```
12    public Response apply(Invoc invoc) {
13      int i = ThreadID.get();
14      < announce[i] = new Node(invoc); start(i) = max(head); concur(i) = {}; >
15      head[i] = Node.max(head);
16      while (announce[i].seq == 0) {
17        Node before = head[i];
18        Node help =  announce[(before.seq + 1) % n];
19        if (help.seq == 0)
20          prefer = help;
21        else
22          prefer = announce[i];
23        Node after = before.decideNext.decide(prefer);
24        before.next = after;
25        after.seq = before.seq + 1;
26        < head[i] = after; (∀j) (concur(j) = concur(j) ∪ {after}); >
27      }
28      < head[i] = announce[i]; (∀j) (concur(j) = concur(j) ∪ {after}); >
29      SeqObject MyObject = new SeqObject();
30      Node current = tail.next;
31      while (current != announce[i]){
32        MyObject.apply(current.invoc);
33        current = current.next;
34      }
35      return MyObject.apply(current.invoc);
36    }
```

FIGURE 6.8 The apply() method of the wait-free universal construction with auxiliary variables. Operations in angled brackets are assumed to happen atomically.