```
41  void commitTx() {
42    if (writes.empty()) {
43      reads.clear();
44      return;
45    }
46    for (auto &l : writes) {
47      atomic<uint64_t>* l = get_lock(l.first);
48      uint64_t prev = *l;
49      if ((prev&1 == 0) && (prev <= start_time)) {
50        if (!l->compare_exchange_strong(prev, my_lock))
51          abortTx();
52        locks.push_back(l, prev);
53      }
54      else if (prev != my_lock) {
55        abortTx();
56      }
57    }
58    uint64_t end_time = ++clock;
59    if (end_time != start_time + 1) {
60      for (auto l : reads) {
61        uint64_t v = *i;
62        if (((v&1) && (v != my_lock)) || ((v&1==0) && (v>start_time)))
63          abortTx();
64      }
65    }
66    for (auto w : writes)
67      std::atomic_ref<uintptr_t>(*w.first).store(w.second, std::memory_order_release);
68    for (auto l : locks)
69      *l.first = end_time;
70    writes.clear();
71    locks.clear();
72    readset.clear();
73  }
74  void abortTx() {
75    for (auto l : locks)
76      *l.first = l.second;
77    reads.clear();
78    writes.clear();
79    locks.clear();
80  }
```

FIGURE 20.12 Software transactions with ownership records (2/2).