В статье «The Problem with Threads» Эдвард Ли подвергает критике традиционный подход к параллельному программированию с использованием потоков. Он объясняет, что потоки создают высокий уровень недетерминизма, что делает поведение программ непредсказуемым и затрудняет их тестирование. Потоки, по его мнению, разрушают логику последовательности, к которой привыкли программисты, и превращают отладку в поиск «иголки в стоге сена». Вместо этого он предлагает ориентироваться на детерминированные модели взаимодействия — такие как сети процессов, синхронное взаимодействие (rendezvous) и языки координации, где взаимодействие компонентов происходит строго по определённым правилам, что позволяет легче анализировать поведение системы.

Для меня, как студента, эта статья стала откровением. Я осознал, что то, что считается «стандартом» — далеко не всегда эффективно. Мы часто воспринимаем многопоточность как нечто необходимое для скорости и масштабируемости, но на практике она ведёт к неуловимым ошибкам и хаосу в логике. Предложения Ли логичны и своевременны: в эпоху многоядерных систем нам действительно нужны модели, которые можно анализировать формально, а не «на удачу». Особенно ценно, что он не просто критикует, а предлагает альтернативы, которые уже реализованы в таких системах, как Ptolemy II. Его подход подсказывает, что устойчивое программное обеспечение должно начинаться с чёткого проектирования модели взаимодействия — а не с добавления блокировок и синхронизаций «по ходу дела».

Эта статья заставила меня пересмотреть собственное понимание параллелизма. Почему мы всё ещё обучаем студентов писать многопоточные программы, зная, насколько они уязвимы? Возможно, потому что легко объяснить, но трудно масштабировать. Мне кажется, преподавание должно меняться: мы должны сначала показывать модели с детерминированным поведением (например, на базе потоков данных или акторов), а лишь затем — демонстрировать риски и ограничения потоков. Кроме того, идея об использовании языков координации как надстройки над традиционными языками программирования кажется чрезвычайно перспективной. Это откроет путь к более понятным и безопасным системам. Я убеждён, что за архитектурами, построенными на принципах детерминизма, — будущее надёжного программного обеспечения.