
        if GETUSED($leftChild$) $==$ 0 and GETUSED($rightChild$) $==$ 0 or LEAFVALUE($leftChild$) $==$ LEAFVALUE($rightChild$) then q    ▷ If both leaves are unused or have the same value, they can be merged
            MERGECHILDREN($currIdx, leftChild, rightChild$)
        else
            break    ▷ Otherwise, call truncate for each child below
        end if
    end if
    TRUNCATE($leftChild$)
    TRUNCATE($rightChild$)
end procedure
```