В статье, написанной в 2006 году, Эдвард А. Ли критикует потоковую модель параллелизма. Автор утверждает, что, несмотря на господствующее положение потоков в мире программирования, они не подходят для разработки сложных, надежных и предсказуемых параллельных программ. Потоковая модель по своей природе является недетерминированной, и её поведение трудно контролировать и предсказать. По мнению автора, параллельный код на потоках настолько сложен, что разобраться в нём почти невозможно — он даже утверждает, что программисты многопоточных систем фактически « безумны». «Разве смогли бы они понимать свои программы, если были бы нормальными ?!» Попытки сделать эту модель более управляемой — с помощью шаблонов проектирования, атомарных операций, транзакций и других средств — лишь облегчают работу программиста, но не устраняют саму проблему.

Ли настаивает на необходимости перехода к другим моделям параллелизма, в основе которых лежит детерминированный подход. Недетерминизм, по его мнению, допустим, но только в тех случаях, когда он используется осознанно и явно. Он формулирует основной принцип: понятные цели должны достигаться понятными средствами. Эдвард считает, что “deterministic ends should be accomplished with deterministic means”» — потоки должны использоваться только в системном программировании и быть скрыты от прикладного разработчика за более высокоуровневыми абстракциями.

Когда Эдвард Ли писал эту статью, он во многом предугадал, куда будет двигаться индустрия. Уже тогда он довольно точно подметил, что потоковая модель — это не просто неудобный, но и небезопасный инструмент. С годами это только подтвердилось: несмотря на улучшение языков и появление новых инструментов, проблемы вроде гонок данных, взаимных блокировок и сложной отладки по-прежнему доставляют много головной боли разработчикам.

Идея отказаться от работы с потоками в лоб и использовать более простые, детерминированные подходы со временем стала очень востребованной. Это хорошо видно по росту популярности горутин, корутин, async/await и появлению новых языков вроде Go или Rust, где параллелизм встроен в саму модель языка и не требует ручной работы с потоками.Правда, мысль о создании отдельного координационного языка, о чём говорил Ли, так и не прижилась. Вместо этого разработчики предпочли внедрять нужные абстракции прямо в язык, делая параллельное программирование частью повседневной практики. Это, скорее, подтверждает, что идеи автора были не просто актуальны, а действительно полезны — хотя и реализовались немного иначе, чем он предполагал.

Я понимаю, о чём говорит автор: потоки — слишком низкий уровень для того, чтобы на них можно было удобно и безопасно строить массовые параллельные программы. Но при этом, на мой взгляд, полностью избавиться от потоков — утопия. Они глубоко встроены в инфраструктуру и до сих пор нужны. Другое дело, что работать с ними напрямую действительно неудобно. Мне нравится, как этот вопрос решён в языке Go. Там, по сути, всё равно используются потоки, но поверх них построен собственный планировщик, который управляет горутинами. Это такая простая и понятная абстракция, и именно она снимает с программиста необходимость копаться в тонкостях синхронизации.

Вообще, видно, что направление развивается: async/await сильно упростил жизнь при написании асинхронного кода, акторы помогают изолировать состояние и избежать лишних блокировок, а в Rust система владения памятью не даёт забыть про безопасность — приходится думать головой. Но вот что важно: полностью убрать недетерминизм не получится. Мир вокруг нас сам по себе конкурентный, всё происходит параллельно, события не ждут друг друга. Поэтому цель, как по мне, не в том, чтобы уничтожить недетерминизм, а в том, чтобы держать его под контролем — не везде, а там, где это действительно нужно.