



FIGURE 7.30A The CompositeLock class: an execution. In part (a), thread A (which acquired Node 3) is in the critical section. Thread B (Node 4) is waiting for A to release the critical section, and thread C (Node 1) is in turn waiting for B. Threads D and E are backing off, waiting to acquire a node. Node 2 is free. The tail field refers to Node 1, the last node to be inserted into the queue. At this point, B times out, inserting an explicit reference to its predecessor, and changing Node 4's state from WAITING (denoted by W) to ABORTED (denoted by A). In part (b), thread C cleans up the ABORTED Node 4, setting its state to FREE and following the explicit reference from 4 to 3 (by redirecting its local myPred field). It then starts waiting for A (Node 3) to leave the critical section. In part (c), E acquires the FREE Node 4, using compareAndSet() to set its state to WAITING. Thread E then inserts Node 4 into the queue, using compareAndSet() to swap Node 4 into the tail, then waiting on Node 1, which was previously referred to by tail.