Algorithm 1 Block creation

function $BlockCreation()$
    $TODO$

The algorand protocol specs don't ensure a specific way or internal order in which transactions are added to the block. In fact, they don't force transactions to be added at all, being entirely possible for a node to propose an empty block. More research is needed on how block data is put together. Block metadata is well documented, but is added into it later in the block proposal stage.

Algorithm 2 Block proposal

function $BlockProposal(Block\ B,\ Accounts\ A)$
    $, lowestPriority$
    for $a \in A$ **do**
        $< sorthash, \pi, j > \leftarrow Sortition(a.sk, seed, t, role, a.w[], W)$
        if $j > 0$ **then**
            $priority \leftarrow Min_{n \in [1,j]} Hash(sorthash||n)$
            $lowestPriorityUser \leftarrow a$
            $GOSSIP\_MESSAGE(priority, SIGNED_{a.sk}(< sorthash, \pi >))$
    $msgs \leftarrow incomingMsgs[round, step = 0]$
    while $elapsed\ time < proposal\ time$ **do**
        $incomingMsgs.push(listen())$
    $lowestHashMsg \leftarrow m \mid m.priority = Min_{m.priority\ \in\ incomingMsgs}\{verifySortition(m.sorthash, m.pi, m.pk)$
    $seed \leftarrow ComputeSeed(r, lowestHashMsg)$
    $FullBlockToPropose \leftarrow empty_block(round)$
    $FullBlockToPropose.seed \leftarrow seed$
    while $elapsedtime < fullblocktime$ **do**
        $m \leftarrow msgs.next()$
        $< priority, sorthash, \pi > \leftarrow m$

PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT

Algorithm 3 Soft Vote

function $SoftVote$
    $hblock \leftarrow CountVotes(ctx, round, REDUCTION\_ONE, Tstep, tstep, lblock + lstep)$
    $empty\_hash \leftarrow H(Empty(round, H(ctx.last\_block)))$
    if $hblock = TIMEOUT$ **then**
    $CommitteeVote(ctx, round, REDUCTION\_TWO, tstep, empty\_hash)$ **else**
    $CommitteeVote(ctx, round, REDUCTION\_TWO, tstep, hblock1)$
    $hblock \leftarrow CountVotes(ctx, round, REDUCTION\_TWO, Tstep, tstep, lblock + lstep)$
    if $hblock = TIMEOUT$ **then return** $empty\_hash$ **else return** $hblock$

The soft vote stage (step 1, or labeled as reduction in the ASBAC paper) aims to reduce any ammount of potentially conflicting proposed blocks in the stage prior into a binary choice, either a block proposed by a user or an empty one. In the first sub-step, each commitee member gossips the proposed block, and then does a preliminary vote count while it waits for other nodes to catch up. In the second sub-step, committee members vote for the hash that received at least the ammount of votes set by the threshold for this step, or the hash of the default empty block if no

hash was observed to receive enough votes. It relies heavily on the assumption that, if the block proposer was honest, most users will get to the soft vote stage with the same hblock parameter. However if the proposer was dishonest, then no single hblock may be popular enough to cross the threshold (and therefore this stage will return an empty_hash). Therefore, there will be at most one non-empty block that can be returned by soft vote for all honest users.

Algorithm 4 CertifyVote

```
1:  function CertifyVote(hblock)
2:      while step < 256 do
3:
4:          < hblock, bConfirmed >← BLOCK_STEP(hblock, step)
5:          if bConfirmed then return hblock
6:          step++
7:
8:          < hblock, bConfirmed >← EMPTY_STEP(hblock, step)
9:          if bConfirmed then return hblock
10:         step++
11:
12:         CommonCoinFlipVote(hblock, step)
13:         step++
14:
15:         HangForever()
16:         =0
```

The certify vote stage (which potentially encompasses steps 2 through 255) is the core of the BA* algorithm. PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXTPLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT PLACEHOLDER TEXT

Algorithm 5 BlockConfirmation

```
1:  function BlockConfirmation(hblock)
2:      r ← CountVotes(ctx, round, FINAL, Tfinal, tfinal, lstep)
3:      if r = hblock then
4:      return < FINAL, BlockOfHash(hblock) > else
5:      return < TENTATIVE, BlockOfHash(hblock) >
    =0
```

Finally, the proposed block (be it an empty block or a specific one proposed by a designated user) is confirmed and added to the ledger. We introduce the notions of FINAL and TENTATIVE consensus. Specifically, if a block is confirmed in the very first step of the certify vote stage, its vote is cast as FINAL. If enough users reproduce this behavior, the vote count on line 2 for votes cast in this fashion is over the required threshold, and the consensus is labeled as FINAL. This means that the block is simultaneously added to the ledger and confirmed, as well as immediately confirming any prior blocks that could have been labeled TENTATIVE. If this node is not able to ensure that enough users considered the block represented by hblock as FINAL, the consensus is labeled as TENTATIVE, and it may be confirmed by a later FINAL block. The BlockOfHash() function outputs the full block for the provided hash, either by returning a locally available copy or by requesting it from other users.