Идеи реализации

Очевидно, что так как мы хотим покрыть максимальное число вариантов выполнения алгоритма, то стратегия выделения интересных инструкций во время запуска на определенных данных будет тупиковой по причине того, что потребуется перебрать много вариантов входных данных (сколько? Никто не знает, как вариант пока граф не перестанет расти, но могут быть выбраны такие данные, что он вообще не будет меняться).
=> Имеет смысл построить граф основываясь на статическом анализе исходного кода.

Как выбирать какие паттерны проверять первоочередно? В процессе статического анализа можно ввести метрику «расстояние между инструкциями» (почти как в Ctrigger). Логично считать, что сперва проверяем те паттерны, для которых расстояние максимально (больше вероятность прерывания).

Сколько паттернов проверять за раз? А почему бы не начать с 1? Да, будет работать несколько дольше, но есть шанс, что таким образом мы будем максимально приближены к условию линеаризуемости. И, возможно, сможем приблизительно указывать, где ошибка.

Каким образом запускать всё это безобразие? Мне кажется, что вариант «в одном потоке» лучше так как потом на другом потоке можно будет запустить проверку другого паттерна (не уверен в этом, просто предположение). В любом случае, запускать многопоточный алгоритм и заставлять остальные потоки ждать — кощунство.

Что делать с вариантами, которые не удалось проверить? К примеру, внутри какого-то if была интересная ветка. Считать, что она получится при других данных. Как вариант, хранить в какой-нибудь бд или что-то вроде того.