Contents

Chapter 4

Accessing State

Weakly isolated concurrent programs are more difficult to write than strongly isolated concurrent programs. Transactional Memory systems are weakly isolated so they have complex concurrent semantics and are prone to pathologies which make their run-time behaviour unpredictable. This chapter describes how isolating shared state in linearizable objects provides a concurrent programming model that has intuitive concurrent semantics and that is not prone to isolation pathologies.

Section 4.1 identifies weak isolation as the reason why Transactional Memory systems have complex concurrent semantics and are prone to pathologies.

Section 4.2 describes how Immutable Data Structures can implement linearizable objects.

Section 4.3 describes a check pointing technique which relies on the composition of Immutable Data Structures.

Section 4.4 compares a concurrent application which calculates the minimum spanning tree of a graph with a similar application which uses Transactional Memory.

Bibliography